Do Desenvolvimento em Cascata para Iterativo— Uma Transição Desafiadora para Coordenadores de Projeto

Philippe Kruchten

Rational Software White Paper

TP 173, 5/00

Rational®
the software development company

Índice Analítico

Introdução

O RUP (Rational Unified Process) defende uma abordagem interativa ou espiral para o ciclo de vida de desenvolvimento de software, porque essa abordagem tem se comprovado superior à abordagem em cascata em muitos respeitos. Mas não acredite, nem por um segundo, que os vários benefícios que um ciclo de vida iterativo fornece venham gratuitamente. O desenvolvimento iterativo não é uma varinha de condão que quando balançada resolve todos os problemas ou dificuldades possíveis no desenvolvimento de software. Os projetos não são fáceis de configurar, planejar ou controlar apenas porque eles são iterativos. O coordenador de projetos realmente terá uma tarefa mais desafiadora, especialmente durante seu primeiro projeto iterativo, e certamente durante as iterações iniciais desse projeto, quando os riscos são altos e as primeiras falhas possíveis. Neste artigo, descrevi alguns dos desafios do desenvolvimento iterativo a partir da perspectiva do coordenador de projetos. Descrevi também algumas das "armadilhas" ou imprevistos comuns que nós, na Rational, notamos que os coordenadores de projeto caem, através de nossa experiência em consultoria ou de relatórios e estórias de guerra de nossos colegas da Rational.

Desenvolvimento Iterativo

Os processos clássicos de desenvolvimento de software seguem o ciclo de vida em cascata, conforme ilustrado na seguinte figura. Nessa abordagem, mostrada na Figura 1, o desenvolvimento prossegue linearmente da análise de requisitos aos designs, teste de código e unidade, teste de subsistema e teste de sistema, com feedback limitado nos resultados das fases anteriores.

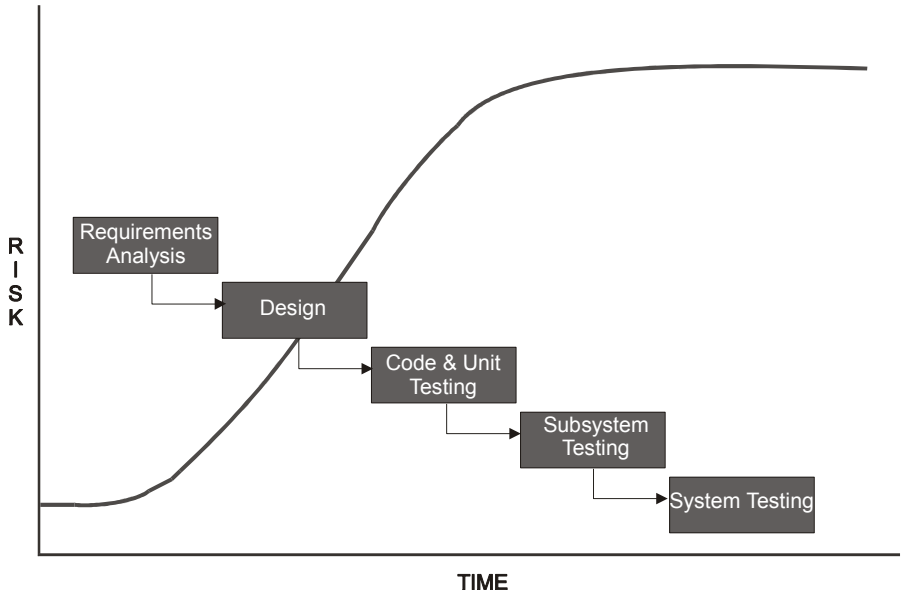

Figura 1: O Processo de Desenvolvimento em Cascata

O problema fundamental dessa abordagem é que ela gera riscos com o tempo, e fica caro desfazer os enganos das fases anteriores. Um design inicial provavelmente terá falhas com relação aos seus requisitos-chave e, além disso, a descoberta tardia de defeitos de design tende a resultar em overruns caros e/ou cancelamento do projeto. A abordagem em cascata tende a mascarar os riscos reais para um projeto até ser muito tarde para fazer algo significativo com eles.

Uma alternativa à abordagem em cascata é o processo iterativo e incremental, conforme mostrado na Figura 2.

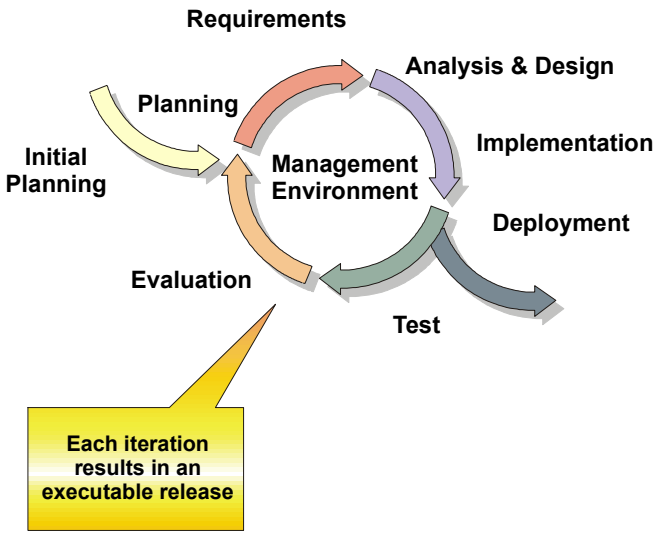

Figura 2: Uma Abordagem Iterativa para Desenvolvimento

Nessa abordagem, construída sobre o trabalho do modelo espiral de Barry Boehm (consulte "Leitura Adicional"), a identificação de riscos em um projeto é forçada precocemente no ciclo de vida, onde é possível atacar e reagir a eles a tempo e de maneira eficiente. Essa abordagem é uma descoberta, uma invenção e uma implementação contínua com cada iteração forçando a equipe de desenvolvimento a conduzir o fechamento dos artefatos do projeto de forma previsível e repetível.

A Parte Boa: Os Benefícios do Desenvolvimento Iterativo

Comparado com o processo em cascata tradicional, o processo iterativo tem muitas vantagens.

1. Interpretações incorretas sérias ficam evidentes no início do ciclo de vida, quando é possível reagir a eles.
2. Ele permite e incentiva o feedback do usuário, para fazer o levantamento dos requisitos reais do sistema.
3. A equipe de desenvolvimento é forçada a se concentrar nesses problemas que são mais críticos para o projeto e os membros da equipe ficam protegidos desses problemas que os distraem dos riscos reais dos projetos.
4. O teste contínuo e iterativo permite uma avaliação objetiva do status do projeto.
5. As inconsistências entre requisitos, designs e implementações são detectadas precocemente.
6. A carga de trabalho da equipe, especialmente a equipe de teste, é distribuída mais uniformemente através do ciclo de vida.
7. Essa abordagem permite que a equipe alavanque lições aprendidas e, portanto, aprimorem continuamente o processo.
8. Os envolvidos no projeto podem receber evidência concreta do status do projeto em todo o ciclo de vida.

Mitigação de Riscos

Um processo iterativo permite mitigar riscos antecipadamente porque a integração é, geralmente, a única hora em que os riscos são descobertos ou enfocados. Conforme caminha para as iterações precoces, você passa por todos os componentes do processo, valendo-se de muitos aspectos do projeto, incluindo ferramentas, software off-the-shelf e habilidades pessoais. Os riscos aparentes provarão não ser riscos e riscos novos e insuspeitos serão descobertos.

Se um projeto precisar ser reprovado por alguma razão, deixe que isso aconteça logo, antes de gastar muito tempo, esforços e dinheiro. Não fuja dos problemas; confronte os riscos. Entre outros riscos, como construir o produto errado, existem duas categorias de riscos que um processo de desenvolvimento iterativo ajudar a mitigar precocemente:

- Riscos de Integração

- Riscos arquiteturais

Um processo iterativo resulta em uma arquitetura mais robusta, pois os erros são corrigidos após várias iterações. As falhas são detectadas em situações precoces, conforme o produto se move além da iniciação. Os gargalos de desempenho são descobertos a tempo, quando ainda podem ser enfocados, em vez de serem descobertos na véspera da entrega.

A integração não é um "big bang" no final do ciclo de vida; em vez disso, os elementos são integrados progressivamente. Na verdade, a abordagem iterativa que recomendamos envolve uma integração quase contínua. O que costumava ser um período longo de incertezas e dificuldades—que levava até 40% do esforço total no final de um projeto—agora é dividido em seis a nove integrações menores que iniciam com um número muito menor de elementos a serem integrados.

Acomodando Alterações

Você pode prever várias categorias de alterações:

- Alterações em Requisitos

  Um processo iterativo permite que você leve em conta a alteração de requisitos. A verdade é que os requisitos normalmente serão alterados. As alterações nos requisitos e a "ascensão lenta dos requisitos" têm sido sempre as principais fontes de problemas para um projeto, levando à atraso de entrega, à perda de data, a clientes insatisfeitos e a desenvolvedores frustrados. Mas ao expor os usuários (ou representantes de usuários) a uma versão precoce do produto, você assegura uma melhor adaptação do produto à tarefa.

- Alterações Táticas

  Um processo iterativo fornece gerenciamento com uma maneira de fazer alterações táticas no produto—por exemplo, competir com produtos existentes. Você pode decidir lançar um produto antecipadamente com funcionalidade reduzida como reação a uma movimentação de um concorrente ou poderá adotar um outro fornecedor de uma determinada tecnologia. Você também pode organizar o conteúdo de uma iteração para aliviar um problema de integração que precisa ser corrigido por um fornecedor.

- Alterações Tecnológicas

  Em uma extensão menor, uma abordagem iterativa permite acomodar alterações tecnológicas. Você pode utilizá-la durante a fase de elaboração, mas deve evitar esse tipo de alteração durante a construção e transição porque ela é inerentemente arriscada.

Aprendendo Durante o Caminho

Uma vantagem do processo iterativo é que os desenvolvedores podem aprender durante o caminho e as várias capacidades e especialidades são empregadas mais completamente durante o ciclo de vida inteiro. Por exemplo, os testadores iniciam o teste cedo, os técnicos escrevem cedo e assim por diante, enquanto em um desenvolvimento não-iterativo, as mesmas pessoas estarão aguardando o começo de seus trabalhos, fazendo plano após plano. As necessidades de treinamento—ou a necessidade de ajuda adicional (talvez externa)—são apontadas precocemente durante as revisões de avaliação.

O próprio processo também pode ser aprimorado e refinado durante o caminho. A avaliação ao final de uma iteração examina o status do projeto de uma perspectiva do produto/planejamento e analisa o que precisa ser alterado na organização e no processo para melhorar o desempenho na próxima iteração.

Maior Oportunidade de Reutilização

Um processo iterativo facilita a reutilização de elementos do projeto porque é mais fácil identificar as partes comuns quando estão parcialmente projetadas ou implementadas em vez de identificar todas as semelhanças no início. É difícil identificar e desenvolver partes reutilizáveis. As revisões de design nas iterações iniciais possibilitam que os arquitetos identifiquem reutilizações insuspeitas e potenciais para que desenvolvam e amadureçam o código comum em iterações subseqüentes. É durante as iterações na fase de elaboração que as soluções comuns para problemas comuns são localizadas e os padrões e mecanismos arquiteturais que se aplicam no sistema são identificados.

Qualidade Global Melhor

O produto que resulta de um processo iterativo terá uma qualidade global melhor do que os produtos que resultam de um processo seqüencial convencional. O sistema foi testado várias vezes, aprimorando a qualidade de teste. Os requisitos terão sido refinados e estarão mais proximamente relacionados às necessidades reais dos usuários. E na hora da entrega, o sistema terá sido executado por mais tempo.

A Parte Difícil: Downside Inesperado e Armadilhas Comuns

O desenvolvimento iterativo não significa necessariamente menos trabalho e planejamentos mais curtos. Sua principal vantagem é que o resultado e o planejamento são mais previsíveis. Ele trará produtos de maior qualidade, que satisfarão as necessidades reais dos usuários finais, porque você terá tempo de desenvolver requisitos, bem como um design e uma implementação.

O desenvolvimento iterativo realmente desenvolve muito mais planejamento e, portanto, dá mais responsabilidade ao coordenador de projeto: um plano global precisa ser desenvolvido e planos detalhados serão, por sua vez, desenvolvidos para cada iteração. Ele também envolve negociação contínua entre o problema, a solução e o plano. O planejamento mais arquitetural também ocorrerá antes. Os artefatos (planos, documentos, modelos e código) terão de ser modificados, revistos e aprovados repetidamente em cada revisão. As alterações táticas ou alterações em escopo forçarão algum replanejamento contínuo. Portanto, a estrutura da equipe terá de ser modificada ligeiramente em cada iteração.

Armadilha: Planejamento Muito Detalhado Até o Final

Normalmente, é anti-econômico construir um plano de ponta a ponta detalhado, exceto como um exercício na avaliação do envelope global de planejamento e recursos. Esse plano ficará obsoleto antes de chegar ao final da primeira iteração. Antes de ter uma arquitetura no local e um controle firme dos requisitos—o que ocorre aproximadamente no marco LCA,—você não poderá construir um plano realístico.

Portanto, incorpore precisão no planejamento comensurado com seu conhecimento da atividade, do artefato e da iteração sendo planejada. Planos de curto prazo são mais detalhados e melhor granulados. Planos de longo prazo são mantidos no formato grosseiro.

Resista à pressão de gerenciamento inexperiente ou mal-informado poderá auxiliar na tentativa de fazer o levantamento de um "plano global abrangente." Treine os coordenadores e explique a noção de planejamento iterativo e o esforço gasto ao tentar prever detalhes no futuro. Uma analogia que é útil: uma viagem de carro de New York a L.A. Você planeja a rota global, mas precisa apenas de instruções detalhadas para sair da cidade e apenas no primeiro trecho da viagem. Planejar os detalhes exatos para guiar até Kansas, sem falar na chegada em Califórnia, é desnecessário, porque você verá que a estrada até Kansas está em conserto e precisará localizar uma rota alternativa, etc.

Reconhecendo o Retrabalho Direto

Em uma abordagem em cascata, ocorre muito retrabalho no final, como uma conseqüência irritante e, freqüentemente, não planejada de localizar erros graves durante o teste e integração finais. E o pior é que você descobre que a principal causa da "interrupção" vem de erros no design, que você tenta contornar na implementação construindo soluções alternativas que levam a mais interrupções.

Em uma abordagem iterativa, você simplesmente reconhece que haverá retrabalho e, inicialmente, muito retrabalho: conforme descobre problemas nos protótipos arquiteturais iniciais, você precisa consertá-los. Além disso, para construir protótipos executáveis, stubs e andaimes terão de ser construídos para serem substituídos posteriormente por implementações mais maduras e robustas. Em um projeto iterativo mais saudável, a porcentagem de scrap ou retrabalho deve diminuir rapidamente; as alterações devem ser menos difundidas conforme a arquitetura se estabiliza e os problemas difíceis estão sendo resolvidos.

Armadilha: Projeto Não Convergente

O desenvolvimento iterativo não significa sucatear tudo em cada iteração. O scrap e o retrabalho devem diminuir de iteração para iteração, especialmente depois da arquitetura ter linha de base no marco LCA. Os desenvolvedores freqüentemente querem levar vantagem do desenvolvimento iterativo para começarem a trabalhar: apresentar uma técnica ainda melhor, executar retrabalho, etc. O coordenador de projeto precisa estar vigilante para não permitir retrabalho dos elementos que não estão interrompidos—que estão OK ou bons o suficiente. Além disso, conforme a equipe de desenvolvimento aumenta de

tamanho, e algumas pessoas estão se mudando, são introduzidos os recém-chegados. Eles tendem a ter suas próprias idéias sobre como as coisas devem ser feitas. Similarmente, os clientes (ou seus representantes no projeto: marketing, gerenciamento de produtos) poderão querer abusar da latitude oferecida pelo desenvolvimento iterativo para acomodar alterações e/ou alterar ou incluir requisitos sem fim. Esse efeito é, às vezes, chamado de "Ascensão Lenta de Requisitos." Novamente, o coordenador de projeto precisa ser implacável ao fazer comercialização e na negociação de prioridades. Em torno do marco LCA, os requisitos têm linha de base e, ao menos que o planejamento e o orçamento sejam renegociados, toda a alteração tem um custo finito: colocar alguma coisa significa tirar outra.

E, lembre-se de que o "O ótimo é inimigo do bom" (Ou em francês: "Le mieux est l'ennemi du bien.")

Armadilha: Vamos Iniciar; Depois Decidimos Aonde Ir

O desenvolvimento iterativo não significa fundir o desenvolvimento eternamente. Você não deve simplesmente começar com o design e a codificação apenas para manter a equipe ocupada ou com a esperança de que objetivos claros surgirão de repente. Você ainda precisa definir objetivos claros, escrevê-los e obter concordância de todos os parceiros; então, refiná-los, expandi-los e obter novamente a concordância. O lado bom é que no desenvolvimento iterativo, você não precisa ter todos os requisitos estabelecidos antes de começar o design, a codificação, a integração, o teste e validação deles.

Armadilha: Vítima que Cai em seu Próprio Sucesso

Um risco interessante ocorre próximo do final de um projeto, no momento que o "consumidor" fica um pouco entusiasmado. Com isso queremos dizer que os usuários vão de um extremo a outro: ou acreditam que nada jamais será entregue ou que a equipe poderá realmente realizar o projeto. A boa notícia é que a percepção externa do projeto mudou: enquanto na segunda-feira os usuários estariam felizes se algo fosse entregue, na terça-feira, eles ficam preocupados que nada será entregue. Essa é a má notícia. Em algum lugar entre o primeiro e o segundo beta, você ficará inundado dos recursos que as pessoas pensam em criar no primeiro release. De repente, eles se tornam os problemas principais. O coordenador de projeto se preocupa desde a entrega de mínima funcionalidade aceita até uma situação na qual cada requisito é agora "essencial" para a primeira entrega. É como se, quando há esse pequeno entusiasmo, todos os itens pendentes fossem elevados para um status de prioridade "A". A realidade é que ainda há o mesmo número de coisas a fazer e a mesma quantidade de tempo. Embora a percepção externa possa ter sido alterada, a priorização ainda é extremamente importante.

Se, nesse momento crucial, o coordenador de projeto não for firme e começar a ceder a todos os pedidos, ele realmente colocará o projeto em perigo de planejamento novamente! É nesse ponto que ele deve se manter implacável e não sucumbir a novos pedidos. Mesmo negociar algo novo por algo que foi retirado pode aumentar o risco neste ponto. Sem vigilância, pode-se chegar à derrota pelas garras do sucesso.

Colocando o Software Primeiro

Em uma abordagem em cascata, há muita ênfase nas "especificações" (isto é, a descrição de espaço do problema) e em deixá-las corretas, completas, refinadas e assinadas. No processo iterativo, o software que você desenvolve vem primeiro. A arquitetura de software (isto é, a descrição de espaço da solução) precisa conduzir decisões precoces do ciclo de vida. Os clientes não compram especificações; é o produto de software que é o principal foco de atenção do princípio ao fim, com as especificações e o software se desenvolvendo em paralelo. Esse foco em "software primeiro" tem algum impacto nas várias equipes: os testadores, por exemplo, podem estar acostumados a receber especificações completas e estáveis, com bastante antecedência para iniciar o teste; enquanto em um desenvolvimento iterativo, eles precisam começar a trabalhar de uma vez, com especificações e requisitos que ainda estão em desenvolvimento.

Armadilha: Muito Foco nos Artefatos de Gerenciamento

Alguns coordenadores dizem, "Sou um coordenador de projeto, portanto, devo me concentrar em ter o melhor conjunto de artefatos de gerenciamento possível; eles são a chave para tudo." Isso não é verdade! Embora um bom gerenciamento seja fundamental, o coordenador de projeto deve assegurar-se no final que o produto final será o melhor que poderá ser produzido. A coordenação de um projeto não é uma prática que protege você mesmo de mostrar que houve falha, apesar do melhor gerenciamento possível. Da mesma forma, você pode se concentrar no desenvolvimento da melhor especificação possível porque foi prejudicado pelo mau gerenciamento de requisitos no passado; isso não terá qualquer utilidade se o produto correspondente contiver erros, for lento, instável e frágil.

Atacando Problemas Difíceis Mais Cedo

Em uma abordagem em cascata, muitos dos problemas difíceis, as coisas arriscadas e as incertezas reais são deixadas de lado no processo de planejamento, para resolução durante a temida atividade de integração do sistema. Isso faz com que a primeira metade do projeto seja relativamente confortável, na qual os problemas são tratados no papel, por escrito, sem envolver muitos interessados (testadores, etc.), plataformas de hardware, usuários reais ou o ambiente real. E, então de repente, o projeto entra no terror da integração e tudo fica perdido. No desenvolvimento iterativo, o planejamento baseia-se principalmente nos riscos e incertezas, portanto, as coisas ficam resolvidas desde o começo. Alguns problemas técnicos, difíceis, críticos e, freqüentemente, de nível baixo precisam ser tratados imediatamente, em vez de adiados. Resumindo, como alguém me disse uma vez: em um ambiente iterativo você não pode mentir (para você mesmo e para o mundo) por muito tempo. Um projeto de software destinado para falha deve conhecer seu destino mais cedo em uma abordagem iterativa.

Uma analogia é um curso universitário no qual o professor passa a primeira metade do semestre ensinando conceitos relativamente básicos, dando a impressão que é uma matéria fácil que permite que os estudantes recebam boas notas na metade do período com mínimo esforço. Então, de repente, ocorre a aceleração conforme chega o fechamento do semestre. O professor apresenta todos os tópicos desafiadores pouco antes do exame final. Nesse ponto, o cenário mais comum é a maior parte da classe sob pressão, indo lamentavelmente mal no exame final. É interessante como professores inteligentes são surpreendidos com esse desastre repetido, anos após anos, classe após classe. Uma abordagem mais inteligente seria dar um volume inicial do curso, dando 60% do trabalho antes da metade do período, incluindo algum material desafiador. A correlação para gerenciar um projeto iterativo é não perder tempo precioso no início solucionando problemas não existentes e executando tarefas triviais. A razão mais comum para falha técnica nas inicializações: “Eles passaram todo o tempo fazendo a coisa fácil.”

Armadilha: Adiando o Problema

É freqüente tentar dizer, “É um problema delicado para o qual precisamos de mais tempo para pensar. Vamos adiar sua resolução, com isso teremos mais tempo para pensar.” O projeto então embarca em todas as tarefas fáceis, nunca dedicando muita atenção ao problemas difíceis. Quando ele chega no ponto em que é necessária uma solução, soluções e decisões precipitadas são tomadas ou o projeto descarrila. Você deseja fazer o oposto: lidar com as coisas difíceis imediatamente. Eu às vezes digo, “Se um projeto precisar ser reprovado por alguma razão, deixe que isso aconteça logo, antes de gastar muito tempo, esforços e dinheiro.”

Armadilha: Esquecendo-se de Novos Riscos

Você executou uma análise de riscos na iniciação e a utilizou para o planejamento, mas, então, esqueceu-se dos riscos desenvolvidos posteriormente no projeto. E eles voltaram para prejudicá-lo posteriormente. Os riscos devem ser reavaliados de forma constante, semanalmente ou mensalmente. A lista original de riscos que você desenvolveu era apenas provisória. É apenas quando a equipe começa a fazer o desenvolvimento concreto (software primeiro) que eles descobrem vários outros riscos.

Confrontos Devido a Modelos de Ciclo de Vida Diferentes

O coordenador de um projeto iterativo freqüentemente verá confrontos entre seu ambiente e de outros grupos, como gerenciamento principal, clientes e contratadas, que não adotaram—ou entenderam a natureza—do desenvolvimento iterativo. Eles esperam artefatos concluídos e congelados em marcos-chave; não desejam revisar requisitos em pequenas instalações; ficam chocados com o retrabalho; e não entendem a finalidade ou o valor de algum protótipo arquitetural feio. Eles acham que a iteração é algo que atrapalha, não tem finalidade, joga contra a tecnologia, desenvolve código antes das especificações estarem firmes e testa código desperdiçado.

No mínimo, torna suas intenções e planos claramente visíveis. Se a abordagem iterativa estiver apenas em sua mente e em alguns whiteboards compartilhados com sua equipe, você terá problemas mais tarde.

O coordenador do projeto deve proteger a equipe de ataques e políticas externos para evitar que o mundo externo separe ou desincentive a equipe. Ele deve agir como um pára-choque. Para ser “a mão firme na cana do leme,” o coordenador do projeto deve construir confiança e confiabilidade dentro da comunidade externa. Portanto, visibilidade e “acompanhamento do plano” ainda são importantes, especialmente levando-se em consideração que “o plano” é algo não convencional na visão de algumas pessoas. De fato, é realmente mais importante.

Armadilha: Grupos Diferentes Operando em seus Próprios Planejamentos

É melhor e mais fácil que todos os grupos (equipes ou subcontratadas) estejam operando de acordo com a mesma fase e plano de iteração. Freqüentemente, os coordenadores de projeto vêem alguma otimização de tempo ao adequar melhor o planejamento de cada equipe individual, em que cada uma acaba tendo o seu próprio planejamento de iteração. Quando isso acontece, todos os benefícios aparentes serão perdidos posteriormente e as equipes serão forçadas a se sincronizarem com o grupo mais lento. O quanto for viável, coloque todos no mesmo ritmo.

Armadilha: Licitação de Preço Fixo Durante a Iniciação

Muitos projetos precisam de licitação para desenvolvimento contratual muito cedo, em algum lugar no meio da iniciação. Em um ambiente iterativo, o melhor ponto no tempo para todas as partes fazerem tal licitação é o marco LCA (final da elaboração). Não há nenhuma receita mágica aqui: é necessária alguma negociação e experiência dos envolvidos, mostrando os benefícios de um desenvolvimento iterativo e, eventualmente, um processo de licitação e duas etapas.

A Contabilidade de Progresso É Diferente

O sistema tradicional de valor atribuído para contabilizar o progresso é diferente, porque os artefatos não estão mais completos e congelados, mas são retrabalhados em vários incrementos. Se um artefato tiver um determinado valor no sistema de valor atribuído e você obtiver crédito para ele na primeira iteração em que o criou, então, sua avaliação de progresso será excessivamente otimista. Se você obtiver crédito apenas quando ele ficar estável em duas ou três iterações posteriores, sua medida de progresso se tornará deprimentemente pessimista. Portanto, ao utilizar uma abordagem para monitorar o progresso, os artefatos devem ser decompostos em partes: por exemplo, documento inicial (40%), primeira revisão (25%), segunda revisão (20%), documento final (15%). Para cada parte deve ser alocado um valor. Você pode então utilizar o sistema de valor atribuído sem precisar completar cada elemento.

Um alternativa seria organizar o valor atribuído em torno das próprias iterações e medir o valor a partir dos critérios de avaliação. Então, os pontos de rastreio intermediários (normalmente mensalmente) relatados na Avaliação de Status seriam construídos em torno do Plano de Iteração. Isso requer um rastreio de artefatos mais granulado do que as tradicionais especificações de requisitos, especificações de design, etc., porque você está rastreando a conclusão de vários casos de uso, casos de teste e assim por diante.

Como Walker Royce diz, “Um coordenador de projeto deve estar mais focado na medida e monitoramento de alterações: alterações em requisitos, no design e no código do que na contagem de páginas de texto e linhas de código.” (Consulte “Leitura Adicional” abaixo.) E Joe Marasco adiciona, “Tome cuidado não apenas com as alterações, mas também com a grande quantidade delas. Coisas que são alteradas várias vezes para retornarem ao mesmo ponto inicial são um sintoma de problemas mais profundos.”

No lado positivo, ter um software concreto que logo é executado pode ser utilizado pelo sábio coordenador de projeto para obter alguns pontos de credibilidade iniciais. Isso pode mostrar progresso de uma maneira mais significativa do que o trabalho em papel concluído e revisado com centenas de caixas de opções marcadas. Além disso, os engenheiros preferem “demonstrações de como ele trabalha” para “obter documentação de como ele deveria trabalhar.” Demonstrar primeiro, depois documentar.

Decidindo o Número, a Duração e o Conteúdo das Iterações

O que devemos fazer primeiro? O coordenador que é iniciante no desenvolvimento de iteração, freqüentemente, tem dificuldade para decidir sobre o conteúdo das iterações. Inicialmente, esse planejamento é conduzido por risco, técnico e programático, e pela criticidade das funções ou recursos do sistema sob construção. (O RUP fornece diretrizes para decidir o número e a duração de iterações.) Os critérios também envolvem completamente o ciclo de vida. Na construção, o planejamento é preparado para concluir determinados recursos em determinados subsistemas; na transição, ele é preparado para corrigir problemas e aumentar a robustez e o desempenho.

Armadilha: Exigindo Demais na Primeira Iteração

Acima, falamos sobre não apresentar os problemas difíceis primeiro. Por outro lado, ir muito longe na direção oposta é também uma receita de falha. Há uma tendência de querer enfocar muitos problemas e discutir muitos motivos na primeira ou primeiras iterações. Isso deixa de reconhecer outros fatores: uma equipe precisa ser formada (treinada), novas técnicas precisam ser aprendidas e novas ferramentas precisam ser adquiridas. E, freqüentemente, o domínio de problema é novo para

muitos dos desenvolvedores. Isso, em geral, leva a um excesso grave na primeira iteração, que pode desacreditar a abordagem iterativa inteira. Ou, a iteração é interrompida— declarada concluída quando nada for executado—que é basicamente declarar "vitória" em um ponto no qual nenhuma das lições podem ser esboçadas, perdendo a maioria dos benefícios do desenvolvimento iterativo.

Quando estiver em dúvida ou em crise, diminua isso de alguma forma (isso se aplica ao problema, à solução e à equipe). Lembre-se de que a realização é uma preocupação tardia no ciclo de vida. "A não-realização adequada" deve ser a preocupação inicial do coordenador no ciclo de vida. Se a primeira iteração contiver muitos objetivos, divida-os em duas iterações e, em seguida, priorize implacavelmente quais objetivos irá tentar atingir primeiro.

É melhor atingir um objetivo mais simples e conservador no início do projeto. Note que não dissemos fácil. Ter um resultado adquirido, sólido, no início do processo ajudará a construir a imagem. Muitos projetos que perdem o primeiro marco nunca se recuperam. A maioria que o perde é condenada apesar de esforços heróicos subseqüentes. Planeje para se certificar de que não perderá um marco inicial.

Armadilha: Muitas Iterações

Primeiro, um projeto não deve confundir as construções diárias ou semanais com iterações. Como há uma despesa fixa no planejamento, monitoramento e avaliação de uma iteração, uma organização que não está familiarizada com essa abordagem não deve tentar iterar em uma taxa furiosa em seu primeiro projeto. A duração de uma iteração deve também considerar o tamanho da organização, seu grau de distribuição geográfica e o número de organizações distintas envolvidas. Revisite nosso método empírico "mais seis ou menos três".

Armadilha: Sobrepondo Iterações

Outra armadilha muito comum é sobrepor muitas iterações. Começar a planejar a próxima iteração em algum lugar em direção às últimas quintas partes da iteração atual, enquanto tenta ter uma sobreposição de atividades significativa (isto é, iniciar a análise detalhada, fazer o design e codificar a próxima iteração antes de terminar a atual e aprender com isso) pode parecer atrativo ao olhar um gráfico GANTT, mas levará a problemas. Algumas pessoas não ficarão comprometidas em seguir e concluir a sua própria contribuição com a iteração atual; elas poderão não ficar muito responsivas para corrigir coisas; ou decidirão considerar qualquer feedback, ou todos, apenas na próxima iteração. Algumas partes do software não estarão prontas para suportar o trabalho que foi impulsionado, etc. Embora seja possível desviar alguma mão-de-obra para executar trabalho não relacionado à iteração atual, isso deve ocorrer muito pouco e ser excepcional. Esse problema geralmente é acionado pela habilidade restrita de alguns membros da organização ou em uma organização muito rígida: Joe é um analista e essa é a única cosia que ele pode ou deseja fazer; ele não deseja participar do design, da implementação ou do teste Outro exemplo negativo: Um projeto de comando e controle grande tem suas iterações tão sobrepostas que elas estão basicamente sendo executadas em paralelo em algum ponto no tempo, requerendo que o gerenciamento divida a equipe inteira entre as iterações, sem esperança de feedback das lições aprendidas com as anteriores para as próximas.

Consulte a Figura 3 para ver alguns padrões de iteração não produtiva.

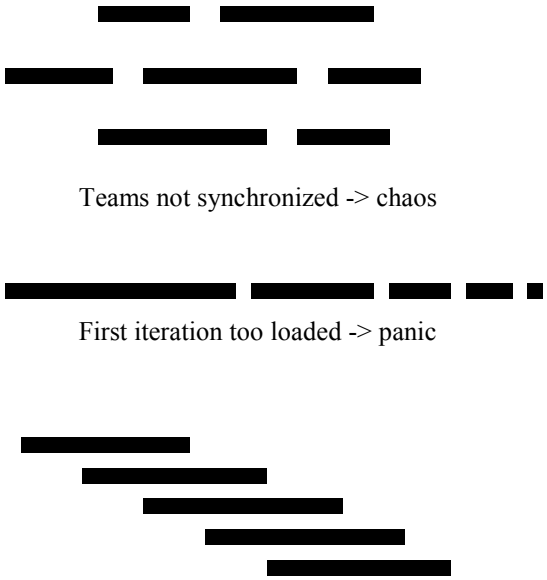

Figura 3: Alguns Padrões de Iteração Perigosos

Um Bom Coordenador de Projeto e um Bom Arquiteto

Para obter sucesso, um projeto de software precisa de um bom coordenador de projeto e um bom arquiteto. O melhor gerenciamento possível e desenvolvimento iterativo não levará a um produto bem-sucedido sem uma boa arquitetura. Inversamente, uma arquitetura fantástica falhará lamentavelmente se o projeto não for bem gerenciado. Portanto, é uma questão de equilíbrio e focar exclusivamente o gerenciamento do projeto não levará ao sucesso. O coordenador de projeto não pode simplesmente ignorar a arquitetura; são necessários o perito em arquitetura e o perito em domínio para determinar os 20% de coisas que devem estar nas iterações iniciais.

Armadilha: Utilizar a Mesma Pessoa como o CP e o Arquiteto

Utilizar a mesma pessoa como o coordenador de projeto *e* o arquiteto não funcionará em projetos pequenos (5-10 pessoas). Para tentativas maiores, ter a mesma pessoa desempenhando a função de coordenador de projeto e arquiteto normalmente fará com que o projeto não seja bem gerenciado nem bem arquitetado. Primeiro, as funções requerem conjuntos de habilidades diferentes. Segundo, as funções internas e as próprias funções são mais do que uma tarefa de tempo integral. Portanto, o coordenador de projeto e o arquiteto devem coordenar diariamente, comunicar-se um com o outro e estar comprometidos. As funções são parecidas com um diretor de cinema e um produtor de cinema. Ambos trabalham visando um objetivo comum, mas são responsáveis por aspectos totalmente diferentes de empreendimento. Quando a mesma pessoa desempenha dois papéis, o projeto raramente é bem-sucedido.

Conclusão

Nesse estágio, você poderá se sentir desincentivado: tantos problemas à frente, tantas armadilhas para cair. Se for muito difícil planejar e executar um desenvolvimento iterativo, por que se chatear? Alegre-se; existem receitas e técnicas para enfocar sistematicamente todos esses problemas e as compensações são maiores do que as inconveniências em termos de obter produtos de software de qualidade confiavelmente alta. Alguns temas-chave: “Ataque os riscos ativamente ou eles irão atacar você.” (Do livro de Tom Gilb, listado em *Referências e Leitura Adicional*.) Software vem primeiro. Reconheça scrap e retrabalho. Escolha um coordenador de projeto e um arquiteto para trabalharem juntos. Explore os benefícios do desenvolvimento iterativo.

O modelo em cascata facilita para o coordenador e dificulta para a equipe de engenharia. O desenvolvimento iterativo está mais alinhado com a forma como os engenheiros de software trabalham, mas com algum custo na complexidade de

gerenciamento. Dado que a maioria das equipes têm uma proporção de 5-para-1 (ou maior) de engenheiros para coordenadores, isso é um ótimo negócio.

Embora o desenvolvimento iterativo seja mais difícil do que as abordagens tradicionais, a primeira vez que você o fizer, haverá uma compensação real de longo prazo. Depois de aprender a fazê-lo bem, você perceberá que se tornou um coordenador bem mais capaz e achará mais fácil gerenciar projetos grandes e mais complexos. Depois de fazer com que a equipe inteira entenda e pense iterativamente, o método será escalado bem melhor do que as abordagens tradicionais.

Nota: John Smith, Dean Leffingwell, Joe Marasco e Walker Royce me ajudaram a escrever este artigo, compartilhando suas experiências no gerenciamento de projeto iterativo. Parte deste artigo está incluída no Capítulo 6 do novo livro de nosso colega Gerhard Versteegen sobre desenvolvimento de software (consulte Referências e Leitura Adicional abaixo).

Sobre o Autor

Philippe Kruchten juntou-se a Rational Software em 1987 e, atualmente, é um Rational Fellow, baseado em Vancouver, B.C. Anteriormente, como Diretor e Gerente Geral da Unidade de Negócios de Processo, ele liderou o desenvolvimento do Rational Unified Process. Além de focar a arquitetura de software e o design, ele está intensamente interessado em práticas de engenharia de software e no processo de desenvolvimento. Ele é formado em engenharia mecânica e possui doutorado em ciência da computação em instituições francesas.

Referências e Leitura Adicional

1. *Rational Unified Process 2000*, Rational Software, Cupertino, Ca., 2000.
2. Barry W. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, Maio de 1988, IEEE, pp.61-72.
3. Tom Gilb, *Principles of Software Engineering Management*, Addison-Wesley, 1988.
4. Philippe Kruchten, *The Rational Unified Process—An Introduction*, Addison Wesley Longman, 1999.
5. Walker Royce, *Software Project Management—A Unified Approach*, Addison Wesley Longman, 1999.
6. Gerhard Versteegen, *Projektmanagement mit dem Rational Unified Process*, Springer-Velag, Berlin, 2000.

Rational®
the software development company

Duas Sedes:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Sem custo: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
Localização Internacional:www.rational.com/worldwide