Uma Comparação de RUP® e XP

John Smith

Rational Software White Paper

TP 167, 5/01

Rational®
the software development company

Índice Analítico

Deleted: 2

Deleted: 5

Deleted: 13

Deleted: 14

Deleted: 14

Deleted: 16

Deleted: 17

Deleted: 18

Deleted: 20

Deleted: 20

Deleted: 20

Deleted: 22

Deleted: 23

Introdução

Este documento compara o RUP (Rational Unified Process®), uma estrutura de processo refinada durante anos pela Rational® Software, que está sendo amplamente utilizada em vários projetos de software, de pequenos a grandes, com o XP (Extreme Programming), uma abordagem de desenvolvimento de software que está ganhando cada vez mais reconhecimento como um método efetivo para construir sistemas menores em um ambiente de requisitos de alteração.

A comparação é direcionada para a natureza e o estilo de cada um como uma descrição de processo e sua estrutura, premissas e apresentação subjacentes. Este documento também examina quais são os destinos potenciais para cada um e as conseqüências de seu uso.

A maioria dos processos têm alguns elementos em comum que tornam possível uma comparação sistemática. Eles requerem seqüências ou grupos de tarefas, que são executadas por funções (normalmente por pessoas que trabalham como indivíduos ou equipes) para gerar artefatos ou produtos de trabalho, dos quais alguns ou todos são entregues a um cliente. A maioria dos processos também reconhece que as instâncias dos processos terão uma dimensão de tempo, com um início e fim, e marcos intermediários interessantes que representam a conclusão de atividades significativas (ou clusters de atividades) e a produção de artefatos associados. Conseqüentemente, este documento explora as seguintes dimensões de processo e as utiliza para fazer a comparação:

- **Alocação de Tempo e Esforços** — discute como cada processo é organizado com o tempo e como a alocação de esforços da equipe é comparado.
- **Artefatos** — compara os produtos de trabalho; aquelas *coisas produzidas* no curso de um projeto baseado no XP e RUP.
- **Tarefas** — discute a forma na qual processo informa que seus artefatos devem ser produzidos.
- **Disciplinas** — compara a forma na qual o XP e o RUP delineiam as principais áreas de preocupação na engenharia de software.
- **Funções** — explora as diferenças entre funções (que executam tarefas) no RUP e funções que estão próximas de posições dentro de uma equipe no XP.

A motivação para esse documento é esclarecer as posições relativas do RUP e do XP no espectro do processo para ver o que cada um pode oferecer ao outro e espalhar a noção de que o XP é uma alternativa *leve* e, portanto, desejável em relação ao RUP *pesado*.

As principais fontes de informações sobre o XP utilizadas ao escrever este documento foram três livros na Addison-Wesley XP Series:

- Extreme Programming Explained [Beck00]
- Extreme Programming Installed [Jeffries01]
- Planning Extreme Programming [Beck01]

Esses livros foram escolhidos porque eles são os repositórios mais óbvios de informações sobre o XP e muitos leitores interessados ou usuários potenciais do XP começarão com eles. Outros livros foram planejados, mas estavam indisponíveis no momento em que este documento foi escrito. A fonte de informações sobre o RUP é o próprio produto RUP da Rational Software Corporation.

Este documento não pretende ser um tutorial sobre o XP ou o RUP, portanto, ele será mais facilmente lido por aqueles que já têm algum conhecimento dessas abordagens ou têm acesso a algum material de referência; por exemplo, [Beck00] no caso do XP e [Kruchten00] no caso do RUP.

Alocação de Tempo e Esforços

O RUP lida com projetos (para desenvolver software), do qual o tempo de vida é dividido em fases denominadas *Iniciação, Elaboração, Construção* e *Transição*. Cada fase é adicionalmente dividida em iterações e cada iteração pode requerer várias construções.

Para a maioria dos projetos no RUP, a duração de uma iteração estará entre duas semanas e seis meses[1]e o número de iterações na existência de um projeto será entre três e nove. Portanto, nessas definições padrão, o RUP abrange projetos que variam de duração de seis semanas a 54 meses.

As iterações do XP têm cerca de duas semanas de duração. Os releases do XP são de dois meses (ou um pouco mais); eles são definidos pelo cliente e liberados para o cliente. Na duração, os releases do XP são como iterações do RUP — durante a Elaboração ou Construção[2] pelo menos, para projetos que são apropriados para o XP; ou seja, para aqueles com um tamanho máximo de equipe de, digamos, 10[3], que estão baseados em uma linha de base arquitetural estabelecida.[4]

Para mostrar isso, vamos assumir um tamanho máximo de equipe XP de 10, descobrir o projeto com o tamanho máximo que é razoável para tal equipe e, então, ver como o RUP manipulará os projetos desse tamanho e menores. Se utilizarmos COCOMO II[5] como nosso modelo de estimativa, ele prevê que uma equipe de 10 está normalmente associada a um projeto que tem uma duração máxima de 15 meses. Um projeto que dura mais de 15 meses normalmente empregará uma equipe maior que 10. Você poderia construir sistemas grandes com apenas 10 pessoas se estivesse preparado para dar a eles um planejamento mais longo, contudo, um projeto é importante o suficiente que você planejará a inclusão de mais equipe se puder fazê-lo efetivamente. Este projeto de exemplo fornecerá cerca de 40.000 a 50.000 linhas de origem de código (slocs) (800-1000 pontos de função[6] em Java) a partir do zero.

De acordo com o modelo, este é o projeto maior que você deve tentar com uma equipe com tamanho para XP se desejar que ele seja feito rapidamente. Além disso, talvez haja outras razões para uma equipe pequena não poder construir efetivamente um sistema grande; por exemplo, devido à perda de familiaridade com o código já construído. (O que uma equipe grande pode fazer em paralelo, uma equipe pequena terá de fazer em série. Portanto, quando chega a hora do teste de integração e depuração, a equipe pequena terá de revisitar o código que escreveu consideravelmente antes.)

Se olharmos um conjunto de projetos sob este tamanho, veremos um mapeamento razoável entre releases do XP e iterações do RUP. Os seguintes exemplos ilustram como eles poderão ser planejados no RUP. Os números são apenas indicativos, mas são razoáveis para projetos desses tamanhos. Eles incluem esforço e tempo para a preparação de todos os artefatos requeridos do RUP que são distribuíveis, além do código, como guias do usuário, informações de instalação e operação e assim por diante.

---

[1] A duração esperada de uma iteração em um desenvolvimento de e-business pode ser mais curta— mais provavelmente na faixa de duas a seis semanas.

[2] As iterações durante a Iniciação são normalmente mais curtas. Observe também que o modelo de fase para o RUP acomoda um grande intervalo de shapes, para que em uma fase de longa transição você possa encarar uma seqüência de iterações, cada uma culminando em uma entrega de software para o cliente, também em intervalos de dois meses, mas a entrada na transição significa que a arquitetura está completamente estável e as alterações contempladas são amplamente adaptativas, perfectivas e corretivas.

[3] Consulte [Beck00], que informa que você provavelmente não poderá executar um projeto XP com 20 programadores, mas que 10 é “seguramente viável”.

[4] O XP está intensamente focado na entrega de valor de negócios direto ao cliente, principalmente como uma função. O XP fornece pouca consideração direta à arquitetura, permitindo que isso surja sobre uma série de recriações do software, conforme funções são incluídas. Se a arquitetura da solução ainda não estiver bem estabelecida, há um risco de que isso leve a uma série de interrupções, conforme as funções são incluídas, que invalidam as premissas anteriores (e soluções especiais) e criam problemas que não são tratáveis na recriação local.

[5] O COCOMO II é um retrabalho do modelo clássico de estimativa de software COCOMO originalmente desenvolvido pelo Dr. Barry Boehm. Ele é calibrado para projetos pequenos de 2.000 slocs. Consulte [Boehm00].

[6] Os pontos de função são uma medida independente de código fonte do tamanho do software, obtida pela quantificação da funcionalidade fornecida ao usuário, baseada apenas no design lógico e nas especificações funcionais. Essa definição foi obtida do Web site International Function Point Users Group em http://www.ifpug.org/.

Field Code Changed

Exemplos de Projetos Tratáveis pelo XP

O Exemplo 1 ilustra um projeto muito pequeno consistindo em 5.000 linhas de novo código Java, requerendo cerca de 12 pessoas-meses no tempo decorrido de 7 meses.[7]

Exemplo 1

	Iniciação	Elaboração	Construção	Transição
Equipe	1	1,5	2	2
Duração em semanas	3	6	18	3
Número de iterações (e duração de cada uma em semanas)	1 (3)	1 (6)	3 (6)	1 (3)

O Exemplo 2 examina um projeto pequeno de 10.000 linhas de novo código Java, requerendo cerca de 27 pessoas-meses no tempo decorrido de 8 meses.

Exemplo 2

	Iniciação	Elaboração	Construção	Transição
Equipe	1,5	2,5	4	3
Duração em semanas	4	7	20	4
Número de iterações (e duração de cada uma em semanas)	1 (4)	1 (7)	3 (7)	1 (4)

O Exemplo 3 ilustra um projeto médio de 40.000 linhas de novo código Java, requerendo cerca de 115 pessoas-meses no tempo decorrido de 15 meses.

Exemplo 3

	Iniciação	Elaboração	Construção	Transição
Equipe	3	5	10	8
Duração em semanas	6	16	36	6
Número de iterações (e duração de cada uma em semanas)	1 (6)	2 (8)	4 (9)	1 (6)

Dentro desses limites (que abrangem inteiramente uma amplitude de desenvolvimento), as iterações do RUP são mapeadas muito proximamente com os releases do XP, na intenção e na duração.

Desenvolvimento de um Sistema Grande Não Adequado para o XP

Em desenvolvimentos muito grandes—que o RUP manipulará, mas o XP não,—as iterações do RUP são muito mais longas. O Exemplo 4 mostra um projeto de 1.500.000 linhas de novo código Java, requerendo 4.600 equipe-meses no tempo decorrido de 45 meses.[8]

---

[7] Isso pode parecer muito longo, mas observe que abrange o ciclo de vida inteiro, de um início estabelecido, com essencialmente nenhuma equipe e nenhum requisito definido (apenas a origem de uma idéia), para aceitação e encerramento do projeto. É também uma saída do modelo COCOMO II, que permite mais compactação do planejamento, mas com a penalidade de maior custo e risco. No entanto, de acordo com o planejamento na tabela, algo com funcionalidade útil poderia estar disponível logo no início de três meses e meio após o início do projeto (no final da primeira iteração de construção).

[8] Você poderá argumentar que nunca deve caracterizar um projeto dessa forma: que um projeto desse tamanho deve ser dividido em projetos menores (cada um poderia ser tratado pelo XP). É claro que projetos desse tamanho são construídos como sistemas de subsistemas (ou para projetos extremamente grandes como sistemas de sistemas), mas quando esses subsistemas ou sistemas precisam ser integrados em um único produto, então, você precisará se preocupar sobre a arquitetura e, em particular, sobre interfaces entre agregados do software e as equipes que o produzem. O XP em sua forma atual não lida com esses problemas.

Exemplo 4

	Iniciação	Elaboração	Construção	Transição
Equipe	35	70	140	100
Duração em semanas	20	50	100	20
Número de iterações (e duração de cada uma em semanas)	2 (10)	2 (25)	3 (33)	2 (10)

Claramente, neste exemplo a equipe não é organizada como uma equipe individual, monolítica—o projeto é composto de várias equipes, cada uma trabalhando em subsistemas, que poderão ser compostos de subsistemas de forma que, em níveis mais baixos, haverá planejamento em uma escala similar àquela para um projeto com tamanho tratável pelo XP. No entanto, esse projeto tem como objetivo representar um sistema integrado de forma que, no nível superior, o planejamento seja requerido para iterações em uma escala (33 semanas) que está bem além dos preceitos do XP. Essas iterações têm essa duração devido à inércia de planejamento de um projeto integrado desse tamanho. O planejamento de iterações dentro do RUP é feito através de planos de integração da construção (que podem existir nos níveis do sistema e do subsistema) e eles serão construídos em uma escala mais próxima aos releases do XP (para esse sistema muito grande) e no nível mais inferior em uma escala próxima às iterações do XP (que devem estar disponíveis em duas semanas) particularmente na elaboração e construção tardias.

Portanto, retornando aos exemplos de sistemas menores, as iterações do RUP são mais ou menos equivalentes aos releases do XP e as construções do RUP são mais ou menos equivalentes às iterações do XP.

O Que as Fases do RUP Mapeiam no XP?

Bem, pelo menos à primeira vista, as fases do RUP são mapeadas para aspectos do ciclo do XP que não são bem delineados. O XP parece estar construído, ou pelo menos descrito, para ajustar-se a uma esfera fixa de releases (harmonizando com um ciclo de negócios) que contém iterações. Há o reconhecimento de que há um *projeto* de determinado tipo que deve ser iniciado (consulte "Fazendo o Escopo de um Projeto" em [Beck01]) e que o *grande plano* construído no momento será então dividido em releases, que, por sua vez, serão divididos em iterações. [Beck01] informa que uma coisa que o grande plano fez foi nos ajudar a decidir que não era bobagem investir no projeto.

Portanto, há um momento no XP antes do ciclo de releases e iterações iniciar, no qual o trabalho está ocorrendo em um projeto para decidir se ele é viável e quanto custará e, para fazer isso, alguma idéia grosseira da funcionalidade requerida precisa ser construída. Isso é o que o RUP chama de fase de iniciação. Você tem a impressão no XP que ela deve ser feita rapidamente, muito rapidamente. No RUP, dizemos que ela leva o tempo que for necessário—dependendo dos riscos inerentes. Mesmo no XP, você lê sobre as seguintes coisas que precisam ser feitas neste momento (por um pequeno número de pessoas):

- levantamento de alguns requisitos (escrever algumas "estórias importantes");
- algum planejamento;
- alguma negociação com o cliente (definindo expectativas);
- recriação em qualquer restrição de negócios.

É fácil ver que levará alguns dias para ir de um início despreparado até ter uma Visão (isso é como as "estórias importantes" são chamadas no RUP) e um Caso de Negócios (orçamento e ROI (Retorno do Investimento) que justifique o projeto) e obter um primeiro corte no Plano de Desenvolvimento de Software.

No RUP, como no XP, você faz explorações e planejamentos suficientes no início do projeto para dar continuidade após estabelecer que há uma boa probabilidade de sucesso. Conforme mostrado anteriormente no Exemplo 2, o planejamento do RUP sugere um investimento (em esforço) de cerca de $12.000 dólares em iniciação para justificar o gasto de cerca de $250.000 dólares. Isso irá variar; em algumas áreas de problema bem dominadas, você não precisará fazer muito; por exemplo, talvez você não esteja iniciando do zero. Em outras ocasiões, você precisará gastar mais iniciando do zero em um domínio não-familiar que requer algum grau de pesquisa e desenvolvimento.

Depois de sua fase equivalente à iniciação, o XP vai direto para o planejamento do primeiro release. O RUP informa que a fase de elaboração segue a iniciação e nas iterações de elaboração, a arquitetura da solução é estabilizada. Em comparação

aparente, o XP sugere que o planejamento de release seja orientado por função para que todos os releases entreguem o valor de negócio para o cliente. O XP é bastante explícito em seu desdém para o que ele chama de planejamento de infra-estrutura: apenas o suficiente deve ser feito para suportar a funcionalidade selecionada para esse release. A infra-estrutura é incluída mais tardiamente que o requerido e, se o sistema for interrompido devido a decisões de infra-estrutura antecipadas serem invalidadas por inclusões tardias de função, então, o código será recriado para fazê-lo funcionar.[9] A idéia é não gastar muito tempo construindo algo que não entrega valor para o cliente.

Em comparação, alguns pontos precisam ser destacados sobre o RUP:

- ***O conceito de arquitetura do RUP não é simplesmente infra-estrutura***

  Para citar a partir do RUP, "…a arquitetura de um sistema de software (em um determinado ponto) é a organização ou a estrutura dos componentes significativos do sistema que interagem por meio de interfaces, com componentes constituídos de componentes e interfaces sucessivamente menores."

  Portanto, a arquitetura enfoca a solução completa—não apenas a infra-estrutura—de um conjunto específico de perspectivas, em um nível adequado de abstração.

- ***A noção do RUP de arquitetura executável entrega o valor de negócios para o cliente***

  Ela permite a demonstração precoce de uma solução que satisfaz os requisitos não funcionais do cliente (bem como, potencialmente, alguns funcionais no processo).

  A redução de risco dessa forma—e, freqüentemente, os requisitos não funcionais são os que apresentam risco—representa em si um valor de negócios considerável para o cliente.

Por que acreditar que é necessário descrever o processo dessa maneira? Como o RUP está por toda a parte conduzindo riscos e acreditamos que haja classes de problemas e sistemas que são simplesmente não tratáveis para a abordagem do XP em permitir que a arquitetura surja da recriação de código—normalmente maiores e mais complexos. Um risco é que em fazer alterações locais na recriação (e o escopo ficará, por necessidade, limitado), serão produzidas otimizações locais que, no agregado, levarão a uma solução muito menos favorável. Isso não é apenas nossa asserção: Kent Beck diz muito sobre si mesmo em [Beck00], no capítulo intitulado *Four Values*, no título *Courage*. A dificuldade é que o XP informa pouco sobre a origem da inspiração para alterações arquiteturalmente significativas—apenas que é preciso ter coragem[10] para aplicá-las!

Então, existem alterações que são difíceis de recriar. Por exemplo, para tornar um sistema com um único usuário e uma única máquina um sistema multiusuário e multiprocessador, você precisará reprojetar várias coisas e ainda acabar com um sistema que tem muitas restrições.

A ênfase do RUP na arquitetura tem como objetivo enfocar aqueles casos nos quais a abordagem inicial pode estar errada e precisa ser completamente reconsiderada. Para sistemas menores, isso é menos que um risco—a recriação pode fazer uma limpeza mais ampla no sistema e há muito menos probabilidade de que a arquitetura inicial estivesse incorreta.

Observe também que onde há menos risco percebido (devido ao tamanho, nova tecnologia, falta de familiaridade, complexidade, outras exigências de desempenho, confiabilidade, segurança e assim por diante) e a solução pode ser entregue em uma estrutura existente bem entendida, então, a fase de elaboração do RUP (a fase na qual problemas arquiteturais são manipulados) poderá ser bem breve (e preocupar-se mais com o levantamento, refinamento e demonstração de requisitos funcionais importantes). Ou seja, o ciclo de vida do RUP é reduzido visando algo que se aproxime a um ciclo de vida do XP. Quando um problema puder ser enfocado com êxito pelo XP, o Caso de Desenvolvimento do RUP também produzirá algo similarmente 'leve' (embora não exatamente o mesmo).

A fase de construção no RUP é equivalente à série de releases do XP com a qualificação que, se você estivesse seguindo o RUP, teria de entregar os resultados de cada iteração de construção ao cliente para ter o mesmo efeito do XP. O XP não tem realmente um equivalente da fase de transição do RUP, porque cada release do XP já está nas mãos do cliente. Há, no final de

[9] Observe que a recriação apenas resulta em aprimoramentos locais: a recriação não enfoca a estrutura global. Se a estrutura global estiver resolvendo o problema errado, então, a recriação não produzirá uma solução correta. A única solução é fazer alterações radicais ou, em casos extremos, reiniciar. Isso tem riscos óbvios de orçamento e planejamento.

[10] Ninguém discutiria que coragem é uma virtude—e por conclusão alguém possuído por "boa pessoa". Contudo, mesmo o mais corajoso as vezes irá precisar de um jeito organizado de trabalhar e de uma estrutura na qual os problemas difíceis podem ser resolvidos.

um projeto do XP, o que [Beck00] chama de morte do projeto (consulte as informações no título *Ciclo de Vida do XP* mais adiante), na qual ocorreriam algumas atividades de encerramento do projeto.

O Que as Fases do XP Mapeiam no RUP?

As fases do XP (que se aplicam aos releases e iterações do XP) são exploração, confirmação e direcionamento. No nível do release, elas se alinham com coletas específicas de tarefas (atividades no RUP) na disciplina Gerenciamento de Projetos do RUP. São elas Plano para a Próxima Iteração (mapeamento para exploração e confirmação) e Monitorar e Controlar Projeto (mapeamento para direcionamento). Isso é ilustrado nas seguintes seções.

Fases do XP

Em [Beck00], há um capítulo sobre o ciclo de vida de um projeto XP ideal e os títulos de primeiro nível são **exploration**, **planning**, **iterations to first release**, **'productionizing'**, **maintenance** e **death**. Essas parecem ser as fases de nível superior, mas o resultado disso não é bem preciso. Um projeto do XP será limitado por uma fase de exploração *inicial* e "morte" quando o projeto estiver encerrado. Mas a esfera principal é o release e *cada release tem uma fase de exploração.* Portanto, a fase de exploração que suporta o projeto (e leva para o primeiro release) é um caso especial pois há uma porta inicial a ser passada (consulte *Fazendo o Escopo de um Projeto* em [Beck01]) na qual uma decisão é tomada sobre a sensatez de continuar. Os releases e iterações do XP têm as três fases—exploração, confirmação e direcionamento. A "manutenção" realmente caracteriza a natureza do projeto XP depois do primeiro release.

Ciclo de Vida do XP

Globalmente, o ciclo de vida do XP para um projeto com sete releases acima de 15 meses poderá ficar semelhante à Figura 1.

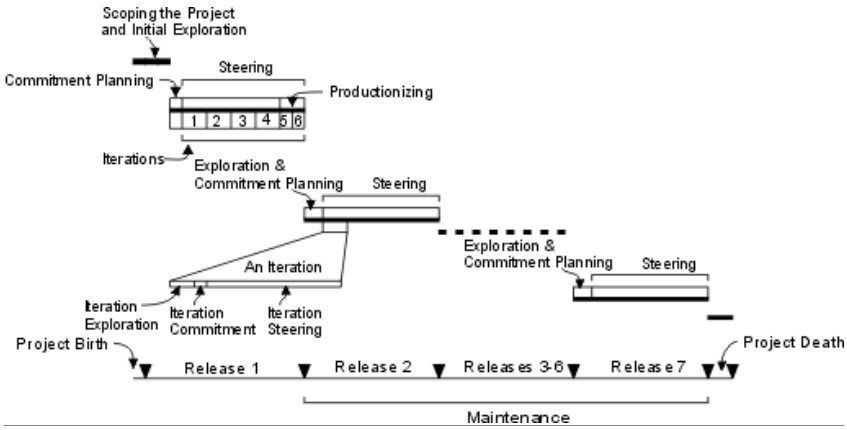

Figura 1

Cada release depois do primeiro (que tem três módulos) tem cerca de dois meses de duração e cada iteração cerca de duas semanas, exceto nos estágios posteriores de um release ("Productionizing", conforme denominado no [Beck00]) quando o ritmo da iteração é acelerado.

Faz sentido de um ponto de vista da métrica de planejamento? Neste exemplo inventado, tentamos seguir as diretrizes prescritas pelo XP—que os releases devem ter cerca de dois meses de duração, com o primeiro release estando entre dois e seis meses—e mantivemos o tamanho da equipe abaixo de 10. Se este projeto for como o Exemplo 3 ilustrado anteriormente, ele fornecerá cerca de 40.000 linhas de código Java ou 800 pontos de função (se aceitarmos o fator de conversão encontrado no modelo COCOMO II) no total. Se isso for uniformemente dividido entre releases, então, cada release fornecerá cerca de 115 pontos de função.

A parte duvidosa será o primeiro release: iniciar do zero (digamos, um membro da equipe designado, nenhum requisito capturado, nenhum escopo feito ainda) para uma entrega a um cliente de algo de qualidade de produção em três meses, será

um desafio. Mesmo que permitamos alta produtividade, digamos, 12 pontos/equipe-mês (com base nos dados do segmento de mercado da David Consulting Group, consulte http://davidconsultinggroup.com/indata.htm), não será possível aumentar a equipe em uma taxa arbitrariamente alta—o problema é que o espaço com o qual estamos lidando aqui (115 pontos de função) é muito pequeno e não pode ser dividido em partes arbitrariamente pequenas, senão uma equipe grande poderia atacá-la.

Field Code Changed

No entanto, se assumirmos que isso pode ser feito, então, uma equipe de tamanho médio, com aproximadamente quatro, será necessária e o projeto gerará o release um com uma equipe de sete. Se mais de um membro da equipe for incluído na equipe durante o projeto, então, para o restante do projeto estaremos no modo normal do XP (manutenção) com uma equipe de oito—que está na zona de conforto de tamanho. Conforme o tamanho entregue aumenta, a produtividade certamente cairá um pouco e o intervalo de release diminuído (dois meses) deslocará os membros da equipe aumentada, para que cada release entregue cerca do mesmo peso funcional adicional do primeiro. O esforço global do projeto tem então 108 equipe-meses—um pouco menor que no Exemplo 3 mostrado anteriormente.

Ciclo de Vida Padrão do RUP

Em comparação, o ciclo de vida *padrão* para um projeto do RUP de tamanho similar se assemelha ao seguinte:

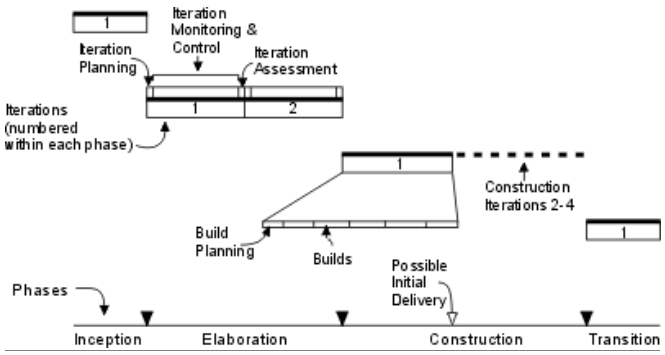

Figura 2

Novamente, este plano baseia-se no Exemplo 3. As iterações do RUP são numeradas dentro de cada fase. Utilizando o ciclo de vida padrão do RUP, mais provavelmente a oportunidade mais próxima para *entregar* qualquer coisa que se aproxime da qualidade de produção ao cliente seria o final da primeira iteração de construção, que ocorre sete meses dentro do projeto. No entanto, nesse momento, o RUP poderia entregar cerca de um quarto, ou 200 pontos de função, da capacidade total (como nos 115 pontos de função do XP após três meses). Normalmente, a primeira entrega ao cliente ocorreria no final da construção, que nesse caso seria 13 meses para o projeto, momento em que essencialmente toda a capacidade estaria presente. Isso não significa que a primeira vez que o cliente verá o produto; com o RUP, o cliente é convidado para as avaliações de iteração e pode ver o produto em desenvolvimento sob teste.

Por que existe essa diferença entre o RUP e o XP? Ela ocorre porque, nesse caso padrão, a suposição é de que há risco suficiente na arquitetura da solução (porque, por exemplo, ela não tem precedente, tecnologia é nova ou os requisitos não funcionais são particularmente onerosos) para garantir duas iterações na elaboração para estabilizá-la, antes de tentar construir a funcionalidade requerida pelo cliente. Isso significa que no final da elaboração, apenas os requisitos arquiteturalmente significativos serão explorados (e possivelmente implementados).

O XP não faz isso. Ele continua a entregar uma série de soluções verticalmente completas em cada release, assumindo-se que a arquitetura não será interrompida entre os releases ou que a recriação poderá reparar qualquer interrupção que ocorra. Acreditamos que isso limita a aplicação sensível do XP a classes do sistema que podem ser construídas em arquiteturas existentes de capacidade conhecida.

Ciclo de Vida Real do RUP

Para tais sistemas, o ciclo de vida real do RUP será um pouco diferente; a fase de elaboração será bem mais reduzida e provavelmente a fase de iniciação também. Se negociarmos uma iteração de elaboração para uma iteração de construção, poderemos adiantar o final da primeira iteração de construção em alguns meses, cinco em vez de sete.

Isso provavelmente entregará um pouco menos de 200 pontos de função no exemplo padrão (porque o número de equipes trabalhando na iteração de construção mais adiantada será menor), mas isso está se aproximando da agilidade de planejamento do XP—com afrouxamento suficiente para reduzir significativamente o risco. Ao extremo, podemos imaginar uma abordagem baseada no RUP estabelecida em torno da entrega inicial de um sistema pequeno (digamos, 115 pontos de função como no exemplo do XP), seguida por uma série de ciclos evolutivos completos do RUP (iniciação completa, elaboração, construção e seqüências de transição) para entregar o restante do sistema em um modo de manutenção análogo ao XP.

Novamente, por volta de seis meses, a entrega inicial provavelmente estará no limite externo do intervalo de tempo recomendado para o XP, mas, então, o RUP reconhece e requer que você planeje (e, portanto, aloque tempo e esforço) coisas que o XP parece atenuar, como a implantação de releases.

Podemos concluir outras restrições, a partir dessas observações, sobre a natureza de projetos para os quais o XP é adequado. O XP requer explicitamente um relacionamento muito íntimo entre o cliente e a equipe de desenvolvimento, requerendo um cliente em tempo integral on-site e que os clientes escrevam histórias e definam os releases. No XP, o cliente é realmente uma função na equipe: a pessoa ou grupo que desempenha a função poderá ou não pertencer a uma empresa de aquisição separada, mas eles têm a autoridade de falar com quem estiver adquirindo o software. Essencialmente, eles têm os requisitos.

Esse tipo de relacionamento é mais freqüentemente encontrado em desenvolvimentos domiciliares do que em desenvolvimentos contratados formalmente para clientes externos. Um ciclo de release de dois meses também seria mais aceitável em tais ambientes. A implantação de um novo release (com novos recursos, não apenas correções de defeitos) a cada dois meses, através dos limites da empresa, provavelmente terá uma despesa logística inaceitável.

Com esses tipos de afrouxamento (equipe pequena, arquitetura já estabelecida, desenvolvimento de estilo domiciliar com pouca implantação de praxe) um ciclo (ou ciclos) ajustado do RUP terá um efeito similar a um desenvolvimento do XP. A duração inicial do release poderá ser um pouco mais longa do que o ideal do XP, mas terá menos risco.

Na direção oposta (equipe grande, arquitetura sem precedente, desenvolvimento contratado formalmente com restrições de entrega e implantação) é difícil ver como o XP pode escalar—e, para ser imparcial, as fontes deste documento não afirmam que ele possa. O RUP, por outro lado, é projetado para enfrentar a formalidade e a cerimônia que acompanham tais projetos. Talvez uma abordagem semelhante à do XP (ou seja, utilizar técnicas do XP como programação em pares e recriação) possa ser incorporada *dentro* de um projeto maior do RUP, mas apenas quando a arquitetura estiver estável.

Artefatos

O RUP descreve mais de 100 artefatos, o que tem levado a críticas de que ele impõe uma despesa burocrática intolerável para projetos pequenos. Essa visão está incorreta em quatro cálculos:

- Os projetos não *têm* de produzir todos os artefatos: a seleção e o ajuste de artefatos é um parte necessária do processo. O RUP fornece diretrizes sobre o que pode ser otimizado e o que pode ser ajustado.
- Um artefato (no RUP, uma descrição *de artefato*) é uma entidade de especificação—ele especifica as informações a serem produzidas por uma tarefa e, para fazer isso sistematicamente, poderá descrever a forma *abstrata* de um artefato. A caracterização de artefatos do RUP como modelos, elementos de modelo ou documentos (cerca de metade dos artefatos do RUP são classificados como documentos) não tem o objetivo de implicar uma realização *particular*.

  Um modelo UML poderá ser capturado e apresentado utilizando uma ferramenta com finalidade de construção (como o Rational Rose) ou, no outro extremo, como diagramas feitos utilizando uma ferramenta gráfica simples ou até mesmo como um esboço em um whiteboard—isso ainda contaria como um artefato em termos de RUP (embora haja um risco de perda com um whiteboard!).

  O termo "documento" tem conotações históricas que ainda influenciam as pessoas a pensarem em um documento como um produto de trabalho distinto, baseado em papel, com um conjunto específico (e requerido) de títulos de parágrafo—dos quais todos têm de ser preenchidos com texto (e talvez diagramas) para o documento ser concluído. Mas isso é apenas uma realização de um documento do RUP.

  Os documentos do RUP são conjuntos de informações compostos de texto e diagramas. Para ser útil, o RUP especifica o que devem ser essas informações e uma maneira conveniente e sistemática de fazer isso é apontar para um gabarito, que enumera e descreve os itens e problemas a serem considerados. É certamente possível utilizar esses gabaritos de uma maneira direta e formal para perceber os artefatos na forma de documento (no senso geral); ou seja, eletrônico ou papel. Sabendo que muitos projetos desejarão fazer isso, escolhemos a apresentação de um conjunto de gabaritos que poderão ser utilizados, mais ou menos diretamente, dessa maneira. Mas é essencial fazer isso—nem tudo apresentado em um gabarito é necessariamente relevante para um projeto específico—e o RUP fornece a um projeto a liberdade de escolher qualquer percepção adequada (e mecanismo de entrega) para um artefato. O importante são as informações que ele contém.
- Sentimos que era essencial que todos os produtos de trabalho potenciais do processo fossem claramente identificados para consideração pelo engenheiro de processo e coordenador de projeto, portanto, as decisões para omiti-los ou ajustá-los poderiam ser feitas conscientemente e com o entendimento total das conseqüências da omissão. Acreditamos que tornar a lista de artefatos completa e explícita dessa maneira fornece objetividade à configuração do processo. Também dá suporte ao coordenador de projeto menos experiente chamando sua atenção a coisas que um coordenador mais experiente "veria ao ler", onde não foram mencionadas, mas que o iniciante poderia apenas não notar. Ter um conjunto de documentos bem definido também ajuda o cliente e o coordenador de projeto a concordarem antecipadamente com o que deve ser entregue e em que forma para evitar os argumentos desagradáveis que freqüentemente caracterizam o encerramento do projeto.
- O RUP descreve vários artefatos que são compostos e também continua a descrever seus componentes como artefatos. Isso permite um descrição granulada das entradas e saídas de tarefas, onde for útil. Em sua descrição, o RUP poderia ter parado na composição, por exemplo, o Modelo de Design e referido-se a Classes simplesmente como partes do Modelo de Design e não como artefatos por sua própria conta. A identificação desses artefatos permite uma descrição precisa de seu uso, ao mesmo tempo que está em conformidade com o metamodelo do processo à custa de aumentar a contagem global de artefatos.

O XP parece estar imune a críticas de ser cheio de artefatos, mas isso é uma simplificação excessiva por duas razões:

- O XP já *está ajustado* para adequar um determinado tipo de desenvolvimento, lidando com um subconjunto das disciplinas do RUP em uma determinada escala e, portanto, o esperado é que busque menos artefatos.

- O XP enfatiza a importância de histórias do usuário e códigos, mas outras coisas que são produtos de trabalho são mencionadas na transmissão ao descrever o processo, portanto, a contagem de artefatos não é tão pequena quanto parece a princípio.

Considerando a abordagem do XP, talvez não seja surpresa que nos três livros que utilizamos como material fonte para essa comparação (porque eles são o carro-chefe do processo), os termos “artefato” e “produto de trabalho” não aparecem no índice. No entanto, não é difícil ler o texto e selecionar referências que são artefatos. A seguir são apresentados alguns exemplos:

- Histórias
- Restrições
- Tarefas
- Tarefas técnicas
- Testes de aceitação
- Código de—software
- Releases
- Metáforas
- O design—CRC, esboço UML
- Documentos de design—produzidos no final do projeto
- Padrões de codificação
- Testes de unidade
- Espaço de trabalho (desenvolvimento e outros recursos)
- Plano de liberação
- Plano de iteração
- Relatórios e notas sobre reuniões
- Plano global—orçamento
- Relatórios em andamento
- Estimativas de estórias
- Estimativas de tarefas
- Defeitos (e dados associados)
- Documentação adicional a partir de conversações
- Documentação de suporte
- Dados de Teste
- Ferramentas de estrutura de teste
- Ferramentas para gerenciamento de código
- Resultados de teste
- Pontas (soluções)
- Registro de tempo de trabalho em tarefas
- Dados métricos
    - Recursos
    - Escopo
    - Qualidade
    - Tempo
- Outras métricas

- Resultados de rastreio

Há cerca de 30 artefatos, alguns dos quais também são compostos. A lista deve ser indicativa, não exaustiva. Os livros do XP não perdem muito tempo descrevendo a maioria deles e, no nível em que os livros foram escritos, não poderemos esperar isso deles. No entanto, depois que um projeto precisar entendê-los, mais detalhes sobre seu conteúdo e forma serão necessários. Um projeto pode certamente fazer isso sem se deter, mas isso utiliza tempo do trabalho real; em comparação, o RUP fornece orientação direta, portanto, economizando o tempo de projeto.

Por Que o XP Precisa dos Artefatos do RUP?

Uma razão é que o XP não tem o escopo do RUP. Isso é completamente intencional; o XP está disponível para programação para atender uma necessidade de negócios. Como a necessidade de negócios ocorreu—e como é modelada, capturada e deduzida—não é uma preocupação principal do XP. O membro da equipe do XP que é o cliente apresenta a destilação dos requisitos como histórias para a equipe de desenvolvimento do XP; eles também são os árbitros do valor de negócios. A mágica de como as histórias vieram a ser expressas (ou exprimíveis) nessa forma não é a preocupação do XP. Portanto, por exemplo, o que o RUP descreve na disciplina Modelagem de Negócios está fora do escopo do XP (a Modelagem de Negócios no RUP tem ~14 artefatos). O XP descreve um processo que tem releases regulares para o cliente. As logísticas de implantação desses releases não são a preocupação de desenvolvimento, portanto, a disciplina Implantação do RUP está amplamente fora do escopo do XP (a Implantação no RUP tem ~9 artefatos). Portanto, para um projeto pequeno tratável pelo XP, você esperará omitir ~23 artefatos ao ajustar o RUP.

Outra razão é que o XP afirma que a captura de requisitos e design pode ser feita simplesmente: os requisitos são capturados como estórias do usuário (que podem às vezes ter sido divididas) que são, então, decompostas em tarefas—que, essencialmente, são o design—tendo em mente uma metáfora para o sistema. Isso é possível para todos os sistemas? Definitivamente, não. Isso é possível para alguns sistemas? Certamente, e para ser imparcial com o XP, não é solicitado o enfoque em todos os sistemas. Além disso, talvez os autores do XP estivessem com suas línguas firmemente cravadas na bochecha ao fazerem essas solicitações.

O comportamento desejado de sistemas maiores e mais complexos pode ser muito difícil de articular sem alguma abordagem sistemática, como casos de uso. Nem será possível confiar na conversação entre o cliente e o desenvolvedor para elaborar consistentemente estórias complexas do usuário, sendo a memória humana falível como é. Além disso, o desenvolvimento das estruturas que *entendem* comportamento complexo em sistemas grandes é ajudado pela habilidade de formar e deduzir várias visões abstratas da arquitetura do sistema. Os artefatos que o RUP descreve nas disciplinas Requisitos (~14 artefatos) e Análise & Design (~17 artefatos), permitem que ele encare a variedade, o tamanho e a complexidade desses sistemas.

No roteiro de projetos pequenos do RUP, a contagem de artefatos (para Requisitos e Análise & Design) é reduzida para sete, com alguma redução obtida simplesmente com a referência ao artefato composto. Isso não é uma escamoteação; ele lida com artefatos da mesma maneira que o XP. Por exemplo, devido à forma que ele provavelmente é entendido em um projeto pequeno, o RUP informa o seguinte sobre o Modelo de Design:

> "O Modelo de Design deve evoluir através de uma série de sessões de sugestões de idéias nas quais os desenvolvedores utilizarão cartões CRC e diagramas desenhados à mão para explorar e capturar o design. O Modelo de Design será mantido apenas enquanto os desenvolvedores o acharem útil. Ele não será mantido consistente com a implementação, mas será preenchido para referência."

Finalmente, o RUP permite uma formalidade maior (e "cerimônia") no gerenciamento de projeto, onde isso for necessário, e em algumas disposições de contrato haverá necessidade disso**.**Novamente, muitos artefatos de gerenciamento de projeto do RUP (existem 15 na disciplina Gerenciamento de Projeto) fazem parte dos artefatos compostos e precisam apenas ser percebidos como documentos separados quando a formalidade do projeto solicitar isso. Por exemplo, no RUP o Plano de Desenvolvimento de Software "contém" artefatos como o Plano de Gerenciamento de Risco e o Plano de Aceitação do Produto. Em projetos menores, menos formais, poderá ser possível lidar com os problemas enfocados por esses planos simplesmente com um parágrafo ou dois no Plano de Desenvolvimento de Software. No roteiro de projetos pequenos no RUP, o número de artefatos de gerenciamento de projeto será reduzido para seis.

Comparando Artefatos para um Projeto Pequeno

Portanto, quando você ajustar o RUP para um projeto pequeno e ajustar os requisitos do artefato, o que acontecerá globalmente? Quando você vê o roteiro de um projeto pequeno no RUP e conta os artefatos, o número é 30 (26 se você retirar

alguns de implantação)—afinal, não está tão cheio de artefatos! O RUP simplesmente delineia claramente o que o XP deixa obscuro, permite que você decida o que é necessário e fornece orientação sobre como fazer a seleção. Agora, a granularidade varia nos dois lados, mas o ponto é que a contagem de artefatos no RUP para projetos pequenos do tipo que o XP enfocaria confortavelmente é da mesma ordem do XP.

Mapeamento Grosseiro de Artefatos do XP para Artefatos do RUP

XP	Roteiro de Projetos Pequenos do RUP
Histórias Documentação adicional a partir de conversações	Visão Glossário Modelo de Caso de Uso
Restrições	Especificações Complementares
Testes de aceitação Testes de unidade Dados de Teste Resultados de teste	Modelo de Teste
Código de—software	Modelo de Implementação
Releases	Produto Notas de Release
Metáfora	Documento de Arquitetura de Software
Design—CRC, esboço UML Tarefas Tarefas Técnicas Documentos de design—produzidos no final do projeto Documentação de suporte	Modelo de Design
Padrões de codificação	Diretrizes de Design Diretrizes de Programação
Espaço de trabalho (desenvolvimento e outros recursos) Ferramentas de estrutura de teste	Ferramentas
Plano de liberação Estimativas de estórias Estimativas de tarefas Plano de iteração	Plano de Desenvolvimento de Software Plano de Iteração
Plano global—orçamento	Caso de Negócios Lista de Riscos Plano de Aceitação do Produto
Relatórios em andamento Registro de tempo de trabalho em tarefas Dados de métricas: Recursos, Escopo, Qualidade, Tempo Outras métricas Resultados de rastreio Relatórios e notas sobre reuniões	Avaliação de Status
Defeitos e dados associados	Controle de Alterações
Ferramentas para gerenciamento de código	Plano de Gerenciamento de Configuração Repositório de Projetos Espaço de Trabalho
Ponto	Protótipos
	Caso de Desenvolvimento Templates Específicos do Projeto

Diretrizes

Associado aos artefatos, o RUP fornece diretrizes, que são essencialmente informações adicionais sobre esse artefato, seu significado, representação, relacionamentos com outros artefatos, conteúdo e uso. Essas diretrizes irão abranger várias centenas de páginas físicas, se impressas, o que pode parecer desanimador, mas então você apenas lê o que precisa, para os artefatos que necessita.

Os livros do XP também contêm muita orientação sobre práticas e artefatos, sem tentar (ou precisar) modelar os relacionamentos rigorosamente. O corpo total da literatura sobre o XP não é pequeno. Os três livros disponíveis no momento da escrita totalizam quase 600 páginas e os dois outros estarão disponíveis logo incluindo outras 700 páginas ou mais. Então, há o livro sobre recriação [Fowler99] com mais de 400 páginas.

Além disso, o XP é discutido em vários Web sites. No entanto, a discussão tende a sobrepor—examinando o XP de pontos de vista diferentes e incluindo na base de experiência. Realmente, isso ilumina outra diferença entre o RUP e o XP. O RUP é um produto; o XP não. Não há nenhuma fonte única para o XP, embora os livros sejam um bom início. A forma mais rápida de "adquirir" o XP seria através do treinamento que está comercialmente disponível a partir do líderes de opinião por trás dele.[11]

[11] O RUP é às vezes retratado como "proprietário", mas qualquer pessoa pode comprar o RUP e utilizar as idéias contidas nele, incluí-lo, excluir partes que não são relevantes para sua organização ou projeto e assim por diante, contato que respeite os direitos autorais e o contrato de licença. A manifestação do XP na forma de livro também é propriedade intelectual pertencente aos autores e editores; a única diferença é a extensão em que cada um pode ser completamente compreendido a partir de suas fontes. O RUP, como um produto, tenta ser completo; os livros atuais do XP precisam de alguma suplementação (através de treinamento imersivo ou de consulta).

Tarefas

O RUP define formalmente o termo "tarefa" como trabalho executado por uma função, utilizando e transformando artefatos de entrada e produzindo artefatos de saída novos e alterados. O RUP então enumera essas tarefas e as categoriza de acordo com a "disciplina" ou a "área de preocupação" principal dentro do projeto (conforme o RUP a definir). Essas disciplinas são:

- Modelagem de Negócios
- Requisitos
- Análise & Design
- Implementação
- Teste
- Implantação
- Gerenciamento de Configuração & Alterações
- Gerenciamento de Projeto
- Ambiente

As tarefas estão relacionadas ao tempo por meio dos artefatos que produzem e consomem: uma tarefa pode logicamente ser iniciada quando suas entradas estiverem disponíveis (e em um estado de maturação apropriado). Isso significa que os pares de tarefas produtor-consumidor podem se sobrepor no tempo, se o estado do artefato permitir. Eles não precisam obedecer a uma seqüência rígida.

As tarefas no RUP têm como objetivo tornar o processo intelectual de produzir um artefato menos opaco—dar alguma orientação forte sobre como algo deve ser produzido. As tarefas também podem ser utilizadas para ajudar o coordenador de projeto com seu planejamento. Combinadas por meio do RUP, conforme descritas em termos de ciclo de vida, artefatos e tarefas são as "boas práticas": princípios de engenharia de software que comprovadamente produzem softwares de qualidade construídos para orçamento e planejamento previsíveis.

Através de suas tarefas e seus artefatos associados, o RUP suporta e realiza essas melhores práticas—elas são temas executados através do RUP. Observe que o XP utiliza a noção de "práticas" também, mas, como veremos, não corresponde exatamente ao conceito de boas práticas do RUP.

O XP (consulte [Beck00], Capítulo 9) apresenta uma visão de desenvolvimento de software que é atraentemente simples, composta por quatro tarefas básicas que devem ser ativadas e estruturadas de acordo com algumas práticas de suporte:

- Codificação
- Teste
- Atendimento
- Design

Na verdade, as tarefas do XP estão mais relacionadas em escopo às disciplinas do RUP do que às tarefas do RUP e muito do que acontece em um projeto do XP (além da codificação, teste, atendimento e design) é resultado da elaboração e da aplicação de suas práticas.

Existe um Equivalente das Tarefas do RUP no XP?

Sim, há, mas as "tarefas" do XP não são formalmente identificadas ou descritas; por exemplo, verificando o Capítulo 4, *Estórias de Usuários* em [Jeffries01], você encontrará o título "*Definir Requisitos com Estórias Escritas em Cartões* " e em todo o capítulo há uma mistura da descrição do processo e da orientação sobre o que são estórias de usuários e como (e por quem) elas devem ser produzidas. E assim vai, como os livros descrevem o XP sob títulos principais (que são uma mistura em [Jeffries01]—algumas estão focadas em artefatos, outras em atividades); "Coisas Concluídas" e "Coisas Produzidas" estão descritos com graus variáveis de prescrição e detalhes.

A prescrição aparente do RUP é resultado da conclusão e maior formalidade no tratamento sistemático de tarefas e de suas entradas e saídas. *Não* falta prescrição para o XP, porém, na sua tentativa de permanecer "leve", a formalidade e os detalhes são simplesmente omitidos. Os detalhes estão presentes no RUP e terão de ser incluídos quando o XP for implementado em

um projeto. A falta de especificidade não é uma vantagem ou uma desvantagem, mas você não deve confundir a falta de informações detalhadas no XP com simplicidade. Em algum ponto, as pessoas no projeto precisarão saber o que fazer e, nesse momento, precisarão dos detalhes.

Disciplinas

Uma disciplina no RUP é a coleta de tarefas (e conceitos associados) que produzem um conjunto específico de artefatos, que representa algum aspecto ou preocupação importante no desenvolvimento de software. Esse uso alinha-se bem com a definição no dicionário de disciplina como uma ramificação de conhecimento ou ensinamento.

Conforme observado anteriormente, as disciplinas do RUP são Modelagem de Negócio, Requisitos, Análise & Design, Implementação, Teste, Implantação, Configuração & Gerenciamento de Alterações, Gerenciamento de Projeto e Ambiente.Isso não abrange cada aspecto do que uma organização ou empresa pode fazer quando emprega pessoas para desenvolver, implantar, operar, suportar, vender, comercializar ou, de alguma outra forma, lidar com sistemas que são amplamente de software. Atualmente, o RUP não abrange a Engenharia de Sistemas, por exemplo. Nem abrange todos os requisitos de alguns padrões internacionais de processo de software, como ISO 15504 (que extrai aspectos relacionados à aquisição de software e ao gerenciamento de recursos humanos, por exemplo). Isso é feito por escolha: esses outros aspectos, embora importantes, estão fora do foco de engenharia do RUP.

O XP se restringe ainda mais: ele inclui quatro atividades básicas—codificação, teste, atendimento e design (observado anteriormente como mais proximamente alinhadas com as disciplinas do RUP)—executadas utilizando um conjunto de práticas, que requer a execução de outras atividade, que são mapeadas para algumas das outras disciplinas no RUP. As práticas do XP, mencionadas de forma literal em [Beck00], são:

- *"O Jogo do Planejamento*—Determina rapidamente o escopo do próximo release, combinando as prioridades do negócio e as estimativas técnicas. Quando a realidade superar o plano, atualize-o.
- *Pequenos Releases*—Coloca um sistema simples em produção rapidamente, em seguida, libera novas versões em um ciclo muito curto.
- *Metáfora*—Guia todo o desenvolvimento com uma estória simples compartilhada de como todo o sistema funciona.
- *Design Simples*—O sistema deve ser projetado o mais simplesmente possível a qualquer momento. Qualquer complexidade extra deve ser removida assim que for descoberta.
- *Teste*—Os programadores escrevem testes de unidade continuamente, que devem ser executados sem problemas para que o desenvolvimento prossiga. Os clientes escrevem testes demonstrando que as características foram alcançadas.
- *Recriação*—Os programadores reestruturam o sistema sem alterar seu comportamento para remover a duplicação, aprimorar a comunicação, simplificar ou incluir flexibilidade.
- *Programação aos pares*—Todo o código de produção é gravado com dois programadores em uma máquina.
- *Propriedade coletiva*—Qualquer pessoa pode alterar qualquer código em qualquer lugar do sistema a qualquer momento.
- *Integração contínua*—Integra e constrói o sistema muitas vezes por dia, sempre que uma tarefa é concluída.
- *40-horas por semana*—A regra é não trabalhar mais de 40 horas por semana. Nunca faça hora extra duas semanas seguidas.
- *Cliente on-site*—Inclui um usuário real, ativo na equipe, disponível tempo integral para responder a perguntas.
- *Padrões de codificação*—Os programadores escrevem todo o código de acordo com as regras, enfatizando a comunicação por meio do código."

Algo que deve ser notado no tópico *Design Simples* é que "complexidade extra" é um termo meio subjetivo e difícil de definir, sendo amplo na opinião do observador experiente.

As atividades executadas como resultado da prática *O Jogo do Planejamento*, por exemplo, serão mapeadas principalmente para a disciplina de Gerenciamento de Projeto do RUP. No entanto, alguns tópicos estão fora do escopo do XP; por exemplo, Modelagem de Negócio. O *levantamento* de requisitos está ainda mais fora do escopo de XP—o cliente (on-site com a equipe) define e fornece os requisitos (na forma de estórias). A implantação do software liberado está fora do escopo do XP.

Devido à escala e aos tipos de desenvolvimento que ele enfoca, o XP pode lidar *muito* superficialmente com problemas nas disciplinas Ambiente e Gerenciamento de Configuração & Alterações que o RUP cobre em detalhes.

Uso de Práticas do XP no RUP

Nas disciplinas em que o XP e o RUP se sobrepõem, algumas das práticas descritas no XP poderão ser implementadas no RUP (e algumas já estão); por exemplo:

- *Programação em pares:* O XP requer que o código de produção seja produzido pelos programadores que trabalham em pares em uma única estação de trabalho e afirma que isso tem efeitos benéficos na qualidade do código e, depois de adquirida a habilidade, torna-se mais agradável. O RUP não descreve o mecanismo de produção de código nesse nível tão refinado e, certamente, será possível usar a programação em pares em um processo baseado no RUP. Algumas informações sobre essa prática e o *Design anterior ao teste e recriação* (consulte abaixo) são agora fornecidas com o RUP na forma de white papers. Obviamente, não é um requisito utilizar essa prática no RUP nem acreditamos que seja necessário determinar isso.

  A evidência de benefícios *na escala industrial* é insuficiente e anedótica; estudos realizados por ([Nosek98] e [Williams00]) compararam pares com indivíduos (trabalhando isolados), mas boas equipes não trabalham dessa forma. Em um ambiente de equipe, com uma cultura de comunicação aberta, na qual indivíduos sentem-se à vontade para fazer perguntas de seus pares (e líderes de equipes ou gerenciamento) e trabalham para um processo definido, podemos arriscar um palpite de que seria muito difícil discernir os benefícios da programação em pares (em termos de efeito sobre o custo total do ciclo de vida). É normal que as pessoas se reúnam para discutir e resolver problemas em uma equipe bem integrada, sem que sejam obrigadas a fazer isso.

  [Beck00] tem uma visão meio obscura de pessoas e processos quando diz: "Outro recurso poderoso da programação em pares é que algumas das práticas não funcionarão sem ela. Sob estresse, as pessoas mudam. Elas ignorarão os testes escritos. Elas descartarão a recriação …". Elas devem ser práticas naturais que levam a um melhor resultado—por que alguém não as seguiria?

  A iniciação de um bom processo é não-intrusivo; a sugestão que deve ser reforçada no nível de "micro" é desagradável. Entretanto, em algumas circunstâncias, o trabalho em pares é obviamente vantajoso, porque cada um pode ajudar o outro; por exemplo:

  - no início da formação da equipe, quando as pessoas ainda estão se conhecendo;
  - em equipes que não tenham experiência em uma determinada tecnologia nova;
  - em equipes formadas tanto por pessoal experiente como por novatos.

- *Design anterior ao teste e recriação:* Essas são boas técnicas que podem ser aplicadas na disciplina Implementação do RUP. O design anterior ao teste é, em particular, uma forma excelente de esclarecer requisitos em um nível mais detalhado.
- *Cliente on-site:* Muitas tarefas do RUP se beneficiarão bastante em termos de aumento de eficiência ao ter um cliente on-site como um membro da equipe. Com o cliente nesse caminho, a necessidade de vários distribuíveis intermediários (particularmente documentos) pode ser reduzida.

  O XP reluta em dizer que qualquer outra coisa que não seja código deve ser capturada e realça a conversão como o meio de comunicação preferido. Isso conta com a continuidade e familiaridade para obter sucesso. Quando um sistema precisa ser transicionado, então, mesmo para sistemas pequenos, outras coisas precisarão ser produzidas.

  A XP finalmente permite isso, mas algo como uma reflexão posterior; por exemplo, projetar documentos no final de um projeto. De fato, o XP não proíbe a produção de documentos ou outros artefatos (ele apenas não se pronuncia sobre isso), dizendo que você deve produzir apenas aqueles de que realmente precisa; aqueles que são utilizados. O RUP concorda, mas continua a listar e a descrever o que *poderá* ser necessário quando a continuidade e a familiaridade não forem ideais.

- *Padrões de codificação:* O RUP tem um artefato (Diretrizes de Programação) que quase sempre será considerado como obrigatório (a maioria dos perfis de risco do projeto, sendo um condutor principal de ajuste, o tornarão).

- *Integração contínua:* O RUP suporta essa prática por meio de construções nos níveis de subsistema e de sistema (dentro de uma iteração); componentes testados pela unidade são integrados e testados no contexto do sistema emergente.

Práticas do XP que Não São Escaláveis

No entanto, algumas práticas não são escaladas (e o XP não afirma que sejam) e tornaremos seu uso sujeito a essa condição no RUP; por exemplo:

- *Propriedade Coletiva:* É útil ter os membros de uma pequena equipe, responsáveis por um sistema pequeno ou um subsistema de um sistema maior, familiarizados com todos os seus códigos. Porém, se você deseja que todos os membros da equipe estejam em condições iguais de efetuar mudanças em qualquer lugar, isso vai depender da natureza e da complexidade do sistema ou do subsistema. Geralmente, é mais rápido (e seguro) que a correção seja efetuada pelo indivíduo (ou par) que esteja trabalhando no momento com um segmento do código.

  A familiaridade com o código mais bem escrito diminui rapidamente através do tempo, especialmente se ele for complexo pelo aspecto algorítmico.

- *Recriação:* Em um sistema grande, a recriação freqüente não é uma substituta para a falta de arquitetura. [Beck00] diz, "A estratégia de design do XP assemelha-se a um algoritmo escalando uma montanha. Você pega um design simples, torna-o um pouco mais complexo, depois um pouco mais simples e de novo mais complexo. O problema com os algoritmos que escalam montanhas é alcançar o ponto ideal, em que nenhuma pequena alteração pode aprimorar a situação, mas uma alteração grande poderia."

  No RUP, a arquitetura fornece a visualização e o acesso à "grande montanha" tornando flexível um sistema grande e complexo.

- *Metáfora:* Para sistemas maiores, complexos, a arquitetura como metáfora é simplesmente insuficiente. O RUP fornece uma estrutura descritiva muito mais rica para a arquitetura que não é apenas, como [Beck00] a descreve com rejeição, "grandes caixas e conexões".
- *Releases rápidos:* A freqüência com que um cliente pode aceitar e implantar novos releases depende de muitos fatores; normalmente incluindo o tamanho do sistema, que normalmente está correlacionado ao impacto comercial.Um ciclo de dois meses pode ser muito curto para algumas classes de sistema—sendo proibido pela logística de implantação.

E algumas práticas, que são em princípio obviamente seguras e potencialmente utilizáveis no RUP, precisam de pouca elaboração e cuidado quando aplicadas em geral:

- *Design simples:* O XP é muito direcionado à funcionalidade: estórias de usuários são selecionadas, decompostas em tarefas e, em seguida, implementadas. De acordo com [Beck00], o "design correto para o software em qualquer momento determinado é aquele que:

  1. Executa todos os testes
  2. Não tem lógica duplicada…
  3. Declara toda intenção importante para os programadores
  4. Tem poucas classes e métodos possíveis."

  Formatted: Bullets and Numbering

  O XP não acredita na inclusão de nada que não seja necessário agora. Existe um problema nele meio parecido com o problema de otimização local ao lidar com uma classe de requisitos denominada não funcional no RUP. Esses requisitos também agregam valor de negócio para o cliente, mas são mais difíceis de expressar como estórias—algumas das quais o XP chama de restrições caem nessa categoria.

  O RUP também não defende que se faça design a mais do que seja necessário em qualquer tipo de modo especulativo, mas defende o design com um modelo arquitetural mental - modelo este que é uma das chaves para se encontrar os requisitos não funcionais.

  Assim, o RUP concorda com o XP: o "design simples" deve executar todos os testes, mas com a condição de que ele inclua testes que demonstrem que o software atenderá aos requisitos não funcionais. Mais uma vez, isso se torna um problema importante à medida que o tamanho do sistema e a complexidade aumentem, quando a arquitetura for inédita ou quando os requisitos não funcionais forem dispendiosos. Por exemplo, a necessidade de dados

desordenados (para operar em um ambiente distribuído de forma heterogênea) parece tornar o código excessivamente complexo, mas ele ainda será uma necessidade global.

- *Quarenta horas por semana:* Como no XP, o RUP sugere que trabalhar horas extras não deve ser uma condição crônica. A XP não sugere um limite rígido de 40 horas, admitindo diferentes margens de tolerância para o período de trabalho. Os engenheiros de software costumam trabalhar muitas horas sem gratificação extra, trabalham apenas pela satisfação de verem algo concluído, e o gerente não necessariamente precisa colocar arbitrariamente um ponto final nisso.

  O que um gerente não deve fazer é explorar essa situação ou impô-la—e ele deve sempre coletar métricas sobre as horas *realmente* trabalhadas, ainda que não compensadas. Se o registro de horas trabalhadas por alguém parecer alto durante um período extenso, então, certamente isso deve ser investigado. Entretanto, esses problemas devem ser resolvidos em circunstâncias particulares nas quais eles surgem, entre o gerente e o indivíduo, admitindo preocupações que o restante da equipe poderia ter. Quarenta horas é apenas um guia (mas forte).

Funções

No RUP, as tarefas são executadas por funções.[12] As funções também têm responsabilidade por artefatos específicos—a função responsável normalmente criará o artefato e garantirá que as alterações feitas por outras funções (se permitidas) não danifiquem o artefato. Uma função no RUP pode ser executada por um indivíduo ou por um grupo de pessoas. Igualmente, um indivíduo ou grupo pode executar várias funções. Um função não precisa ser mapeada para uma única posição ou "fenda" em uma organização; o mapeamento de funções para organizational units também pode ser feito de muitos-a-muitos.

Funções do RUP

O RUP define um total de 30 funções: não existe um mapeamento exato para disciplinas porque uma função—por exemplo, um arquiteto de software—não fica necessariamente confinado a uma disciplina, mas aproximadamente (colocando a função onde ela pertencente):

- Modelagem de Negócios tem três funções
- Requisitos tem cinco funções
- Análise & Design tem seis funções
- Implementação tem três funções
- Teste tem duas funções
- Implantação tem quatro funções
- Gerenciamento de Configuração & Alterações tem duas funções
- Gerenciamento de Projeto tem duas funções
- Ambiente tem três funções

Por Que Há Tantas Funções no RUP?

As funções no RUP são utilizadas para particionar tarefas e para discriminar sutilmente as habilidades e competências necessárias para executar a função, o que ajuda a guiar a seleção da equipe para executar essas funções. O nível de divisão também facilita a identificação das novas posições organizacionais quando a importância de uma função é alterada. Por exemplo, em um projeto pequeno, informal, a função de Coordenador de Projeto e Gerente de Configuração (funções do RUP) pode ser bem executada pelo mesmo indivíduo; em um projeto de espaço aéreo militar, maior e formal, o trabalho do Gerente de Configuração poderá ser bem especializado e oneroso o suficiente para garantir uma equipe pequena. Ao descrever funções dessa maneira, o RUP facilita esse mapeamento.

Funções do XP

[Beck00] identifica sete funções aplicáveis ao XP e, então, continua a descrever as responsabilidades das funções, as habilidades e os traços requeridos das pessoas que as executarão. Essas funções são:

- Programador
- Cliente
- Testador
- Rastreador
- Técnico
- Consultor
- Diretor

São feitas referências a essas funções em alguns dos outros livros de XP, em locais que elaboram as tarefas que a função executa.

[12] Para ser mais preciso (e exato), as tarefas são executadas por indivíduos ou equipes que *desempenham* as funções.

A diferença no número de funções do XP e do RUP é facilmente explicada:

- O XP não está convertendo todas as disciplinas do RUP.
- As funções do XP estão realmente mais próximas de posições do que as funções do RUP; por exemplo, um programador do XP realmente executa várias funções do RUP—aquelas do Implementador, Integrador e Revisor de Código—e elas requerem poucas competências diferentes.

Quando funções do RUP são mapeadas para um projeto pequeno (como no roteiro de projetos pequenos do RUP), o número de funções semelhantes do XP, ou seja posições, que são mapeadas reduz consideravelmente para 30. No roteiro de projetos pequenos, o número de posições é cinco, conforme mostrado na seguinte tabela.

Funções do RUP Mapeadas para um Projeto Pequeno da Empresa ABC

Cargo na Empresa ABC	Função RUP
Coordenador de Projeto	Coordenador de Projeto Engenheiro de Processo Gerenciador de Implantação Revisor de Requisitos Revisor de Arquitetura
Executivo na Empresa ABC	Revisor de Projeto Investidores Revisor de Requisitos
Programador-chefe	Analista de Sistemas Especificador de Requisitos Designer da Interface com o Usuário Arquiteto de Software Revisor de Design Engenheiro de Processo Especialista em Ferramentas Gerenciador de Configuração Gerenciador de Controle de Alterações *e, em uma extensão menor, as mesmas funções do* Programador
Programador	Designer Implementador Revisor de Código Integrador Designer de Teste Testador
Assistente Administrativo	*Responsável por:* • *manter o Web site do "Projeto Pequeno"* • *ajudar a função* Coordenador de Projeto *a planejar atividades* • *ajudar a função* Gerente de Controle de Alterações *a controlar alterações em artefatos* • *também pode fornecer assistência a outras funções, conforme necessário*

Conclusões

O XP não é a mesma coisa que o RUP. O RUP é uma estrutura de processo a partir da qual determinados processos podem ser configurados e, em seguida, instanciados. O RUP *precisa* ser configurado e isso é realmente uma etapa requerida definida no próprio RUP. Especificamente falando, deve-se comparar uma versão adaptada do RUP com XP, com o RUP adaptado às distinções do projeto que o XP explicitamente estabelece (e as que podem ser inferidas). Tal processo de RUP adaptado dessa forma poderia acomodar muitas práticas do XP (como programação em pares, design anterior ao teste e recriação), mas ainda não seria idêntico ao XP, por causa do reconhecimento da importância da arquitetura, abstração (em modelagem) e risco e de sua estrutura diferente no tempo (fases, iterações).

O RUP permitirá a construção de processos para acomodar projetos que estão fora do escopo do XP em escala ou tipo. O RUP é pesado apenas por ser uma *descrição* completa de uma família de processos que pode ser leve ou pesada—em artefatos, distribuíves, formalidade, prescrição, cerimônia ou qualquer outra medida de "peso"—na implementação, conforme desejado. O XP é certamente leve por ser direcionado intencionalmente à implementação de um processo (leve), mas as descrições do XP (pelo menos nos livros) também não são elaboradas. Portanto, em uma implementação do XP, haverá itens que precisam ser descobertos, inventados ou definidos rapidamente. Então, comparado com o RUP, o XP é também leve no material descritivo.

Isso provavelmente irá mudar, de fato, já deve estar mudando com a publicação de outros dois livros, um dos quais, [Succi01], terá 512 páginas. No entanto, como as coisas se encontram, o perfil do esforço de adoção será diferente entre as duas abordagens. O RUP desloca muito do esforço direto, em requisitos de treinamento e ajuste de processos. É mais do que provável que uma organização também irá ajustar o RUP para aplicação em toda a organização em tipos e tamanhos específicos de projetos e utilizará os resultados em vários projetos. Com o XP, haverá algum treinamento direto requerido, mas o restante do esforço de adoção será espalhado através de um projeto, conforme ele se expande e captura todas essas coisas auxiliares que passaram a ser necessárias para que o XP funcione. O XP obviamente não motiva a captura de "memória corporativa", deixando uma organização de adoção (se ela não salvar sua experiência de processo) vulnerável à troca de turno da equipe.

Rotular o RUP como pesado e o XP como leve sem qualificação adicional causa prejuízos obscurecendo o que cada um é e o que cada pretende fazer. E, quando isso é feito de uma forma pejorativa, é simplesmente uma atitude sem sentido. São as implementações deles como processos que serão "pesadas" ou "leves" e eles devem ser pesados ou leves conforme as circunstâncias demandarem.

O XP não tem um formato livre, tudo resulta em disciplina—ele focaliza minuciosamente um aspecto específico de desenvolvimento de software e uma forma de entregar valor e é bem prescritível sobre a forma que isso deve ser obtido.

A abrangência do RUP é mais ampla e profunda, o que explica seu "tamanho" aparente. No entanto, no nível micro do processo, o RUP permite e oferece ocasionalmente alternativas igualmente válidas, o que o XP não faz; por exemplo, a prática de programação em pares. Isso não é uma crítica ao XP; simplesmente uma ilustração de como o XP, como seu nome implica, estreitou seu foco.

Referências

[Beck00] *Extreme Programming Explained,* Kent Beck, Addison-Wesley, 2000

[Beck01] *Planning Extreme Programming,* Kent Beck, Martin Fowler, Addison-Wesley, 2001

[Boehm00] *Software Cost Estimation with COCOMO II,* Barry W. Boehm et al, Prentice Hall PTR, 2000

[Fowler99] *Refactoring: Improving the Design of Existing Code,* Martin Fowler et al, Addison-Wesley, 1999

[Jeffries01] *Extreme Programming Installed,* Ron Jeffries, Ann Anderson, Chet Hendrickson, Addison-Wesley, 2001

[Kruchten00] *The Rational Unified Process, An Introduction, Second Edition,* Philippe Kruchten, Addison-Wesley, 2000

[Martin01] *Extreme Programming in Practice,* Robert C. Martin, James W. Newkirk, Addison-Wesley, 2001 (ainda não publicado)

[Nosek98] *The Case for Collaborative Programming,* John T. Nosek, *Comm. ACM,* Vol. 41, No. 3, 1998, pp. 105-108

[Succi01] *Extreme Programming Examined,* Giancarlo Succi, Michele Marchesi, Addison-Wesley, 2001 (ainda não publicado)

[Williams00] *Strengthening the Case for Pair Programming,* Laurie Williams, Robert R. Kessler, Ward Cunningham, Ron Jeffries, IEEE Software, Vol. 17, No. 4, 2000, pp. 19-25

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

Field Code Changed
Field Code Changed
Field Code Changed