Comparação entre Metodologias RUP e XP

Carlos G. Vasco, Marcelo Henrique Vithoft, Paulo Roberto C. Estante

Programa de Pós Graduação em Informática Aplicada
Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba, PR – Brasil
cgvasco@gmail.com, vithoft@ppgia.pucpr.br, estantep@ppgia.pucpr.br

Abstract. *This paper compares the software development methodology RUP and XP, two methods that aim at risk reduction by making the system development in an incremental fashion. The paper focus on the structure of the processes and in the activity division, concluding that XP is most suitable for minor projects, and RUP being the most suitable for major projects due to requirements, communication and documentation.*

Resumo. *Este artigo compara as metodologias de desenvolvimento de software RUP e XP, dois métodos que buscam a redução de riscos desenvolvendo o sistema de forma incremental. O artigo tem como foco a estruturação dos processos e a divisão das atividades, concluindo com uma indicação de uso do XP para projetos menores, e com o uso do RUP para projetos maiores, devido a requerimentos, comunicação e documentação.*

1. Introdução

O sucesso de um projeto de desenvolvimento de software começa no devido planejamento e na escolha de uma metodologia compatível com as características do mesmo.

A etapa do planejamento deve estruturar o processo de desenvolvimento em torno dos recursos disponíveis (i.e. orçamento, força de trabalho, tempo) visando à entrega de um produto de qualidade que atenda às necessidades do cliente dentro do prazo previsto.

A metodologia tradicional de desenvolvimento, em cascata (*waterfall*), consiste em etapas que são seguidas estritamente de forma seqüencial: o produto de cada etapa é tomado como base para o início da próxima. Esta metodologia implica que um enorme esforço deve ser empregado nas fases de levantamento de requerimentos e de desenho da arquitetura. Se durante uma das fases finais do projeto (i.e. implementação, testes) for localizado uma falha, ou mesmo ocorrer uma alteração de requisitos, o projeto pode sofrer enormes (e dispendiosas) alterações.

Tanto o Rational Unified Process (RUP) quanto o Extreme Programming (XP) são metodologias que dividem o desenvolvimento de um projeto em ciclos, visando redução de riscos (e custos) nos casos de alterações de requisitos ou funcionalidades.

O presente trabalho tem o propósito de analisar as similaridades e diferenças presentes nas metodologias empregadas pelos processos RUP e XP. O capítulo 2 apresenta os tipos de metodologia de projeto, já o 3 define os processos RUP e XP, o 4 compara duas metodologias. Por fim o capítulo 5 sintetiza as idéias apresentadas.

2. Tipos de Metodologias de Projeto

2.1. Metodologias Iterativas

Metodologias iterativas têm como objetivo o desenvolvimento de projetos de forma incremental. A cada iteração uma parte do sistema é desenvolvida, sendo o produto de cada nova iteração superior à da iteração anterior.

Desenvolver um sistema de forma incremental traz vantagens ao projeto. Ao se construir gradualmente o sistema, os próprios desenvolvedores aprendem sobre o mesmo, possibilitando a localização de futuros problemas em fases iniciais. Outra vantagem de se construir um sistema gradualmente é a inata acomodação para mudanças, dado que o "todo" não é desenvolvido em apenas uma etapa.

2.2. Metodologias Ágeis

Metodologias ágeis também dividem o desenvolvimento do software em iterações, buscando redução de riscos ao projeto. Ao final de cada iteração, uma versão (*release*) funcional do produto, embora restrita em funcionalidades, é liberada ao cliente.

As metodologias ágeis destacam aspectos humanos no desenvolvimento do projeto, promovendo interação na equipe de desenvolvimento e o relacionamento de cooperação com o cliente. Comunicação face-a-face é preferida à documentação compreensiva.

2.3. Contrastes de Metodologias Iterativas e Ágeis

As metodologias ágeis compartilham as idéias de construção incremental de sistemas provenientes das metodologias iterativas, porém os períodos das iterações são geralmente menores, medidos em semanas, enquanto iterações de processos iterativos são geralmente medidos em meses.

3. Metodologias de Desenvolvimento

3.1. Rational Unified Process

RUP é uma metodologia iterativa de desenvolvimento. RUP é adaptável, podendo ser customizada para diversos tipos e tamanhos de produtos e projetos de software. A Rational Software (atual divisão da IBM) desenvolveu e mantém o RUP.

A metodologia RUP identifica cada ciclo de desenvolvimento do projeto em quatro fases, cada uma com respectivos marcos de finalização definidos (chamados *milestones*). Os *milestones* são os indicadores de progresso do projeto, e são usados como base para decisões para continuar, abortar, ou mudar o rumo do projeto. As fases do RUP são:

1. Início (*Inception*): determinação do escopo do desenvolvimento, sendo levantado uma visão do produto final a partir de um caso de uso (básico) definido.
2. Elaboração (*Elaboration*): planejamento de atividades e recursos necessários, onde são definidas funcionalidades e a arquitetura a ser desenvolvida.

3. Construção (*Construction*): implementação do software, contrução do código. Em projetos grandes esta fase pode ser segmentada em várias iterações, visando à divisão em partes menores e mais facilmente gerenciadas.
4. Transição (*Transition*): o produto é passado aos usuários. Nesta fase ocorre treinamento dos usuários (e possíveis mantenedores) e a avaliação do produto ("*beta-testing*").

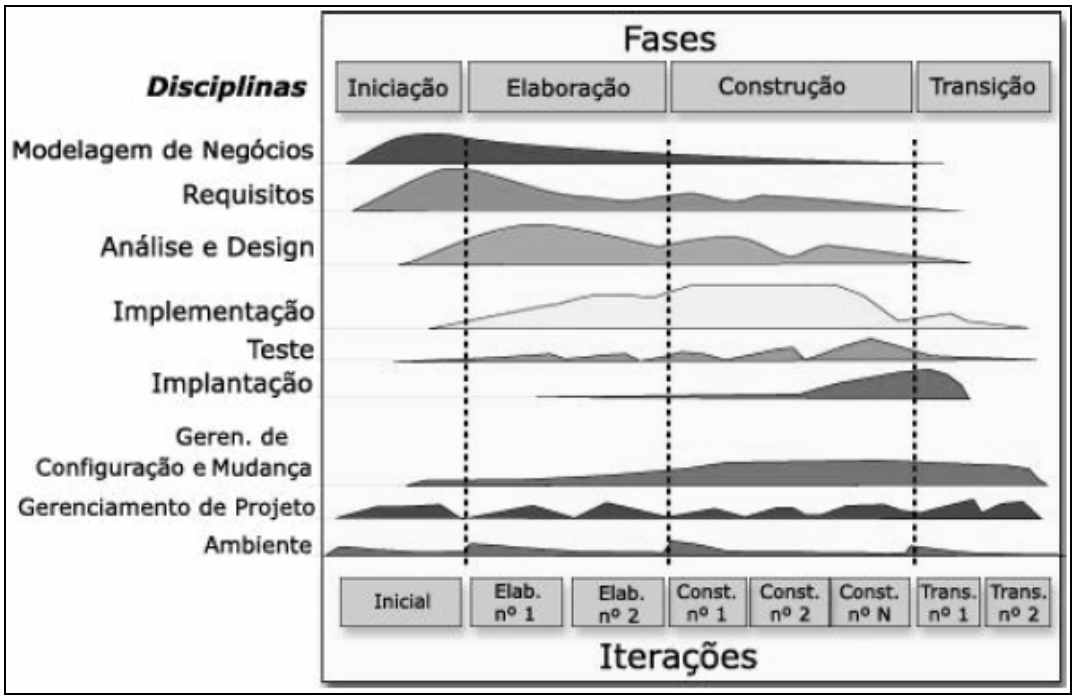

Figura 1. Fases do RUP - Adaptado de [Luiz 2005]

A Figura 1 apresenta a arquitetura de projetos que seguem a metodologia RUP. O eixo horizontal define aspectos dinâmicos, como ciclos, fases, iterações e marcos (*milestones*), já a vertical define os aspectos estáticos, como atividades, disciplinas, artefatos e papéis. Alguns dos fatores citados serão abordados ao longo do capítulo 4.

Os valores do RUP são [Beck apud Runeson 2004]: desenho do sistema via casos de uso, ajuste do processo, e ferramenta de suporte.

3.2. Extreme Programming

XP é uma metodologia ágil, para o desenvolvimento de projetos de IT cujas especificações são passíveis a alterações.

As iterações do XP costumam ser curtas, provendo constantes versões do produto (*releases*) para o cliente, que por sua vez provê comentários e opiniões que realimentam a próxima iteração. O objetivo do XP é tornar o projeto flexível, diminuindo o custo a possíveis mudanças. O código produzido é tomado como indicador de progresso do projeto.

O ciclo XP é dividido em seis fases:

1. Exploração: nessa fase o cliente escreve cartões de estórias, cada um contendo uma funcionalidade desejada para o primeiro *release.*
2. Planejamento: ocorre definição de prioridades entre as estórias junto com o cliente. Nesta etapa os programadores estimam o esforço e o cronograma para cada uma das estórias.
3. Iterações para *Release*: nessa fase ocorrem diversas iterações até o primeiro *release* ser completado. Na primeira iteração é criado o sistema com toda a arquitetura, nas iterações seguintes serão adicionadas às funcionalidades de acordo com as prioridades estabelecidas.
4. Validação para Produção (*Productionizing*): nessa fase são feitos testes extensivos e verificações para validação do software para ser utilizado em ambientes de produção.
5. Manutenção: após o primeiro *release* para produção, há novas edições do sistema com novas funcionalidades.
6. Morte: quando não há mais estórias a serem implementadas, quando o cliente está satisfeito com o código.

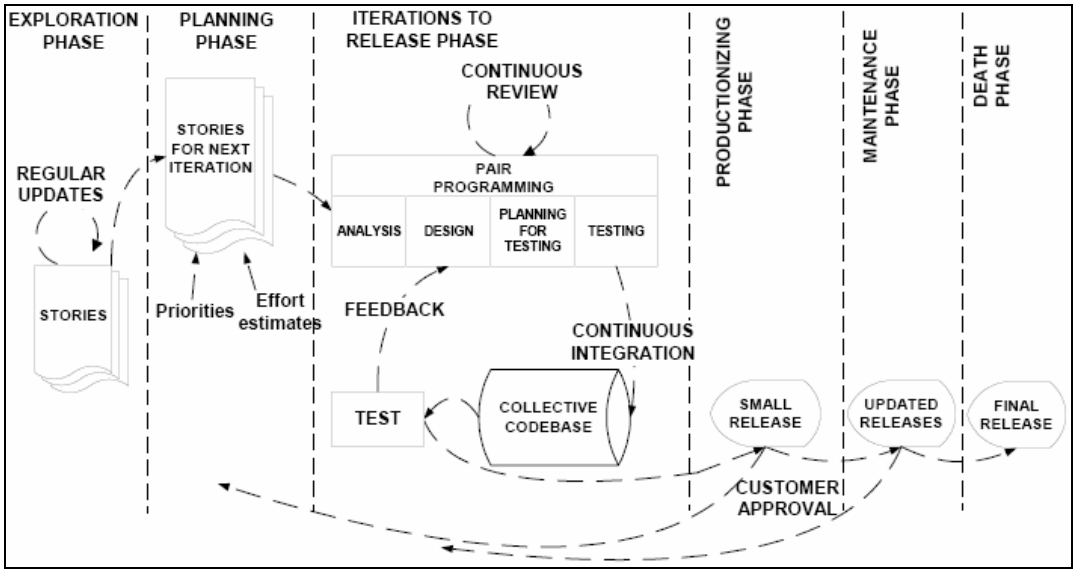

Figure 2. Fases do XP – Adaptado de [Abrahamsson et al 2002]

Os valores do XP são: simplicidade, coragem, *feedback*, e comunicação.

4. Comparação entre RUP e XP

As metodologias RUP e XP têm seu funcionamento baseado em iterações, são orientadas ao cliente e baseadas em papel. Uma análise superficial nos diria que tratam a dinâmica de desenvolvimento de software da mesma forma. Porém através da análise dos tópicos descritos pelo presente capítulo as diferenças serão verificadas por aspectos como a forma e freqüência que os artefatos são gerados, quantidade de papéis, etc.

4.1. Alocação de Tempo e Esforços

RUP segue 4 fases seqüenciais (descritas no item 3.1) constituindo um ciclo de desenvolvimento, produzindo uma nova versão de software. Em cada fase há um número de iterações, as quatro fases têm seu foco em diferentes atividades, podendo ocorrer em paralelo. A primeira fase, chamada de início foca o modelamento do negócio e a definição dos requisitos. A fase de elaboração foca em projetar (*design*), a de construção dá enfoque a implementação e aos testes e por fim a fase de transição onde a implantação e o gerenciamento de modificações são verificados.

O aspecto tempo em XP é diretamente relacionado com código produzido. No começo do projeto o foco é o núcleo do sistema, e após facilidades extras. Porém como as liberações são definidas pelos clientes, os mesmos irão delimitar o tempo do projeto a partir de seu grau de satisfação com o software recebido.

4.2. Artefatos

Um artefato é um pedaço de informação que é produzida, modificada ou usada por um processo [Kruchten 2000]. Como exemplos de artefatos têm-se modelos, código fonte e arquivos executáveis.

A comunicação dentro de um processo RUP é baseada em artefatos. Já em XP é baseada em comunicação oral, direta, o que restringe o uso de XP em projetos com grande distribuição geográfica.

A pouca evidência de artefatos do XP, com foco em estórias de usuário e código, é visto como um fator que aumenta o risco do projeto, o conhecimento fica vinculado aos desenvolvedores e ao código.

4.3. Atividades e Papéis

RUP define uma atividade como sendo o trabalho realizado por um papel, usando artefatos de entrada e produzindo artefatos de saída. Os papéis (total de 30) definem o comportamento e as responsabilidades do individuo, estão agrupados em nove disciplinas:

1. Gerência de Projeto (2 papéis): tem como objetivo prover meios para a entrega do produto para o cliente que atenda as suas necessidades, gerenciando os riscos do projeto.
2. Modelamento de Negócio (3 papéis): visa o entendimento da estrutura onde o software será aplicado e os problemas atuais do cliente. Esta atividade deve assegurar que o cliente, os seus usuários e os desenvolvedores tenham um entendimento comum do produto a ser entregue.
3. Requerimentos (5 papéis): traduz as necessidades do sistema em forma de casos de uso, desenhando a interface com o usuário.
4. Analise e Desenho (6 papéis): especifica a forma de implementação (arquitetura) dos requerimentos (casos de uso).
5. Implementação (3 papéis): implementa as classes e os objetos em formas de componentes, os quais são individualmente testados.

6. Teste (2 papéis): testa e verifica se o produto funciona como o esperado, documentando falhas e problemas. Provê *feedback* à gerência do projeto sobre a qualidade do software.
7. *Deployment* (4 papéis): tem como objetivo a distruibuição, instalação e teste (quando "*beta*") em campo, provendo treinamento e possíveis migrações (banco de dados) entre o sistema anterior e o atual.
8. Configuração e Controle de Mudanças (2 papéis): o objetivo destas atividades (Gerência de Configuração e Controle de Mudanças) concentra-se na garantia da rastreabilidade de versões dentro de um projeto. Através do definição de uma política adequada e de ferramentas específicas para versionamento (ClearCase, CVS, SVN) é possível garantir o mapeamento de alterações efetuadas bem como a recuperação de códigos e versões antigas.
9. Ambiente (3 papéis): provê suporte adequado à organização do projeto, em ferramentas, métodos e processos.

XP [Beck e Fowler 2000] apresenta apenas quatro atividades básicas: produção de código, testes, *listening* (escutar o cliente), e desenho. XP define sete papéis:

1. Programador: escreve o código e realiza testes individuais.
2. Cliente: é o responsável por escrever as estórias e as prioridades das mesmas.
3. Testador: tem como objetivo auxiliar o cliente a criar testes funcionais. Os testes devem ser realizados regularmente e seus resultados divulgados.
4. *Tracker*: provê *feedback* no processo, comparando as estimativas com os resultados obtidos. Analisa o progresso de cada iteração e avalia se os objetivos podem ser alcançados com os recursos providos.
5. Técnico (*Coach*): tem como responsabilidade o projeto como um todo, guiando os outros membros da equipe.
6. Consultor: membro externo que auxilia o time com assuntos (problemas) técnicos específicos.
7. "*Big Boss*": responsável pelo projeto, toma decisões e está em constante comunicação com a equipe para diagnosticar possíveis problemas ou falhas no processo.

Ao compararmos as definições das atividades (disciplinas) e os papéis, temos que o RUP faz uma divisão de tarefas de forma específica, enquanto a divisão de papéis proposta pelo XP tem um caráter de "uso-geral", sem atribuições específicas dentro das atividades.

4.4 Ferramentas

O XP não especifica nenhuma ferramenta em específico para o processo, embora haja ferramentas livres como o *XPlanner* e o *Junit*. Já o processo no RUP utiliza softwares, como o *Rational Rose*, que devem ser adquiridos da própria IBM, juntamente com a documentação.

4.5 Responsabilidade do Código

RUP trata o código (geralmente de sistemas grandes) em subsistemas, e para cada subsistema existem membros responsáveis pelo mesmo.

Já o XP trata o código como coletivo, qualquer programador que encontre algum problema, ou algum trecho que possa ser otimizado, tem permissão para fazê-lo. Isso pode agilizar o processo no caso de um programador necessitar corrigir algum módulo para interagir corretamente com o seu. Porém isso pode ser algo prejudicial, pois podem ocorrer casos onde o código aparentemente incorreto teve um motivo para ser estruturado da maneira que foi.

4.6 Diretivas de Projeto

RUP é baseado em casos de uso, as descrições do uso do sistema são continuamente implementadas, integradas e testadas. Já o XP aplica o projeto baseado em teste, casos de teste são implementados antes da escrita do código. XP possui estórias de usuário para guiar o que deve ser implementado. Essas estórias são descrições mais simples se comparadas aos casos de uso do RUP.

As duas metodologias indicam que o projeto completo não pode ser planejado em detalhe. RUP indica modificação continua dos planos, enquanto o XP propõe planejar em detalhes somente o futuro próximo.

5. Conclusão

Metodologias para o desenvolvimento de software independente de seus processos específicos buscam reduzir riscos e aumentar a qualidade do produto gerado. As metodologias RUP e XP têm esse fim, pois apresentam mesmos valores como, envolvimento do cliente, iterações, testes contínuos e flexibilidade. Porém busca-se esses objetivos de forma diferente, através de implementações diferentes.

De forma geral o XP se apresenta como uma metodologia a ser utilizada em projetos onde os requisitos são voláteis ou não claros sendo assim muito flexível, porém existe uma limitação de uso em equipes pequenas, pois não dá ênfase à documentação e sim a comunicação oral restringindo sua aplicação em projetos com equipes distribuídas geograficamente. A divisão de atividades (tarefas e papéis) no XP também não é muito específica, sendo uma desvantagem para a divisão de responsabilidades em projetos grandes.

O RUP estrutura o projeto fazendo uma divisão bem definida de atividades (tarefas e papéis) e define uma coleção de artefatos que são usados como produtos de entrada e de saída de processos. Essa estruturação (com auxílio de *softwares*) do processo permite que o RUP seja utilizado em projetos grandes, e com distribuição geográfica, com um custo (esforço) adicional de gerência do projeto. Esse custo adicional pode não ser justificável em pequenas equipes.

Referências

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta J. (2002) "Agile Software Development Methods: Review and Analysis", Espoo 2002, VTT Publications 478.

Beck, K. and Fowler, M. (2000) "Planning Extreme Programming", Addison Wesley, 1a edição.

Kruchten, P. (2000) "The Rational Unified Process – An Introduction", Addison-Wesley 2a edição.

Luiz, R. (2005) “Obtendo Qualidade de Software com o RUP”, TCC, Universidade de Uberaba.

Pollice, G. (2003) "Using the IBM Rational Unified Process for Small Projects: Expanding Upon eXtreme Programming", IBM whitepaper, ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/TP183.pdf, Abril.

Runeson, P. and Greberg, P. (2004) "Extreme Programming and Rational Unified Process – Contrasts or Synonyms?", Lund University, Sweden.

Smith, J. (2003) "A Comparison of the IBM Rational Unified Process and eXtreme Programming", IBM Whitepaper, ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/TP167.pdf, Abril.