Um Método de Desenvolvimento de Sistemas de Grande Porte Baseado no Processo RUP

Francisco Flávio de Souza[1], Rosana Teresinha Vaccare Braga[2]

[1]Mestrado em Informática Aplicada – Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Edson Queiroz – 60.811-905 – Fortaleza – CE – Brasil

[2]ICMC – Universidade de São Paulo (USP) São Carlos – SP – Brasil

fsouza@uol.com.br, rtvb@icmc.usp.br

Abstract. *When developing large systems, besides the usual care that needs to be taken with analysis, design and implementation, special care with requirements elicitation is needed, as they are prone to possible changes before the delivery of the final system. In this paper we suggest an adaptation of the RUP process to solve this problem, promoting the incremental development of subsystems. As each subsystem is deployed, risks are mitigated, process efficiency is enhanced through the reuse of the acquired knowledge and produced artifacts, as well as allowing the customer to see the results of his investments before the conclusion of the whole project.*

Resumo. *Ao lidar com o desenvolvimento de sistemas de grande porte, além da preocupação com análise, projeto e implementação, uma atenção ainda maior deve ser dada à elicitação de requisitos, por estarem sujeitos a possíveis mudanças até a entrega do produto final. Neste artigo, sugere-se uma adaptação ao processo RUP para resolver esse problema, promovendo o desenvolvimento de subsistemas de forma incremental. A cada subsistema implantado, os riscos são mitigados, aprimora-se a eficiência do processo pelo reuso do conhecimento adquirido e dos artefatos produzidos, e propicia-se ao cliente retorno de parte do seu investimento antes da conclusão de todo o projeto.*

1. Introdução

Ao se desenvolver um sistema de grande porte – admitindo-se que essa denominação tenha-lhe sido atribuída devido ao esforço necessário para seu desenvolvimento, segundo projeções a partir de métricas, tais como: a quantidade de pontos-de-função, o número de instruções de código-fonte, o total de recursos (em hora/homem/mês, por exemplo) ou, simplesmente, os custos necessários à sua implementação [1] – é inquestionável o aumento de sua complexidade em relação a sistemas de menor porte, visto que, além de um número maior de artefatos ter que ser gerado e administrado durante o seu ciclo de vida, os problemas inerentes à construção de qualquer sistema passam, neste caso, a ser incrementados também. O mesmo pode-se dizer em relação aos riscos do projeto que, segundo Fowler [2], possivelmente conterão ocorrências, no mínimo, em alguma(s) dessas categorias: riscos de requisitos, riscos tecnológicos, riscos de habilidades e riscos políticos. Já a qualidade do produto de software resultante, quer seja interna (a totalidade de características relativas à sua implementação), externa (a totalidade de características relativas à sua execução) ou ainda a de uso (a visão do usuário final sobre a qualidade do produto) [3] é outra questão desafiadora a ser

enfrentada nesse caso. Dessa forma, o desenvolvimento seqüencial (ou em "cascata") não seria uma escolha das mais indicadas, visto que, como ele assume que não haverá mudanças nos requisitos após o seu levantamento e que é possível propor a solução mais adequada para o projeto logo na fase inicial, sua utilização mais coerente seria em sistemas de pequeno porte, com pouca novidade funcional e tecnológica [4].

Este trabalho propõe um modelo de desenvolvimento de sistemas de grande porte com enfoque em casos de uso – especificação de uma seqüência de interações entre um sistema e um agente externo (ator) [5] – e em arquitetura de software – a estrutura dos componentes de um sistema, seus inter-relacionamentos e as diretrizes do seu projeto [6] – que, embasado na Teoria Geral dos Sistemas e no princípio de "dividir para conquistar", aplica o processo RUP® (*Rational Unified Process®*) [1, 4] iterativamente ao longo do ciclo de vida do sistema e promove a antecipação da implantação dos subsistemas que vão sendo concluídos, começando pelos mais críticos em termos da importância de suas funcionalidades para a organização e das novidades funcionais e técnicas envolvidas, a fim de atacar com antecedência os maiores riscos e, assim, reduzir as chances de insucesso do projeto [4].

A seguir, na seção 2, é apresentada uma visão geral do RUP; na seção 3, é exposta a proposta do método; na seção 4, uma discussão sobre o método, comparando-o com outra proposta existente para desenvolvimento em grande porte; e, na seção 5, as conclusões.

2. Visão Geral do RUP

O RUP é um processo de engenharia de software que oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Sua meta é garantir a produção de software de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e um orçamento previsíveis. Pode ser visto como tendo duas dimensões (Figura 1): o eixo horizontal representa o tempo e mostra os aspectos do ciclo de vida do processo à medida que se desenvolve (fases), enquanto o eixo vertical representa as disciplinas, que agrupam as atividades de maneira lógica, por natureza.

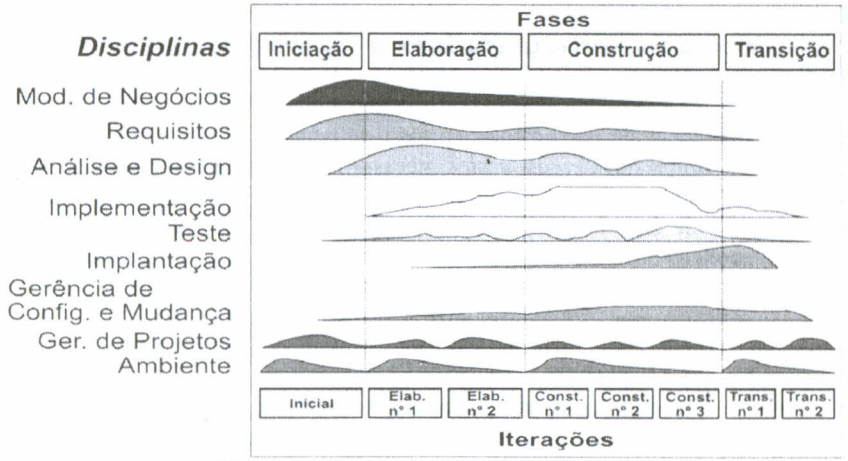

Figura 1. Fases e disciplinas do RUP [4]

A primeira dimensão representa o aspecto dinâmico do processo e é expressa em termos de fases, iterações e marcos (Figura 2). A segunda representa o aspecto estático do processo, como ele é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis do processo.

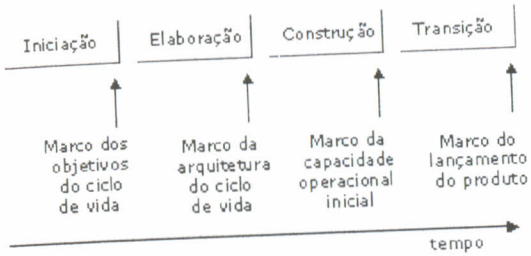

Figura 2. Fases e marcos do RUP [4]

3. O Método

3.1. Objetivos

O método proposto neste artigo pretende:

- Oferecer ao cliente, num período de tempo bem mais curto que o do desenvolvimento de todo o sistema, partes dele (os subsistemas ou módulos) já concluídas, proporcionando-lhe retorno de parte do que já foi investido;
- Possibilitar ao cliente melhor condição de avaliar o que está efetivamente sendo feito com seu investimento (através de métricas externas e, principalmente, de uso), no sentido de atender suas necessidades;
- Ajudar a reduzir "pressões" políticas, erros nas estimativas de prazos e custos globais, incertezas devido a novas tecnologias, entre outros riscos;
- Tornar as condições contratuais em torno de conformidades, prazos, qualidade de produto, etc. mais fáceis de serem aferidas e, em certos casos, justificadas (quando não atingidas) e renegociadas;
- Tornar o trabalho da empresa que realiza o desenvolvimento do sistema mais "transparente" e confiável na medida em que a consecução dos objetivos é atingida e comprovada, na prática, pelos próprios usuários;
- Permitir que a maioria dos problemas detectados somente após a implantação do sistema possa vir à tona bem antes disso - logo com o(s) primeiro(s) subsistema(s) - e, conseqüentemente, possa ser tratada mais cedo;
- Usar a existência do software (já implementado e implantado) como fator inibidor de mudanças durante o desenvolvimento do restante da aplicação e considerar qualquer alteração de requisitos como uma evolução de produto, contratualmente falando, com possível revisão de prazos, recursos, custos, etc.

3.2. Fases

As fases do método proposto neste trabalho são as mesmas do processo RUP, com a diferença básica de que elas são realizadas não apenas no escopo do sistema, mas também de cada subsistema, havendo, ainda, um revezamento entre elas ao longo do tempo (Figuras 3 e 4). Isso permite que, após a fase de transição (implantação) do(s) primeiro(s) subsistema(s), o conhecimento adquirido e os artefatos produzidos em seu desenvolvimento sejam devidamente explorados nos subsistemas que ainda restam, além de possibilitar melhor condição de avaliação do andamento e das estimativas do

projeto como um todo, já que foi adquirida uma amostragem de desenvolvimento do sistema, em cada uma de suas fases.

3.2.1. Concepção (ou Iniciação)

a) do sistema

Objetivos principais:

- Estabelecer o escopo do projeto, além de uma visão operacional básica e de critérios gerais de aceitação;
- Discriminar os casos de uso críticos do sistema, com os principais cenários de operação (mas sem detalhamento ainda);
- Apresentar uma visão sistêmica (com os subsistemas) do projeto, baseada nos casos de uso, que serão agrupados pela afinidade e natureza de suas funcionalidades. Caso algum subsistema tenha uma complexidade considerável, pode-se proceder de forma semelhante, considerando-o como um sistema composto de subsistemas, e aplicar o mesmo procedimento sobre eles;
- Exibir pelo menos uma opção de arquitetura geral;
- Numa primeira estimativa, apresentar o custo geral e a programação para o projeto inteiro;
- Estimar riscos em potencial, estrategicamente falando;
- Com a participação efetiva do cliente, decidir a seqüência de subsistemas a serem abordados, considerando-se a sua importância e o seu grau de risco;
- Preparar o ambiente básico de suporte para o projeto;
- Seqüencialmente, realizar a concepção de cada um dos subsistemas que comporão o sistema.

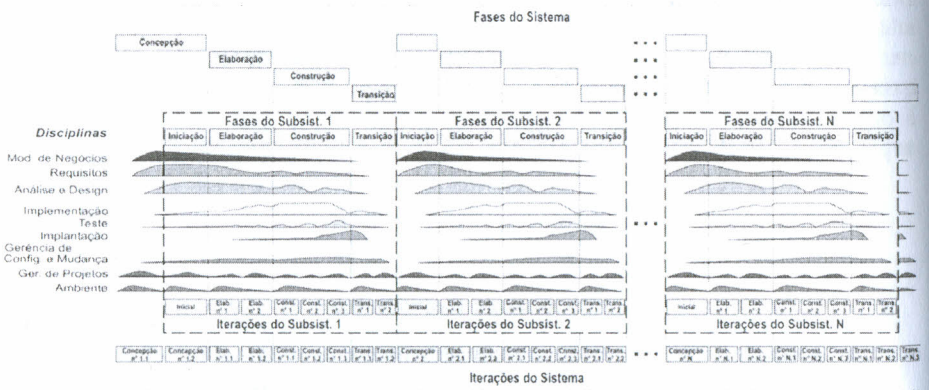

Figura 3. Fases e disciplinas do método proposto

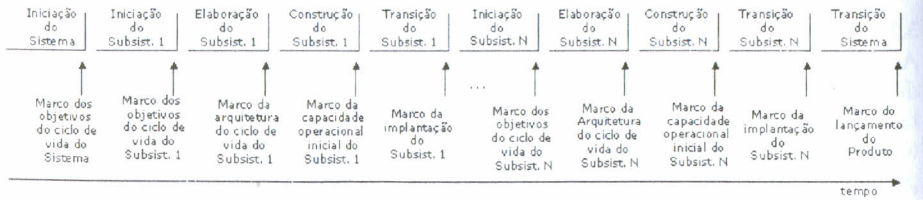

Figura 4. Fases e marcos do método proposto

b) de um subsistema

Objetivos principais:

- Estabelecer o escopo do subsistema e as condições limite, incluindo uma visão operacional, critérios de aceitação específicos e o que deve ou não estar no produto;
- Discriminar os casos de uso críticos do subsistema, os principais cenários de operação que direcionarão as funcionalidades e as alternativas (*trade-offs*) de seu *design*;
- Exibir, e talvez demonstrar, pelo menos uma opção de arquitetura para alguns cenários básicos, levando-se em consideração a arquitetura geral já proposta;
- Estimar o custo e a programação para o projeto (e estimativas detalhadas para a fase de elaboração imediatamente a seguir), confrontando-os com o custo e a programação geral (do sistema);
- Estimar riscos potenciais pertinentes ao subsistema;
- Verificar (e evoluir) o ambiente de suporte para o projeto.

3.2.2. Elaboração

a) do sistema

Objetivo principal: seqüencialmente, realizar a elaboração de cada um dos subsistemas que comporão o sistema.

b) de um subsistema

Objetivos principais:

- Assegurar que a arquitetura, os requisitos e os planos sejam estáveis o suficiente e que os riscos sejam amplamente diminuídos a fim de determinar com segurança o custo e a programação para a conclusão do desenvolvimento;
- Tratar todos os riscos significativos do ponto de vista da arquitetura do projeto (do subsistema);
- Estabelecer uma *baseline* com base no tratamento dos cenários significativos do ponto de vista da arquitetura, que normalmente expõem os maiores riscos técnicos do projeto;
- Produzir um protótipo evolutivo dos componentes de qualidade de produção, assim como um ou mais protótipos descartáveis a fim de diminuir riscos específicos;
- Demonstrar que a arquitetura de *baseline* suportará os requisitos do sistema a um custo justo e em tempo justo;
- Estabelecer um ambiente de suporte;
- Para atingir esses objetivos básicos, é também importante configurar o ambiente de suporte para o projeto (pela criação de *templates* e diretrizes, a configuração de ferramentas, etc.)

3.2.3. Construção

a) do sistema

Objetivo principal: seqüencialmente, realizar a construção de cada um dos subsistemas que comporão o sistema.

b) de um subsistema

Objetivos principais:

- Minimizar os custos de desenvolvimento, otimizando recursos e evitando retrabalho;
- Atingir a qualidade adequada com rapidez e eficiência;

- Atingir as versões úteis (alfa, beta e outros *releases* de teste) com rapidez e eficiência;
- Definir quais recursos de hardware e de software do sistema atual (caso já exista um) serão substituídos ou desativados para dar lugar ao novo subsistema;
- Desenvolver uma camada temporária de software que irá intermediar o convívio de parte do sistema atual (caso exista) com o subsistema a ser implantado, tentando manter o nível de integração entre eles (possivelmente, com tecnologias e conceitos diferentes) de forma a garantir, ao menos, o mínimo de satisfação dos requisitos (quanto a desempenho, atualização e consistência dos dados, funcionalidades, etc.);
- Concluir a análise, o *design*, o desenvolvimento e o teste de todas as funcionalidades necessárias;
- Desenvolver, de modo iterativo e incremental, um produto completo que esteja pronto para a transição dentro de uma comunidade de usuários. Isso implica descrever os casos de uso restantes e outros requisitos, incrementar o *design*, concluir a implementação e testar o software;
- Decidir se o software, o hardware, os usuários, etc. estão prontos para que o aplicativo seja implantado;
- Atingir um paralelismo proveitoso entre o trabalho das equipes de desenvolvimento (pela identificação de componentes que podem ser desenvolvidos de forma independente, o que possibilita o paralelismo natural entre equipes e acelera as atividades de desenvolvimento, mas acarreta o aumento da complexidade do gerenciamento de recursos e da sincronização dos fluxos de trabalho).

3.2.4. Transição

a) do sistema

Objetivos principais:

- Seqüencialmente, realizar a transição de cada um dos subsistemas que comporão o sistema;
- Consolidar todos os artefatos provenientes da concepção inicial do sistema e de seus subsistemas, formando a sua *baseline* completa;
- Promover o treinamento de usuários e equipe de manutenção (com uma visão do produto final);
- Efetuar atividades de ajuste, como correção de erros, melhoria no desempenho e na usabilidade;
- Avaliar as *baselines* de implantação tendo como base a visão completa do sistema e os critérios de aceitação gerais para o produto;
- Promover a obtenção de auto-suportabilidade do usuário;
- Obter o consentimento dos envolvidos de que as *baselines* de implantação estão completas e são consistentes com os critérios de avaliação da visão completa do sistema.

b) de um subsistema

Objetivos principais:

- Aplicar teste beta para validar o novo subsistema de acordo com as expectativas do usuário;
- aplicar teste beta e operação paralela relativa a um sistema legado que está sendo substituído;
- Efetuar a migração de software existente e conversão de bancos de dados operacionais;
- Promover o treinamento dos usuários e da equipe de manutenção;
- Realizar engenharia voltada para implantação, como preparação, empacotamento e treinamento de pessoal em campo;
- Efetuar atividades de ajuste, como correção de erros, melhoria no desempenho e na usabilidade;
- Avaliar as *baselines* de implantação tendo como base a visão completa e os critérios de aceitação para o produto;
- Promover a obtenção da auto-suportabilidade do usuário;
- Obter o consentimento dos envolvidos de que as *baselines* de implantação estão completas e são consistentes com os critérios de avaliação da visão do subsistema.

3.3. Disciplinas e artefatos

Assim como as fases, as disciplinas e os artefatos são os mesmos do RUP. Não houve necessidade de adaptação, com exceção de que se deve considerar, dependendo da iteração, que o enfoque pode ser o sistema como um todo (primeira e última iterações) ou um de seus subsistemas (as demais iterações).

4. Algumas considerações sobre o método

A idéia principal do modelo proposto na seção 3 é resultado de um estudo feito após o desenvolvimento de um sistema real de grande porte (que durou quatro anos e meio) para a gestão de uma operadora nacional de planos de saúde. Embora o RUP não tenha sido usado durante a concepção e a implementação de tal sistema, foi feito, posteriormente, um estudo sobre como o RUP se relacionava com os passos que tinham sido seguidos, chegando-se ao modelo proposto. Portanto, de certa forma, pode-se considerar que a essência do modelo proposto já foi aplicada pelo menos uma vez na prática, e os resultados foram realmente satisfatórios, visto que, um ano após o seu início, o primeiro e, certamente, um dos mais importantes subsistemas foi colocado em produção com sucesso, e, três anos depois, o sistema completo estava em produção, atendendo a todo o estado do Ceará.

Pode-se dizer ainda que o modelo proposto, embora em um nível de abstração superior, assemelha-se ao padrão de projeto estrutural *Composite*, que compõe objetos em estruturas de árvore para representarem hierarquias partes-todo, permitindo que se ignore a diferença entre composições de objetos e objetos individuais, tratando-os de maneira uniforme [7]. Neste caso, fica evidente essa correlação, visto que os subsistemas (as partes) são submetidos às mesmas fases do processo RUP a que seria submetido o sistema (todo), sem usar o método.

Em média, um projeto de dois anos de duração, seguindo estritamente o processo RUP, só começa a fase da implantação quando faltam aproximadamente dois meses e meio para o término desse prazo [4]. Entretanto, no método proposto neste trabalho, espera-se que, bem antes disso, parte do sistema já esteja em produção.

Uma outra proposta para o desenvolvimento de sistemas de grande porte, também baseada no RUP, utiliza o modelo de sistemas de sistemas interconectados [8]. Ela também utiliza o conceito de subsistemas, só que o enfoque dado difere em três pontos: existe o conceito de sistema "superordenado" para controlar a construção de todos os sistemas "subordinados" (os subsistemas), que serão desenvolvidos paralelamente, contendo todas as interfaces entre eles, previamente definidas. São necessárias, portanto, várias equipes de desenvolvimento e uma equipe extra para gerenciar o sistema "superordenado", contendo o seu próprio conjunto de artefatos. Um

caso de uso no sistema "superordenado" é quebrado freqüentemente em partes que se tornam casos de uso em seus sistemas "subordinados". Do ponto de vista de cada sistema "subordinado", os demais são vistos como atores no seu diagrama de casos de uso. Como a própria autora da proposta diz, ao lidar com sistemas ou equipes de desenvolvimento fisicamente separados, corre-se o risco de diminuir o reuso. Para resolver esse problema, é necessário incluir um grupo de pessoas para supervisionar todo o esforço de desenvolvimento. Acrescentaríamos a isso, os riscos adicionais em virtude da possibilidade de mudanças nas interfaces criadas exatamente no início do desenvolvimento, e o esforço de gerenciamento em virtude dos muitos recursos envolvidos, inclusive, com a missão de propagar o conhecimento específico de cada equipe sobre a aplicação entre as demais, ou seja, a questão do aprendizado e evolução progressivos a cada iteração estaria dependendo desse desempenho.

No nosso caso, com uma única equipe, muitos desses problemas não ocorreriam. Por exemplo, a questão das interfaces não causaria tantos riscos, já que a fronteira do sistema e de seus módulos é definida previamente. Ao finalizar o desenvolvimento de todos os subsistemas, a última iteração de transição apenas consolidaria todo o trabalho das transições parciais já efetuadas, e teríamos então um produto acabado, sujeito a futuras evoluções conforme as necessidades dos seus clientes ao longo do tempo.

5. Conclusões

Usando-se como base a abordagem iterativa e incremental do RUP, promoveu-se um reuso do próprio processo. Pode-se dizer ainda que a simplicidade do método tem reflexo também nos custos, já que praticamente uma mesma equipe participa do desenvolvimento de todos os subsistemas. O processo proposto se sobressai em relação ao processo original do RUP principalmente quando se trata de um ambiente de muitas incertezas, em que resultados práticos (após implantação), e não somente experimentais, em períodos mais curtos, sejam mais eficientes para amenizar essas incertezas.

Como possíveis trabalhos futuros, poder-se-ia, com a aplicação, na prática, do método e, portanto, de posse de um estudo de caso, avaliar e apresentar os resultados obtidos, investigar a necessidade de novos artefatos e/ou de adaptações nos existentes, tornar mais preciso o gráfico de fases e disciplinas do método e produzir um quadro comparativo entre o RUP, o método proposto e/ou outros trabalhos.

Referências

[1] Krutchen, Philippe. *A Rational Development Process*, White Paper, Crosstalk, 1996;

[2] Fowler, Martin; Scott, Kendall. *UML Essencial, um breve guia para a linguagem-padrão de modelagem de objetos*, 2ª. Edição, Bookman, 2000;

[3] ISO/IEC 9126-1. *Software engineering – Product quality* – part 1: Quality model, First Edition, 2001;

[4] Krutchen, Philippe. *The Rational Unified Process, An Introduction*, Second Edition, Addison Wesley, 2000;

[5] Bezerra, Eduardo. *Princípios de Análise e Projeto de Sistemas com UML*, Campus, 2002;

[6] Mendes, Antônio. *Arquitetura de Software, Desenvolvimento orientado para arquitetura*, Campus, 2002;

[7] Gamma, Erich *et al. Padrões de Projeto, Soluções Reutilizáveis de Software Orientado a Objetos*, Bookman, 2000;

[8] Ericsson, Maria. *Developing Large-scale Systems with the Rational Unified Process*, Rational Software White Paper, 2000.