RUP como Metodologia de Desenvolvimento de Software para Obtenção da Qualidade de Software

Alfredo Nazareno P. Boente	Fabiano S. G. de Oliveira	João Carlos Nogueira Alves
Centro Universitário Estadual da Zona Oeste UEZO/CCMAT	Instituto Superior de Tecnologia em Ciência da Computação de Petrópolis – ISTCC-PO	Centro Universitário Serra dos Órgãos UNIFESO
boente.uezo@faetec.rj.gov.br	fgomes@lncc.br	joão_carlos@superig.com.br

Abstract. *The center of interest of our discussion in ensuring the development process appropriate for obtaining quality of software,, examining the method RUP (Rational Unified Process), as an alternative to other solutions that have been submitted for this area. In this paper, we present a conceptual discussion on quality, initially. We discuss the main and relevant aspects of the method RUP, defending it, together with the Unified Modeling Language (UML), for use in the construction of software.*

Resumo. *Focaremos a nossa discussão na garantia do processo de desenvolvimento adequado, para obtenção de qualidade de software, analisando o método RUP (Rational Unified Process), como alternativa a outras soluções que têm sido apresentadas para esta área. Para tanto, neste artigo, apresentamos uma discussão conceitual sobre qualidade, inicialmente. Discutimos os principais e relevantes aspectos do método RUP, defendo-o, em conjunto com a Unified Modeling Language (UML), para uso na construção de softwares.*

1. Introdução

Com o advento da globalização e com os avanços da tecnologia de informação, cada vez mais as empresas se tornam dependentes de sistemas de informação que atendam, de forma eficaz e veloz, às suas necessidades. A economia digital abarca a economia baseada em tecnologias digitais, inclusive redes de comunicação digital (*internet*, *intranet* e VANs, ou redes privadas de valor agregado), computadores, *software* e outras tecnologias da informação relacionadas [Turban, McLean e Wethere 2004].

Todo esse cenário de transformação e adequação à modernidade, nos faz lembrar que, apesar de estarmos no século XXI, ainda existem inúmeros projetos de desenvolvimento de software que são iniciados e não são terminados, pior ainda, outros são terminados consumindo prazos e orçamentos bem acima do que foi estipulado no início do projeto, além de serem considerados produtos de baixíssima qualidade.

De acordo com Kruchten (2003), um produto de qualidade deve ter ausência de defeitos e, principalmente, deve atender aos propósitos desejados.

2. Qualidade de Software

Qualidade hoje em dia, não é apenas um diferencial de mercado para a empresa conseguir vender e lucrar mais, é um pré-requisito que a empresa deve conquistar para conseguir colocar o produto no Mercado Global.

A qualidade de software não pode ser avaliada isoladamente. No desenvolvimento de software, um método pobre ou a ausência de uma metodologia pode ser a causa da baixa qualidade. A avaliação da qualidade está diretamente relacionada com a qualidade de processos e metodologias utilizadas no desenvolvimento do software.

O conceito de qualidade de software pode ser abordado a partir de diversas definições existentes. Pressman (2002) afirma que "a qualidade de software é uma combinação complexa de fatores que variarão de acordo com diferentes aplicações e clientes que a solicitam". A qualidade de software é multidimensional, sendo assim, os requisitos de qualidade entram em conflito e os benefícios em atingi-los não são fáceis de medir [Rocha, Maldonado e Weber 2001]. Numa visão geral, a qualidade deve estar em conformidade com os requisitos dos clientes. Vamos mais além, qualidade é antecipar, satisfazer ou superar as expectativas dos clientes. A partir de uma visão mais tecnicista, podemos afirmar que a qualidade é obtida quando escrevemos tudo aquilo o que deve ser feito num projeto de software e, efetivamente o fazemos.

Qualidade de software é a conformidade de requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais [Pressman 2002].

Segundo a atual norma brasileira que trata sobre o assunto (NBR ISO 8402), qualidade é "a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas". Entende-se por entidade, o produto ou prestação de serviço do qual estamos nos referindo. As necessidades explícitas são as próprias condições e objetivos propostos pelo produtor ou por quem encomenda. As necessidades implícitas incluem as diferenças entre os diversos usuários, a evolução no tempo, as implicações éticas, as questões de segurança e outras visões subjetivas.

Consideramos as seguintes proposições da qualidade de software:

- Qualidade é o sucesso para o negócio de software, como em qualquer outro;
- A maneira mais barata de aumentar a produtividade de software é aumentando sua qualidade;
- A qualidade ao suporte do produto é tão importante quanto a qualidade do próprio software, o ambiente de suporte deve ter engenharia tanto quanto o ambiente de desenvolvimento;
- Para alcançar a qualidade de software, as pessoas e a cultura são tão importantes quanto à tecnologia;

- O único caminho seguro para aumentar a qualidade do software, é melhorar os processos;
- Aumento de processos é normalmente desnecessário a menos que o gerente demonstre compromisso e liderança;
- Qualidade e melhoramento dos processos não é nada fácil de ser implementado: sempre é possível realizar algo um pouco melhor, um pouco mais barato e um pouco mais rápido;
- Um sistema de qualidade compatível com ISO9000 é um bom parâmetro para a maioria das organizações, porém não para todas;
- Um sistema de qualidade para uma organização deve ser medido de acordo com suas necessidades e circunstâncias ou não será eficiente;
- Um sistema de qualidade de software eficiente utiliza de boas práticas da engenharia de software baseado nos seguintes princípios: prevenir defeitos ao invés de consertá-los; ter certeza dos defeitos que forem encontrados, serem corrigidos rapidamente; estabelecer e eliminar as causas, bem como os sintomas dos defeitos e auditar o trabalho de acordo com padrões e procedimentos previamente estabelecidos.

De acordo com Rocha, Maldonado e Weber (2001) "o Subcomitê de Software (SC7) do Comitê Técnico Conjunto (JTC1) da ISO e IEC vem trabalhando na elaboração de normas e relatórios técnicos que permitem especificar e avaliar a qualidade dos produtos de software". Existem documentos que registram tais normas e relatórios, classificados em três grupos:

a) Qualidade de produto de software
   i. ISO/IEC - 9126-1: Modelo de Qualidade (ISO/IEC, 1999a)
   ii. ISO/IEC- 9126-2: Métrica Externa (ISO/IEC, 1999b)
   iii. ISO/IEC- 9126-3: Métrica Interna (ISO/IEC, 1999c)
   iv. ISO/IEC- 9126-4: Métrica da Qualidade em Uso (ISO/IEC, 1999d)

b) Avaliação de produtos de software
   i. ISO/IEC- 14598-1: Visão Geral (ISO/IEC, 1999e)
   ii. ISO/IEC- 14598-2: Planejamento e Gestão (ISO/IEC, 1999f)
   iii. ISO/IEC- 14598-3: Processo para Desenvolvedores (ISO/IEC, 1999g)
   iv. ISO/IEC- 14598-4: Processo para Adquirentes (ISO/IEC, 1999h)
   v. ISO/IEC- 14598-5: Processo para Avaliadores (ISO/IEC, 1998)
   vi. ISO/IEC- 14598-6: Documentação de Módulos de Avaliação (ISO/IEC, 1999i)

c) Teste e requisitos da qualidade em pacotes de software

i. NBR ISO/IEC- 12119: Pacotes de Software - Teste e Requisitos de Qualidade (NBR ISO/IEC, 1998)

A figura 1 [Rocha, Maldonado e Weber 2001], obtida da ISO/IEC 9126-1, mostra a relação existente entre os documentos da série 9126 e 14598, deixando clara a abrangência da norma 14598-1 sobre todo o processo de avaliação, bem como a necessidade de uso da norma ISO/IEC 9126-1 como referência na aplicação das métricas. As empresas estão preocupadas em saber o que significa cada uma dessas normas para melhor utilizá-las ao confeccionar produtos de software.

Segundo o Ministério da Ciência e Tecnologia (2008), em pesquisa realizada em organizações, cerca de 65% das empresas conheciam as normas ISO/IEC 2196 ou ISO/IEC 12119, e um pouco menos (60%), a ISO/IEC 14598 (vide tabela 1 e gráfico 1).

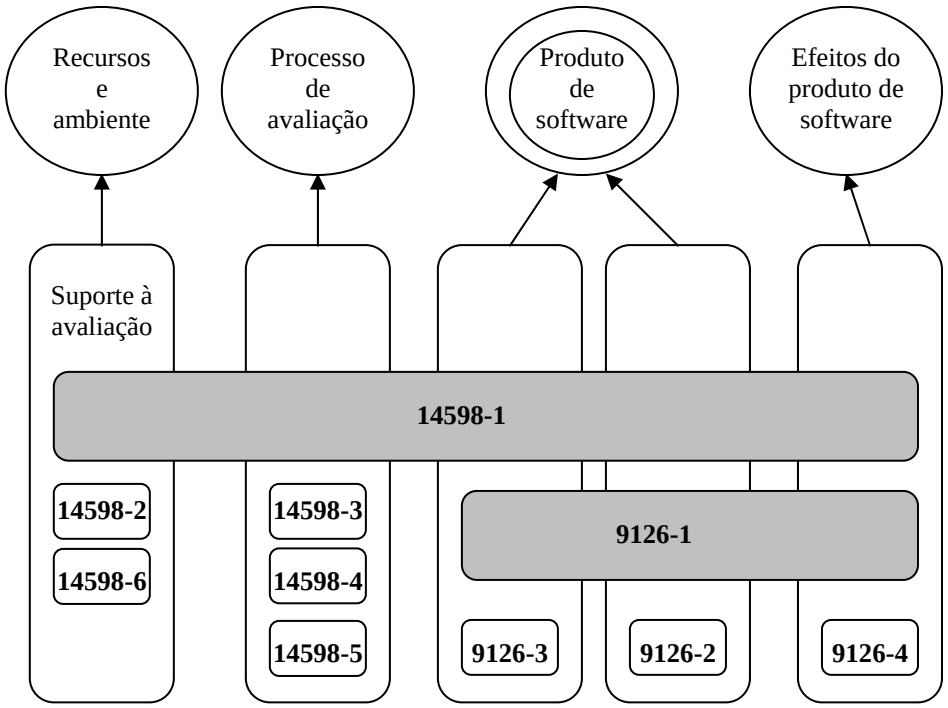

Figura 1. Relacionamento entre as normas das séries ISO/IEC 9126 e 14598

O modelo de qualidade para características externas e internas classifica os atributos da qualidade de software em seis características: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade [Rocha, Maldonado e Weber 2001].

Entende-se como funcionalidade a maneira que as funções e propriedades específicas do produto satisfazem as necessidades do cliente. A confiabilidade é o modo como o produto de software é capaz de manter seu nível de desempenho, ao longo do tempo, nas condições estabelecidas. A usabilidade trata do esforço necessário para utilização do sistema, com base em um conjunto de implicações e condições expostas pelo cliente. Eficiência é como os recursos e tempos envolvidos são compatíveis com o nível de desempenho requerido pelo software. A usabilidade entende-se que está

referida ao esforço necessário para realização de alterações específicas referentes ao produto de software e, a facilidade do produto poder ser transferido de um ambiente para outro, é o que entendemos por portabilidade.

Tabela 1. Conhecimento de normas da qualidade de produtos

Categorias	NBR 13596 (ISO/IEC 9126)		ISO/IEC 14598		ISO/IEC 12119	
	Nº	%	Nº	%	Nº	%
Conhece e usa sistematicamente	16	3,9	5	1,2	10	2,4
Conhece e começa a usar	31	7,5	14	3,4	22	5,4
Conhece, mas não usa	224	54,4	228	55,6	230	56,0
Não conhece	141	34,2	163	39,8	149	36,3
Base	412	100	410	100	411	100

No modelo de qualidade de software em uso, os atributos são classificados em quatro características: efetividade, produtividade, segurança e satisfação [Rocha, Maldonado e Weber 2001].

Entende-se por efetividade a capacidade do produto de software permitir que seus usuários atinjam metas especificadas com acurácia e completitude, em um contexto de uso especificado. A produtividade é a capacidade do produto de software permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso específico. Segurança é a capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedade ou ambiente, em um contexto de uso especificado. Satisfação entende-se pela capacidade do produto de software satisfazer seus usuários, em um contexto específico.

O processo de avaliação dos produtos de software encontra-se definida na série de normas ISO/IEC 14598, que deve ser utilizada em conjunto com a série ISO/IEC 9126 [Rocha, Maldonado e Weber 2001]. A tabela 2 [Ministério da Ciência e Tecnologia 2008] mostra o resultado obtido pela pesquisa realizada com organizações sobre as práticas da engenharia de software adotadas na avaliação da qualidade do produto de software.

Para haver uma produtividade de software qualitativa é necessário que efetivamente se garanta a qualidade do software [Kruchten 2003].

A garantia da qualidade é um conjunto de atividades planejadas e sistemáticas, implementadas com base no sistema de qualidade da organização, a fim de prover confiança de que o projeto irá satisfazer padrões relevantes de qualidade [Square 2000].

De acordo com Rocha, Maldonado e Weber (2001) “o processo de garantia da qualidade da norma ISO/IEC 12207 serve para garantir que os processos e produtos de software, no ciclo de vida do projeto, estejam em conformidade com os requisitos especificados e referentes aos planos estabelecidos.

A garantia da qualidade consiste nas funções gerenciais de auditar e relatar. A meta é fornecer a gerencia os dados necessários para que fique informada sobre a qualidade do produto, ganhando assim compreensão e confiança de que a qualidade do produto está satisfazendo suas metas [Pressman 2002].

Entende-se que garantia da qualidade é assegurar que tudo aquilo que estava escrito no projeto de software seja efetivamente feito. Por esse motivo é que um grupo de SQA (*Software Quality Assurance*) é montado com missão de ajudar a equipe de software a conseguir um produto final de alta qualidade baseando-se em um conjunto de atividades.

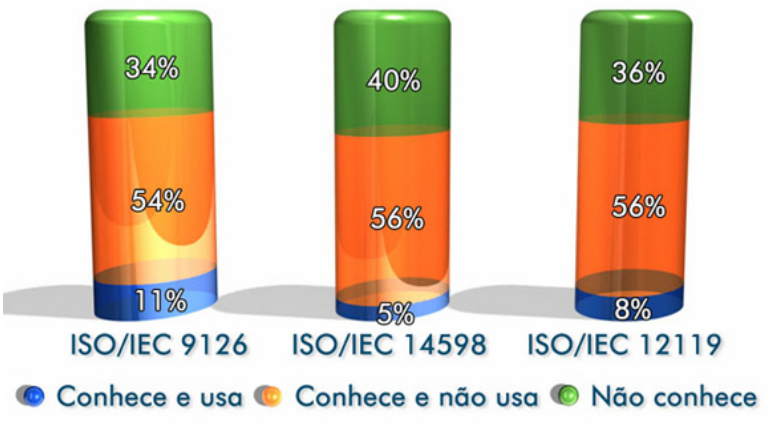

Gráfico 1. Conhecimento de normas da qualidade de produtos

Dessa forma, consegue-se manter um bom gerenciamento das atividades que garantem e asseguram a qualidade de software no produto que está sendo construído. Assim, é importante avaliar, de tempo em tempo, o desempenho global do projeto de software buscando sempre assegurar à satisfação dos padrões relevantes a qualidade do projeto [Boente 2003].

Deve-se definir um processo de garantia da qualidade adequado a cada projeto de desenvolvimento, determinando os objetivos do processo de maneira a assegurar que os produtos de software e os processos utilizados em sua construção estejam em conformidade com os requisitos e planos estabelecidos [Rocha, Maldonado e Weber 2001]. Segundo Square (2000) “a garantia da qualidade está dividida em duas partes: garantia do processo e garantia do produto”.

A garantia do processo implica em garantir que os processos do ciclo de vida do software utilizados no projeto estejam de acordo com o contrato e com os planos; garantir que práticas de engenharia de software estejam de acordo com o contrato, com as negociações e com os planos; garantir, no caso de subcontratação, que os requisitos aplicáveis sejam passados aos subcontratados e que seus produtos de software satisfaçam os requisitos do contrato original; garantir que o adquirente e outras partes envolvidas tenham o apoio e a cooperação necessários; garantir que as medições do produto e do processo de software estejam de acordo com os padrões e procedimentos estabelecidos; garantia que a equipe tenha a qualificação e o conhecimento necessários para o projeto e receba todo o treinamento necessário [Rocha, Maldonado e Weber 2001].

Tabela 2. Práticas de engenharia de software adotadas na avaliação da qualidade do produto

Categorias	Nº de organizações	%
Auditorias	97	22,6
Inspeção formal, Revisão por pares (Peer-review), Walthrough estruturado	70	16,3
Julgamento de especialistas	88	20,5
Levantamento de requisitos de qualidade	78	18,1
Medições da qualidade (Métricas)	75	17,4
Modelos de confiabilidade de software	21	4,9
Prova formal de programas	82	19,1
Segurança do produto final	58	13,5
Testes baseados em erros	236	54,9
Testes de aceitação	246	57,2
Testes de campo	243	56,5
Testes de integração	232	54,0
Testes de unidade	149	34,7
Testes do sistema integrado	222	51,6
Testes estruturais	107	24,9
Testes funcionais	255	59,3
Testes orientados a objetos	91	21,2
Testes para web	135	31,4
Outras	3	0,7
Não adota tais práticas	50	11,6
Base	430	100

Ainda, Rocha, Maldonado e Weber (2001) afirmam que a garantia do produto, implica em padrões de qualidade, metodologia, procedimentos e ferramentas para a execução das atividades de garantia da qualidade; procedimentos para a revisão do

contrato e sua coordenação; procedimentos para identificação, coleta, arquivamento, manutenção e disponibilização dos requisitos de qualidade; recursos, cronogramas e responsabilidades para conduzir as atividades de garantia da qualidade; atividades e tarefas selecionadas dos processos de apoio.

Portanto, pode-se afirmar que a garantia da qualidade do software está diretamente relacionada com a garantia do processo e com a garantia do produto de software a ser confeccionado e, esta garantia envolve aplicação de métodos técnicos, realização de revisões técnicas formais, atividades de teste de software, aplicação de padrões, controle de mudanças, métrica de software e manutenibilidade do produto. Focaremos a nossa discussão na garantia do processo de desenvolvimento, analisando o método RUP, como alternativa a outras soluções que têm sido apresentadas para esta área.

3. Conhecendo o Rational Unified Process

3.1. Visão do RUP

De acordo com Kroll e Kruchten (2003), pode-se ter três definições para o Rational Unified Process:

a) O RUP é uma maneira de desenvolvimento de software que é iterativa, centrada à arquitetura e guiada por casos de uso.

b) O RUP é um processo de engenharia de software bem definido e bem estruturado. Ele define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las. O RUP também provê uma estrutura bem definida para o ciclo de vida de um projeto, articulando claramente os marcos essenciais e pontos de decisão;

c) O RUP é também um produto de processo que oferece uma estrutura de processo customizável para a engenharia de software.

O RUP utiliza a linguagem de modelagem unificada para especificar, modelar e documentar artefatos. Por ser flexível e configurável, ele pode ser utilizado em projetos de pequeno, médio e grande porte.

3.2. Princípios do RUP

Não existe uma fórmula para aplicação do RUP devido ao fato dele poder ser aplicado de várias formas diferentes para cada projeto e organização a ser apresentados. De acordo com Martins (2007), existem alguns princípios que podem caracterizar e diferenciar o RUP de outros métodos iterativos:

a) Atacar os riscos antecipadamente e continuamente;

b) Certificar-se de entregar algo de valor ao cliente;

c) Focar no software executável;

d) Acomodar mudanças antecipadas;

e) Liberar um executável da arquitetura antecipadamente;

f) Construir o sistema com componentes;

g) Trabalhar junto como uma equipe;

h) Fazer da qualidade um estilo de vida, não algo para depois.

Os princípios do RUP acompanha as premissas referentes a garantia da qualidade do processo visando alcançar a garantia da qualidade do produto de software a ser desenvolvido [Kruchten 2003].

3.3. Elementos do RUP

Segundo Kroll e Kruchten (2003), o RUP possui cinco elementos principais: papéis, atividades, artefatos, fluxos de trabalho e disciplinas. Um papel (ou perfil) define o comportamento e as responsabilidades de um determinado indivíduo ou grupo de indivíduos trabalhando como uma equipe. Uma atividade é uma unidade de trabalho que um indivíduo executa quando está exercendo um determinado papel e produz um resultado importante para o contexto do projeto. Um artefato é um pedaço de informação que é produzido, modificado ou utilizado em um processo. Eles então são os produtos de um projeto.

Entende-se que papéis caracterizam os perfis dos profissionais envolvidos no projeto, como por exemplo, analista de sistemas, projetista etc. Quanto à atividade, ele (método) pode ser dividido nos seguintes passos: planejar uma iteração, encontrar casos de uso e atores, rever o projeto e executar testes de performances. Os artefatos podem ter várias formas como um modelo de caso de uso, um modelo de projeto, uma classe, um caso de negócio, código-fonte etc.

A enumeração de atividades, papéis e artefatos não constituem um processo. É necessário saber a seqüência do desenvolvimento das atividades para que possam ser produzidos artefatos de valor para o projeto. Um fluxo de trabalho é uma seqüência de atividades que são executadas para a produção de um resultado valioso para o projeto [Kroll e Kruchten 2003]. Os fluxos de trabalho podem ser representados por diagramas de seqüência, diagramas de colaboração e diagramas de atividades da linguagem de modelagem unificada.

O RUP utiliza três tipos de fluxos de trabalho:

a) Fluxos de trabalho principais, associados com cada disciplina;

b) Fluxos de trabalho de detalhe, para detalhar cada fluxo de trabalho principal;

c) Planos de iteração, que mostram como a iteração deverá ser executada.

Segundo Martins (2007) uma disciplina é uma coleção de atividades relacionadas que fazem parte de um contexto comum em um projeto. As disciplinas proporcionam um melhor entendimento do projeto sob o ponto de vista tradicional de um processo cascata. A separação das atividades em disciplinas torna a compreensão das atividades mais fácil, porém dificulta mais o planejamento das atividades.

O RUP possui nove disciplinas, divididas em disciplinas do processo e de suporte. As disciplinas de processo são: modelagem de negócios, requisitos, análise e projeto, implementação, teste e distribuição. As de suporte são: configuração e gerenciamento de mudanças, gerenciamento de projeto, e ambiente [Kruchten 2003].

Conforme mostra a figura 2 [Martins 2007], em sua arquitetura geral, o RUP possui duas dimensões:

O eixo horizontal representa o tempo e mostra os aspectos do ciclo de vida do processo à medida que se desenvolve. Este representa o aspecto dinâmico do processo e é expresso em termos de fases, disciplinas e marcos.

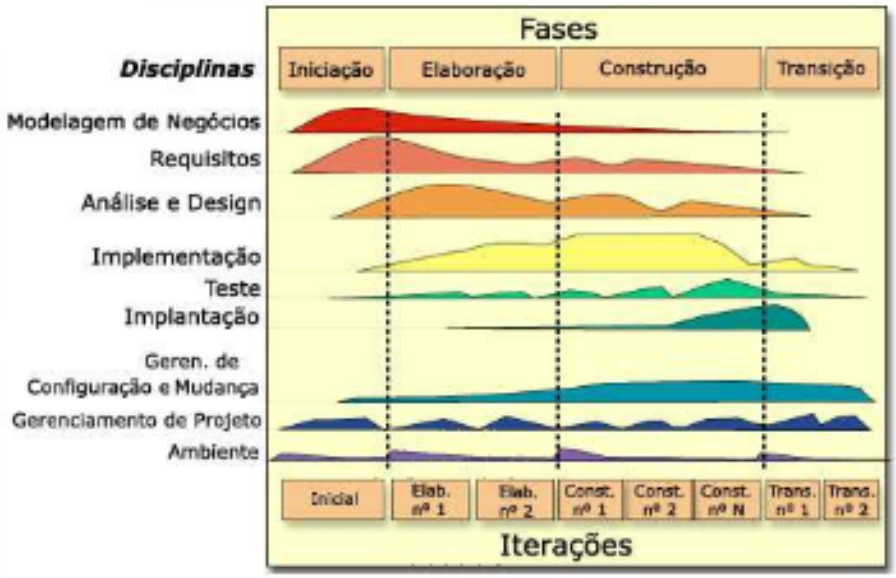

Figura 2. Arquitetura geral do RUP

O eixo vertical representa as disciplinas, que agrupam as atividades de maneira lógica, por natureza. Este representa o aspecto estático do processo e é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis do processo.

3.4. O Ciclo de Vida de um Projeto RUP

Kroll e Kruchten (2003) explicam que o ciclo de desenvolvimento no RUP possui quatro fases: iniciação, elaboração, construção e transição. Cada uma delas é concluída por um marco principal conforme mostra a figura 3 [Martins 2007].

Então, de acordo com a figura 3, constata-se que cada fase é basicamente um intervalo de tempo entre dois marcos principais.

Segundo Martins (2007), o ciclo de desenvolvimento termina com uma versão completa do produto de software. As fases definem estados do projeto, que são definidos por riscos que estão sendo mitigados ou questões que precisam ser respondidas.

A fase de iniciação, foca no tratamento de riscos relacionados com o caso de negócio específico. Como implementar RUP é um projeto que visa qualidade do software, acima de tudo, Boente (2003) afirma que “não existe projeto com risco zero”. Sendo assim, a tarefa de quantificação dos riscos de um projeto, passa a ser prioridade essencial, para a implementação de um projeto RUP, pois nos ajuda a verificar se o projeto é viável e se é financeiramente possível de ser implementado.

Na fase elaboração, o foco deve ser nos riscos técnicos e arquiteturais. O escopo deve ser revisado e os requisitos devem estar mais compreendidos. Durante a construção, a atenção será voltada para os riscos lógicos, e a maior parte do trabalho será realizada.

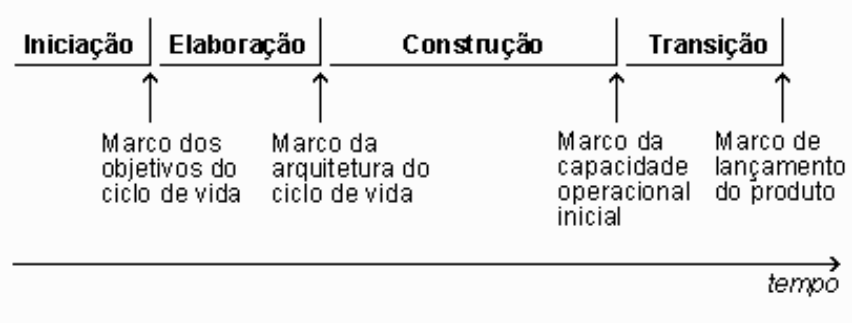

Figura 3. Ciclo de vida do RUP

Na fase de transição, serão tratados os riscos associados com a logística de distribuição do produto para a base de usuários. Portanto, constata-se que a devemos nos preocupar com a garantia da qualidade do projeto RUP, mesmo depois de seu término, de sua implementação.

Embora varie muito, de empresa para empresas e projetos diferentes, um ciclo de desenvolvimento para um projeto de tamanho médio, possui uma distribuição de esforço e programação como é apresentado na tabela 3 e na figura 4 [Martins 2007].

Tabela 3. Distribuição de esforço e programação em projetos de médio porte

	Iniciação	Elaboração	Construção	Transição
Esforço	~5%	20%	65%	10%
Programação	10%	30%	50%	10%

Conforme descrito na documentação do RUP, cada passagem pelas quatro fases gera uma geração do software. A menos que o produto "desapareça", ele irá se desenvolver na próxima geração, repetindo a mesma seqüência de fases de iniciação, elaboração, construção e transição [Kroll e Kruchten 2003]. Esses ciclos subseqüentes que são criados recebem o nome de ciclos de evolução**.** A cada ciclo, são produzidas novas gerações do software.

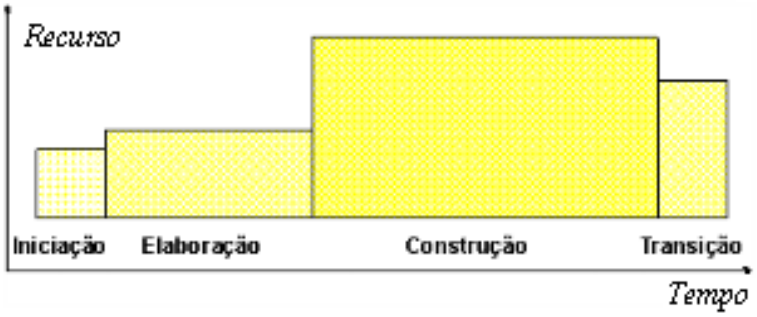

Figura 4. Distribuição de esforço e programação em projetos de médio porte

Os ciclos de evolução podem ser disparados por melhorias sugeridas pelos usuários, mudanças no contexto do usuário, mudanças na tecnologia subjacente, reação à concorrência e assim por diante [Martins 2007]. Em linhas gerais, a menos que ocorram mudanças significativas do produto ou da arquitetura, os ciclos de evolução têm fases de iniciação e elaboração bem menores, pois a definição e a arquitetura básicas do produto foram determinadas por ciclos de desenvolvimento anteriores [Kruchten 2003].

Logo, constata-se que o RUP prima pela qualidade do produto de software que vai ser gerado dando, inicialmente, maior importância a garantia da qualidade de processos e, consequentemente em seguida, a garantia da qualidade do produto de software gerado.

4. Considerações Finais

Com a utilização de uma metodologia de desenvolvimento de software como o RUP, é possível obter qualidade de software, produtividade no desenvolvimento, operação e manutenção de software, controle sobre o desenvolvimento dentro de custos, prazos e níveis de qualidade desejados, sem deixar de levar em conta a estimativa de prazos e custo com maior precisão.

Deve-se ter consciência que os benefícios não virão de maneira imediata. Portanto, é necessário adquirir treinamento adequado, adaptação da metodologia no contexto ao qual ela será utilizada, apoio especializado para as equipes de desenvolvimento e tempo para a absorção da metodologia.

Logo, recomendamos a implementação de projeto de RUP como metodologia de desenvolvimento de software para obtenção da qualidade do produto de software gerado.

Referências

Boente, A. N. P. (2003) “Gerenciamento & controle de projetos”. Rio de Janeiro: Axcel Books.

Martins, J. C. C. (2007) “Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML. 4 ed. Rio de Janeiro: Brasport.

Kroll, P. e Kruchten P. (2003) “The rational unified process made easy: a practitioner's guide to the RUP ”. Addison Wesley.

Kruchten, P. (2003) “The rational unified process: an introduction”. 3 ed. Addison Wesley.

Ministério da Ciência e Tecnologia (2008). “Qualidade e produtividade de software no setor de software”. Site acessado: http://ftp.mct.gov.br/temas/info/Dsi/Quali2001/QualiProutosSW2001.htm, em 09 de fevereiro de 2008, às 23:16h. Brasília: Secretaria de política de informática.

Pressman, R. S. (2002) “Engenharia de software”. 5 ed. São Paulo: McGraw Hill.

Rocha, A. R. C.; Maldonado, J. C.; Weber, K. C. (2001) “Qualidade de software: teoria e prática. São Paulo: Prentice Hall.

Square, N. (2000)“A guide to the project management body of knowledge”. PMI – Project Management Institute. Pennsylvania.

Turban, E.; McLean, E.; Wetherbe, J. (2004) “Tecnologia da informação para gestão: transformando os negócios na economia digital. 3 ed. Porto Alegre: Bookman.