FACULDADE DE TECNOLOGIA SENAC GOIÁS

GESTÃO DA TECNOLOGIA DA INFORMAÇÃO

RUP (RATIONAL PROCESS UNIFIED)

SENAC/ 2017

EZIO BENEDITO MENDONÇA MELO

FACULDADE DE TECNOLOGIA SENAC
GOIÁS

RUP (RATIONAL PROCESS UNIFIED)

EZIO BENEDITO MENDONÇA MELO

SENAC
JUNHO/2017

FACULDADE SENAC

RUP (RATIONAL PROCESS UNIFIED)

Atividade avaliativa, à Faculdade Senac de Tecnologia, como um dos requisitos para a conclusão de curso.

Disciplina: Auditoria e Qualidade de Software
Orientador (a): Prof.° Elias Batista Ferreira

SENAC

JUNHO/2017

INTRODUÇÃO

O RUP (Processo Unificado Racional) é um processo de desenvolvimento de software. Ele engloba as ações necessárias para transformar um conjunto de requisitos do cliente em um sistema de software.

O RUP (Rational Unified Process), é uma “versão” do UP (Unified Process), criada pela Rational, empresa que mais tarde foi comprada pela IBM.

O RUP é dirigido por casos de uso, que são utilizados para capturar os requisitos funcionais e definir as tarefas de cada iteração. Cada iteração lida com um cenário ou conjunto de casos de uso durante todo o tempo.

O RUP é centrado na arquitetura, o que significa que os projetos devem ter bases sólidas e estáveis, mas flexíveis o suficiente para possibilitar os incrementos em cada fase iterativa do processo. O fato de um modelo ser centrado na arquitetura sugere que o sistema será o mais modularizado e componentizado possível, pois isto facilita a inserção de novos módulos e componentes.

O RUP é focado no risco, pois procura mitigar e controlar os riscos mais críticos logo nas entregas da fase de Elaboração, no início do projeto.

O RUP também é baseado em componentes, logo objetos projetados utilizando a linguagem UML são interconectados para formar os artefatos do sistema.

1. O que é? E qual sua relação com o UP

O Processo unificado surgiu como uma nova proposta de desenvolvimento, fugindo do modelo em cascata, seguindo basicamente as mesmas etapas genéricas de desenvolvimento de software, porém visando um desenvolvimento iterativo e incremental, totalmente diferente do modelo em cascata.

O RUP nasce da captura das melhores práticas de desenvolvimento de software, visando dar resposta satisfatória aos diversos problemas inerentes a atividade. Segundo Grady Booch estas são algumas causas de problemas no decorrer de um projeto de software:

- Gerenciamento especial de requisitos
- Comunicação ambígua e imprecisa
- Arquiteturas frágeis
- Complexidade subjugada
- Inconsistências não detectadas em requisitos, construções e implementações
- Teste insuficiente
- Avaliação subjetiva de status do projeto

O RUP *(Rational Unified Process)* é um processo de desenvolvimento de *software*. Ele engloba as ações necessárias para transformar um conjunto de requisitos do cliente em um sistema de *software* combinando os ciclos de vida iterativo, incremental e verificando a qualidade do mesmo de forma que cada entrega do *software* em um ciclo agrega mais valor ao produto em relação ao ciclo anterior.

O principal objetivo do RUP é atender as necessidades dos usuários garantindo uma produção de software de alta qualidade que cumpra um cronograma e um orçamento previsível. Assim, o RUP mostra como o sistema será construído na fase de implementação, gerando o modelo do projeto e, opcionalmente, o modelo de análise que é utilizado para garantir a robustez. O RUP define perfeitamente quem é responsável pelo que, como as coisas deverão ser feitas e quando devem ser

realizadas, descrevendo todas as metas de desenvolvimento especificamente para que sejam alcançadas.

O RUP organiza o desenvolvimento de software em quatro fases, onde são tratadas questões sobre planejamento, levantamento de requisitos, análise, implementação, teste e implantação do software. Cada fase tem um papel fundamental para que o objetivo seja cumprido, distribuídos entre vários profissionais como o Analista de sistema, Projetista, Projetista de testes, entre outros.

2. Valores do RUP

RUP é baseado em um conjunto de princípios de desenvolvimento de *software* e melhores práticas, por exemplo:

1. Desenvolvimento de *software* iterativo
2. Gerenciamento de requisitos
3. Uso de arquitetura baseada em componente
4. Modelagem visual de *software*
5. Verificação da qualidade do *software*
6. Controle de alteração no *software*

3. Disciplinas do RUP

3.1. Disciplinas de Engenharia de *Software*

3.1.1. Modelagem de Negócios

As organizações estão cada vez mais dependentes de sistemas de TI, tornando-se imperativo que os engenheiros de sistemas de informação saibam como as aplicações em desenvolvimento se inserem na organização. As empresas investem em TI, quando entendem a vantagem competitiva do valor acrescentado pela tecnologia. O objetivo de modelagem de negócios é, primeiramente, estabelecer uma melhor compreensão e canal de comunicação entre engenharia de negócios e engenharia de *software*. Compreender o negócio significa que os engenheiros de *software* devem compreender a

estrutura e a dinâmica da empresa alvo (o cliente), os atuais problemas na organização e possíveis melhorias. Eles também devem garantir um entendimento comum da organização-alvo entre os clientes, usuários finais e desenvolvedores.

Modelagem de negócios, explica como descrever uma visão da organização na qual o sistema será implantado e como usar esta visão como uma base para descrever o processo, papéis e responsabilidades.

3.1.2. Requisitos

Esta disciplina explica como levantar pedidos das partes interessadas (*"stakeholders"*) e transformá-los em um conjunto de requisitos que visam estabelecer uma concordância com os clientes e outros envolvidos com sistema. Além disso, essa fase oferece aos desenvolvedores uma compreensão melhor dos requisitos, possibilitando a delimitação das "fronteiras do sistema", a definição de uma interface além do planejamento de custos relacionados ao projeto.

*3.1.3. Análise e Projeto (***Design***)*

A Análise e Projeto busca mostrar como o sistema será realizado. O objetivo é construir um sistema que:

- Execute, em um ambiente de execução determinado, as tarefas e funções especificadas nas descrições de casos de uso;
- Cumpra todas as suas necessidades;
- Seja fácil de manter quando ocorrerem mudanças nos requisitos funcionais;

Resultados de projeto em um modelo de análise e projeto têm, opcionalmente, um modelo de análise. O modelo de design serve como uma abstração do código-fonte, isto é, o projeto atua como uma espécie de "gabarito" de como o código-fonte será estruturado e escrito. O modelo de projeto consiste em

classes de design estruturado em pacotes e subsistemas com interfaces bem definidas, representando o que irá se tornar componentes da aplicação. Ele também contém descrições de como os objetos dessas classes colaboram para desempenhar casos de uso do projeto.

3.1.4. Implementação

Os efeitos da Implementação são:

- Definir a organização do código em termos de subsistemas de implementação organizados em camadas
- Implementar classes e objetos em termos de componentes (arquivos-fonte, binários e executáveis, entre outros)
- Testar, como unidades, os componentes desenvolvidos
- Integrar os resultados produzidos por implementadores individuais (ou equipes) em um sistema executável

Sistemas são criados através da aplicação de componentes. O processo descreve como reutilizar componentes existentes ou implementar novos componentes com responsabilidades bem definidas, tornando o sistema mais fácil de manter e aumentando as possibilidades de reutilização.

3.1.5. Teste

As finalidades da disciplina de Teste são:

- Verificar a interação entre objetos
- Verificar a integração adequada de todos os componentes do *software*
- Verificar se todos os requisitos foram corretamente implementados
- Identificar e garantir que os defeitos são abordados antes da implantação do *software*
- Garantir que todos os defeitos são corrigidos, reanalisados e fechados

O *Rational Unified Process* propõe uma abordagem iterativa, o que significa que, deve-se testar todo o projeto. Isto permite encontrar defeitos tão cedo

quanto possível, o que reduz radicalmente o custo para reparar o defeito. Os testes são realizados ao longo de quatro dimensões da qualidade: confiabilidade, funcionalidade, desempenho da aplicação e desempenho do sistema. Para cada uma destas dimensões da qualidade, o processo descreve como passar pelo teste do ciclo de planejamento, projeto, implementação, execução e avaliação.

3.1.6. Implantação

O objetivo da Implantação é o de produzir com sucesso lançamentos de produtos e entregar o *software* para seus usuários finais. Ela cobre uma vasta gama de atividades, incluindo a produção de *releases* externas do *software*, a embalagem do *software* e aplicativos de negócios, distribuição do *software*, instalação do *software* e s prestação de assistência aos usuários. Embora as atividades de implantação estejam principalmente centradas em torno da fase de transição, muitas delas devem ser incluídas nas fases anteriores para se preparar para a implantação, no final da fase de construção. Os processos ("*workflows*") de "Implantação e Ambiente" do RUP contêm menos detalhes do que outros *workflows*.

3.2. Disciplinas de Apoio/Suporte

3.2.1. Ambiente

O ambiente enfoca as atividades necessárias para configurar o processo para um projeto. Ele descreve as atividades necessárias para desenvolver as diretrizes de apoio a um projeto. A proposta das atividades de ambiente é prover à organização de desenvolvimento de *software* os processos e as ferramentas que darão suporte à equipe de desenvolvimento. Se os usuários do RUP não entendem que o RUP é um *framework* de processo, eles podem percebê-lo como um processo pesado e caro. No entanto, um conceito-chave dentro do RUP foi que o processo RUP pode e, muitas vezes, deve ser refinado. Este foi inicialmente feito manualmente, ou seja, por escrito, um documento de "caso de desenvolvimento" que especificou o processo refinado para ser

utilizado. Posteriormente, o produto IBM *Rational Method Composer* foi criado para ajudar a tornar esta etapa mais simples, engenheiros de processos e gerentes de projeto poderiam mais facilmente personalizar o RUP para suas necessidades de projeto. Muitas das variações posteriores do RUP, incluindo *OpenUP/Basic*, a versão open-source e leve do RUP, são agora apresentados como processos distintos, por direito próprio, e atendem a diferentes tipos e tamanhos de projetos, tendências e tecnologias de desenvolvimento de *software*. Historicamente, como o RUP, muitas vezes é personalizado para cada projeto por um perito do processo RUP, o sucesso total do projeto pode ser um pouco dependente da capacidade desta pessoa.

3.2.2. Configuração e Gerência de Mudança

A disciplina de Gestão de Mudança em negócios com RUP abrange três gerenciamentos específicos: de configuração, de solicitações de mudança, e de *status* e medição.

- Gerenciamento de configuração: A gestão de configuração é responsável pela estruturação sistemática dos produtos. Artefatos, como documentos e modelos, precisam estar sob controle de versão e essas alterações devem ser visíveis. Ele também mantém o controle de dependências entre artefatos para que todos os artigos relacionados sejam atualizados quando são feitas alterações.
- Gerenciamento de solicitações de mudança: Durante o processo de desenvolvimento de sistemas com muitos artefatos existem diversas versões. O CRM mantém o controle das propostas de mudança.
- Gerenciamento de status e medição: Os pedidos de mudança têm os estados: novo, conectado, aprovado, cedido e completo. A solicitação de mudança também tem atributos como a causa raiz, ou a natureza (como o defeito e valorização), prioridade, etc. Esses estados e atributos são armazenados no banco de dados para produzir relatórios úteis sobre o andamento do projeto. A *Rational* também tem um produto

para manter a solicitações de mudança chamado *ClearQuest*. Esta atividade tem procedimentos a serem seguidos.

3.2.3. Gerência de Projeto

O planejamento de projeto no RUP ocorre em dois níveis. Há uma baixa granularidade ou planos de Fase que descreve todo o projeto, e uma série de alta granularidade ou planos de Iteração que descrevem os passos iterativos. Esta disciplina concentra-se principalmente sobre os aspectos importantes de um processo de desenvolvimento iterativo: Gestão de riscos; Planejamento de um projeto iterativo através do ciclo de vida e para uma iteração particular; E o processo de acompanhamento de um projeto iterativo, métricas. No entanto, esta disciplina do RUP não tenta cobrir todos os aspectos do gerenciamento de projetos.

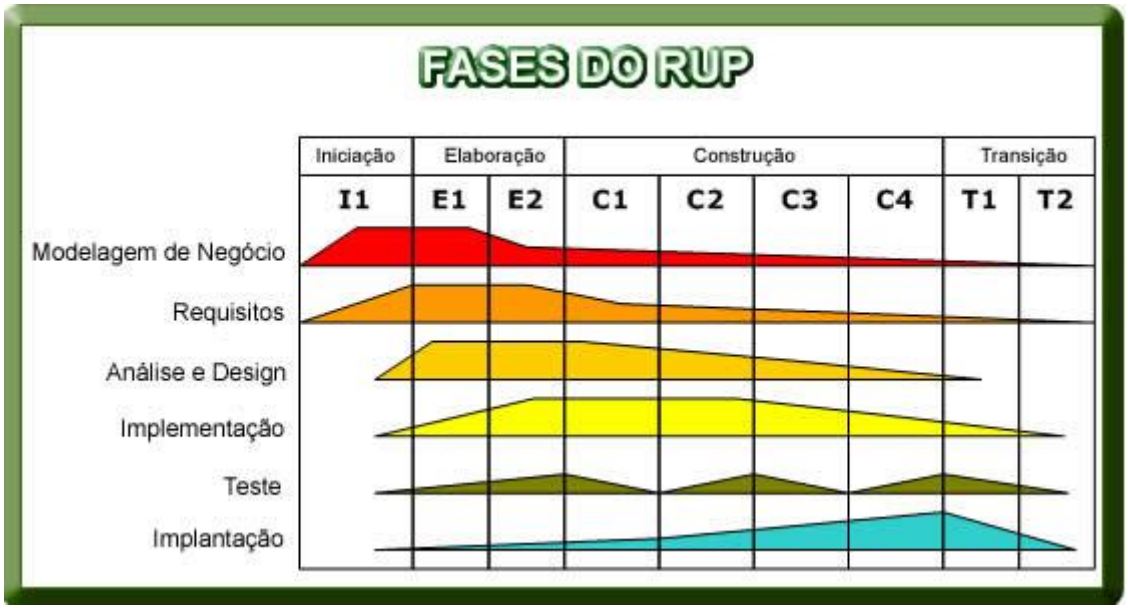
Por exemplo, não abrange questões como:

- Gestão de Pessoas: contratação, treinamento, etc
- Orçamento Geral: definição, alocação, etc
- Gestão de Contratos: com fornecedores, clientes, etc

Fonte: IBM Rational Unified Process

4. Fases/Iterações do RUP

Para exemplificar as fases/iterações do RUP, o gráfico conhecido como "gráfico de baleias" é utilizado. Ele traz um panorama de como um projeto é executado no RUP, levando em consideração as disciplinas, iterações e fases. Observando as curvas do gráfico, é possível notar que em cada fase são abordadas todas as disciplinas.



No RUP, o Engenheiro de Processo é responsável por adaptar o processo de desenvolvimento de *software* para atender o projeto. Para isso, este papel define o tamanho de cada iteração e quantas iterações existirão por cada fase. Nas matérias de GPP e MDS, a equipe de GPP de cada projeto ficou a cargo deste papel. Algumas equipes por exemplo, utilizaram iterações com extensão de duas semanas, com as fases de iniciação e elaboração sendo compostas apenas de uma iteração, enquanto a fase de construção constituída por mais iterações.

4.1. Iniciação

Esta fase do RUP abrange as tarefas de comunicação com o cliente e planejamento. É feito um plano de projeto avaliando os possíveis riscos, as estimativas de custo e prazos, estabelecendo as prioridades, levantamento dos requisitos do sistema e preliminarmente analisá-lo. Assim, haverá uma anuência das partes interessadas na definição do escopo do projeto, onde são examinados os objetivos para se decidir sobre a continuidade do desenvolvimento.

É uma fase muito importante para os esforços de desenvolvimento, é nela que se encontra grandes riscos de negócios e de requisitos que precisam de um cuidado especial a fim de que o projeto flua normalmente. A meta que almejada é que

todos os envolvidos no projeto do software cheguem a um consenso e tenham conhecimento claro sobre o ciclo de vida do mesmo.

Para projetos iniciais esta é uma parte fundamental que requer demasiada atenção. Para projetos que visam melhorias de sistemas esta fase já existe, contudo é uma fase de iniciação mais rápida.

É nesta fase que estabelecem o escopo do projeto de software, os critérios de aceitação e se o projeto deve ou não ser aceito. Os casos de uso começam a ser levantados, verificar quais são os principais cenários que o sistema encontrará. Na iniciação deve se a tentar também a estimar os riscos em potencial, que todo projeto possui. Também nesta mesma fase deve-se prepara o ambiente de suporte para o projeto como a configuração de ambiente para e equipe.

4.2. Elaboração

A fase de elaboração tem como meta criar a baseline de arquitetura do sistema, cujo objetivo é fornecer uma base estável para a construção. Primeiramente é realizado um exame dos requisitos mais significativos, ou seja, os que tem um maior impacto na arquitetura. Durante esta fase são realizados protótipos de arquitetura a fim de que seja verificada a estabilidade da arquitetura escolhida.

Deve se certificar nesta fase que a arquitetura que está sendo utilizada é estável o suficiente e que os riscos sejam minimizados, podendo assim determinar com uma segurança maior, o custo do sistema e quanto de programação será necessário. Além de se certificar da estabilidade da arquitetura é necessário demonstrar que a arquitetura escolhida para a baseline suportará os requisitos do sistema. O preço e o tempo de desenvolvimento devem ser levados em consideração na escolha da arquitetura.

4.3. Construção

Desenvolve ou Adquire os componentes de Software. O principal objetivo desta fase é a construção do sistema de software, com foco no desenvolvimento de

componentes e outros recursos do sistema. É na fase de Construção que a maior parte de codificação ocorre.

A fase de construção é literalmente de manufatura, em que o foco é no gerenciamento de recursos e controle de operação, visando a otimização de custos, programação e um produto de qualidade. Tem como meta concluir o desenvolvimento e esclarecer possíveis requisitos restantes das fases anteriores. Deixa-se o campo do desenvolvimento intelectual e parte para o ataque das áreas de desenvolvimento do produto, para que possa ser implementado durante a construção e transição.

Nesta fase a rapidez e eficiência são sempre metas a serem atingidas. Versões do software utilizáveis com a alfa, beta e outras são disponibilizadas, visando verificar se o software está pronto para ser implementado para o usuário. Deve haver a conclusão de todos os testes de funcionalidade da aplicação nesta fase.

4.4. Transição

A fase de transição do RUP é basicamente a fase em que é feita a entrega do produto ao cliente, ou seja, ela tem o objetivo de assegurar que o software produzido esteja disponível aos usuários finais.

Os objetivos principais desta fase são:

- Fazer testes beta para validar o novo sistema em confronto com as expectativas do usuário.
- Treinamento de usuários e equipe de manutenção.
- Receber um feedback do cliente acerca do produto, com o fim de serem feitos ajustes finos no produto, configuração, instalação e usabilidade.

5. Papéis

O RUP tem alguns papéis genéricos como os de analista, desenvolvedor, testador e gerente. Cada um desses papéis genéricos tem outros papéis mais detalhados,

somente o papel de testador que a única atividade existente é a de testador. Abaixo temos os papéis genéricos e os detalhados de cada um.

Analistas:

- Analista de sistema
- Designer de negócios
- Revisor do modelo de negócios
- Analista do processo de negócios
- Revisor de requisitos
- Especificar de requisitos
- Analista de teste
- Designer de interface de usuário

Desenvolvedores:

- Designer de cápsula
- Revisor de código
- Designer de banco de dados
- Implementador
- Integrador
- Arquiteto de *software*
- Revisor de arquitetura
- Revisor de *desgin*
- Designer
- Designer de teste

Testadores:

- Testador

Gerentes:

- Engenheiro de processo
- Gerente de projeto

- Gerente de controle de mudança
- Gerente de configuração
- Gerente de implantação
- Revisor do projeto
- Gerente de testes

CONCLUSÃO

O RUP tem como vantagem, ser um método evolucionário de desenvolvimento de software. O usuário não espera até a conclusão do projeto para ter contato com o software, devido ao modelo incremental. Após o termino do desenvolvimento é muito difícil encontrar novos erros.

Porem como desvantagem, podem ocorrer divergências entre a documentação e o software. Pode entrar em loop devido ao modelo iterativo e incremental, dependendo do cliente para chegar ao fim do projeto. Aumento de gastos devido à implantação da versão a cada incremento. Pode gerar uma grande mudança em partes já desenvolvidas para realizar algum novo requisito incremental.

O RUP prova ser um processo de desenvolvimento robusto e bem definido, embora bastante complexo/trabalhoso para projetos de software de pequeno porte, ele pode ser bem aproveitado para projetos aonde é preciso manter registro constante do fluxo do projeto.

Referências:

Fonte: http://www.funpar.ufpr.br:8080/rup/index.htm

Fonte: https://pt.wikipedia.org/wiki/IBM_Rational_Unified_Process

Fonte: https://www.ibm.com/software/rational

Fonte: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf