Capítulo

1

Processos Tradicionais de Desenvolvimento de Software

Wislayne Aires Moreira

Ao longo deste capítulo, serão apresentados os processos tradicionais de desenvolvimento software mais utilizados como: RUP, um dos processos mais difundidos; o OpenUp e o MSF; focando em suas origens, características, arquitetura, etapas, disciplinas e ciclo de vida.

# Introdução

Os processos[[1]](#footnote-1) tradicionais de desenvolvimento de software são processos onde se podem presumir todos os requisitos do sistema, que traz a vantagem de tornar os projetos completamente planejados, facilitando a gerência do mesmo, mantendo sempre uma linha, caracterizando o processo como bastante rigoroso [GALDINO 2008]. Os processos orientados a documentos são considerados rigorosos, devido à especificação de requisitos o que torna essa etapa importante para o cliente, onde as suas necessidades são documentadas e definidas. Os processos tradicionais são caracterizados por possuir abordagem voltada ao uso de documentação detalhada das atividades, fases sequenciais de processo e um conjunto de artefatos em cada fase, bons para atividades estáveis, mas caro para ambientes dinâmicos, exige pessoal experiente e o ambiente funciona na organização de responsabilidades.

Muitas pessoas acham que as metodologias tradicionais são processos que não têm viabilidade, e que substituí-las por metodologias ágeis é a única opção para se obter um produto de qualidade. Dessa forma, desenvolvem de qualquer maneira, acham que é um processo “oba-oba”, que vale qualquer coisa e saem desenvolvendo de qualquer forma. Por causa, dessa visão de que o processo pode ser desenvolvido de qualquer forma, é que há casos de fracasso. O que realmente importa não é a metodologia usada, mas sim como o processo está sendo desenvolvido por todos envolvidos.

Podemos citar alguns processos tradicionais existentes: RUP (Rational Unified Process), OpenUp (Open Unified Process), EUp (Enterprise Unified Process), MSF (Microsoft Solutions Frameworks), Catalisys, ICONIX e etc. Neste capítulo o foco será dado ao RUP, ...

# RUP

É um processo de engenharia de software que foi patenteado pela Rational Software Corporation, adquirida posteriormente pela IBM, fornecendo técnicas a serem seguidas pelos integrantes da equipe que desenvolve software com a finalidade de aumentar a produtividade da equipe.

O RUP surgiu em 1998, a partir de abordagens seguidas na Ericson, onde trabalhava Ivan Jacobson (1967), que mais tarde, se uniu a Grady Booch e James Rumbaugh, formando o grupo conhecido como “os três amigos”, os quais começaram as iniciativas para a unificação de suas metodologias, desenvolvidas desde 1981 na Rational. A evolução do RUP pode ser mostrada na Figura 1.1:

**Rational Approach**

**Objectory Process 3.8**

**1995**

**1997**

**Plug-ins-.NET,.J2EE**

**Web Based Development**

**Rational Unified Process 2002**

**Rational Unified Process 2000**

**Rational Unified Process 5.5**

**Rational Unified Process 5.0**

**Rational Objectory Process 4.1**

**Rational Objectory Process 4.0**

**Engenharia de negócio**

**Processos SQA + College Requirements**

**OMT+ Método Booch**

**UML 1.3**

**UML 1.2**

**UML 1.1**

**UML 1.3**

**2002**

**2000**

**1999**

**1998**

**1996**

Figura 1.1. História do RUP

* + 1. O RUP e suas características

O Rational Unified Process (também chamado de processo RUP) é um processo de desenvolvimento iterativo e incremental. Iterativo porque um grande projeto é dividido em projetos menores, que após termino tem-se um produto acabado, que serve para ser utilizado em outra iteração e é incremental por que em cada iteração e é incremental por que cada iteração é um incremento,que é o crescimento do produto. O RUP é definido por três elementos centrais que são [DANTAS 2003]:

* Uma abordagem de desenvolvimento de software iterativo, centrado em arquitetura e casos de uso baseados em risco;
* Um processo de engenharia de software bem definido e estruturado;
* Um produto do processo que fornece uma estrutura customizável do processo.

O RUP é um processo que se caracteriza pelos seguintes elementos: a UML é uma parte integrante do RUP; o RUP e UML foram desenvolvidos juntos; centrado em uma arquitetura, onde promove a definição inicial de uma arquitetura robusta, que facilita a paralelização do desenvolvimento, reutilização e a manutenção e guiados por caso de uso.

* + 1. Visão Geral do RUP

A arquitetura é dividida em duas estruturas,as quais refletem as duas visões em que um sistema pode ser descrito: componentes dinâmicos e componentes estáticos.

**Aspectos dinâmicos**



**Aspectos**

**estáticos**

Figura 1.2 Arquitetura RUP

A dimensão horizontal representa a estrutura dinâmica ou tempo dos processos. Os processos em termos de ciclo, fases (concepção, elaboração, construção e transição), iterações e milestones [DANTAS 2003 e RATIONAL SOFTWARE 2001].

O ciclo na dimensão horizontal da arquitetura representa a evolução do produto. Definem as fases que interligam o início do projeto ao seu fim. As iterações são as seqüências de atividades em um período de tempo definido dentro de um projeto, onde é desenvolvido uma versão estável de um produto e os milestones são os marcos cruciais, com objetivos definidos e alcançáveis, que podem ser analisados pelos clientes e desenvolvedores [MONTEIRO e RAMALHO].

A dimensão vertical representa a estrutura estática do processo. Descreve como elementos do processo (atividades, disciplinas, artefatos e papéis) são agrupados em disciplinas de processo ou workflows [DANTAS 2003]. Na estrutura estática um processo descreve “**quem**” está fazendo “**o quê**”, “**como**” e “**quando**”, conforme demonstrado na Figura 1.3:

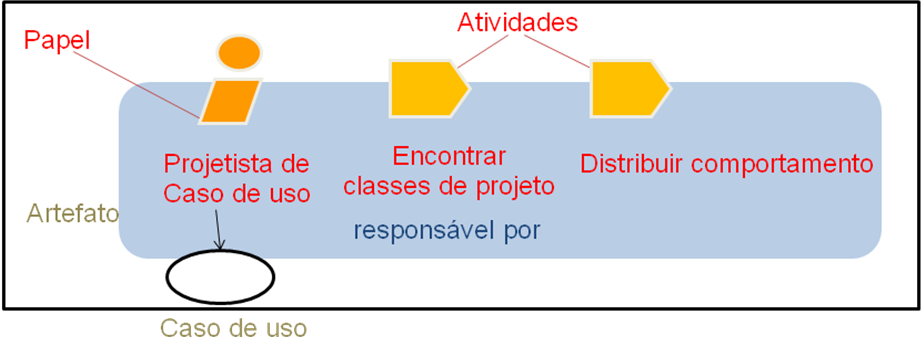


Figura 1.3 Descrição do processo de[bRational Unified Process]

* Papéis - representam quem será responsável por determinada função (tarefa);
* Atividade - representa uma unidade de trabalho que um indivíduo exercendo um papel pode ser solicitado a executar;
* Artefatos - representam os elementos tangíveis de um projeto. Podem assumir várias formas, como: modelo, documento, código fonte ou executável;
* Fluxos de trabalho (workflows) - representa o quando. Uma forma de descrever seqüência significante de atividades que produzem algum artefato e que mostram iterações entre os papéis.

* + 1. **Fases do RUP**

O ciclo de desenvolvimento do RUP é dividido em 4 fases que são: Concepção, Elaboração, Construção e Transição. O ciclo de vida termina com a entrega do produto final. Um erro que frequentemente acontece é que, apesar das fases serem sequenciais, as pessoas as tratam como processo em cascata, onde a primeira fase deve ser encerrada antes de começar a próxima. É interessante lembrar que o RUP é iterativo e baseado em riscos [DANTAS 2008].

A finalidade das fases do RUP não é o de dividir as atividades por tipo (análise, implementação e etc.). Isso já é algo que e concebido pelo conceito de disciplina. O real objetivo das fases é o de fazer o suficiente para que uma determinada atividade atinja os seus objetivos da fase e ao mesmo tempo concluam os milestones. O que faz com que chegue aos objetivos de cada etapa são os riscos envolvidos nas mesma, por exemplo:

* Na fase de Concepção temos que analisar se o projeto compensa financeiramente;
* Elaboração - O foco é nos riscos técnicos, examinando a arquitetura revendo o escopo para que os requisitos fiquem claros;
* Construção - Lida com os riscos lógicos do projeto. Está relacionado a pessoas e infra-estrutura;
* Transição - O risco está associado com a entrega do produto ao usuário final.

**Concepção**

Para quem é iniciante no RUP pode querer desistir, devido a sua complexidade. É por isso que tem que ter um bom entendimento dos objetivos das fases do RUP e analisar o que cada objetivo representa no contexto.

**Marco do Objetivo**

**Concepção**

**Elaboração**

**Construção**

**Transição**

**fim**

**Início**

**tempo**

**Figura 1.4 Fase de concepção**

Na fase de Concepção o foco é em entender o escopo e os objetivos do projeto, e então obter informações suficientes para se decidir se vai seguir com o projeto ou abortá-lo. Os principais objetivos são:

1. Compreender o que se vai ser construído, determinar o escopo do sistema, entendimento do escopo do sistema por todos os *stakeholders*. Para um entendimento de todas as pessoas no projeto é necessário algumas atividades como:

* Produção de um documento de visão: Esse documento pode está pronto na fase de Concepção, mas pode ser refinado ao longo do projeto. Deve apresentar claramente aos *stakeholders* os benefícios do sistema, problemas que serão resolvidos pela aplicação, os principais usuários do sistema e os principais requisitos não funcionais.

Nessa fase deve-se detalhar os atores e casos de uso centrais, assim como gerar protótipos de interface com o usuário que permitirão a validação dos fluxos de eventos com os *stakeholders* e identificarão as funcionalidades centrais do sistema.

Neste momento poderá ser necessário a implementação de uma parte da arquitetura para que os riscos sejam aceitáveis.

Entender o que vai desenvolver é importante, mas determinar os custos e prazos é crucial para desenvolver um projeto

Compartilhar uma visão de como o produto será desenvolvido, ou seja, qual o processo será utilizado.

**Elaboração**

Essa fase as diferenças com o modelo cascata fica evidente em relação ao modelo iterativo, onde apresenta as vantagens do modelo iterativo.

**Concepção**

**Elaboração**

**Construção**

**Transição**

**Fim**

a d

a

**Início**

**tempo**

**marco da arquitetura**

**Figura 1.5 Fase de elaboração**

O objetivo dessa fase é definir e consolidar uma arquitetura de sistema provendo uma base estável para as atividades de projeto e implementação. Os riscos associados a esta fase estão relacionados a requisitos, arquitetura, custo e cronograma e processo e ferramentas. Estes riscos serão analisados abaixo:

* Em relação aos requisitos - tem-se que analisar se a aplicação está correta;
* Na arquitetura - tem-se que analisar se a solução correta está sendo construída;
* Custos e cronogramas- se realmente você está no caminho certo;
* Processo e ferramentas- Se possuem o processo e ferramentas corretos para fazer o trabalho. Quando não há o conhecimento do domínio do sistema, esse objetivo poderá requerer mais iterações para que possa ser atingido e assim os riscos mitigados.

A fase de elaboração é onde a arquitetura do software na qual os requisitos que mais chocam a arquitetura são capturados em forma de caso de uso. Há identificação dos riscos que o projeto proporciona e ao final dessa fase deve ser possível estimar custos, elaborar cronogramas e plano de construção do sistema. Está é a fase mais critica, porque é nela que a engenharia é considerada completa e os custos para modificação do sistema aumentam à medida que o projeto avança. É nessa etapa que o projeto passa de custos e riscos baixos para custos e riscos altos.

**Construção**

Essa fase está relacionada aos riscos “lógicos”, e a maior parte do trabalho é realizado. É a etapa de construção do produto.

**Início**

**Concepção**

**Elaboração**

**Construção**

**Transição**

**Fim**

**Tempo**

**marco da capacidade**

**inicial**

**Figura 1.6 Fase de Construção**

Os objetivos principais dessa etapa são:

* Diminuir os custos de desenvolvimento, otimizar os recursos e evitar retrabalho desnecessário;
* Concluir a análise, o projeto, o desenvolvimento e o teste de todas as funcionalidades necessárias;
* Desenvolver o modelo iterativo e incremental do produto completo que esteja pronto para a transição para a comunidade de usuários. Implica também em descrever os casos de uso restantes e outros requisitos,incrementar o projeto,concluir a implementação e testar o software;
* Decidir se o software, os locais e os usuários estão prontos para que o aplicativo seja implementado;
* Atingir certo paralelismo entre o trabalho das equipes de desenvolvimento.

As atividades básicas da fase de construção são:

* Gerenciamento de recursos, otimização de controle e processo;
* Desenvolvimento completo do componente e teste dos critérios de avaliação definidos e a avaliação dos releases do produto de acordo com os critérios de aceitação para a visão.

Os critérios de avaliação para a etapa de construção envolvem as seguintes perguntas:

* Este release do produto é estável e desenvolvido o suficiente para ser implantado na comunidade de usuários?
* Todos os envolvidos estão prontos para a transição com a comunidade de usuários?
* As despesas reais com recursos ainda são aceitáveis se comparadas com as planejadas?

**Transição**

Nessa fase, o sistema está pronto. Começa a implantação do sistema para o usuário final. O foco nessa fase é garantir que o software esteja disponível para os usuários finais.

**Concepção**

**Elaboração**

**Construção**

**Transição**

**Fim**

**Início**

**Tempo** **Marco da liberação**

**do produto**

**Figura 1.7 Fase de Transição**

No final do ciclo de vida da fase de transição, os objetivos devem ter sido atendidos e o projeto deve está concluído. Em alguns casos o ciclo de vida atual pode combinar com o início de outro ciclo de vida,levando a próxima geração do mesmo produto.

O ciclo de desenvolvimento é quando o processo passa pelas 4 fases do RUP e os ciclos sucessivos é chamado de evolução de ciclos.

tempo geração1

Início do ciclo de desenvolvimento

tempo geração 2

Próxima evolução do ciclo

Evolução

Concepção

Elaboração

Construção

Transição

Evolução

Concepção

Elaboração

Construção

Transição

Figura 1.8 Ciclo de desenvolvimento

A fase de transição pode ser mais simples ou mais complexa, depende do tipo do produto. Por exemplo, uma release de um produto de prateleira, pode ser algo simples, enquanto que a substituição de um sistema de controle aéreo é algo complexo.

As atividades que serão realizadas nas iterações dependem muito da meta que se deseja. Por exemplo, para corrigir erros do sistema, normalmente utilizo as atividades de implementação e teste. As atividades básicas dessa fase são: realizar planos de implantação, testar o produto final, criar release do produto, conseguir *feedback* do usuário, concordar o produto com base no feedback e disponibilizar o produto para o usuário final.

* + 1. **Princípios básicos do RUP**

Os seis princípios mais importantes do RUP:

1. **Desenvolver software iterativamente**

Permite entregar um sistema completo e aumenta a funcionalidade de cada subsistema a cada *release.*

1. **Gerenciar requisitos**

O RUP descreve como documentar a funcionalidade,restrições de sistema, restrições de projeto e requisitos de negócio.

Os casos de uso cenários são exemplos de artefatos que são dependentes do processo que são muito eficazes na captura de requisitos funcionais.

1. **Usar arquiteturas baseadas em componentes**

A arquitetura baseada em componentes cria um sistema que pode ser facilmente extensível, requerendo a reutilização de software e um entendimento intuitivo. O RUP oferece uma forma metódica para construir esse sistema, focando-se em produzir uma arquitetura executável nas fases iniciais do projeto, ou seja, antes de comprometer recursos em larga escala.

1. **Modelar visualmente o software**

O uso de modelos visuais pode permitir aos clientes que tenham um melhor entendimento de um dado problema, e dessa forma possam se envolver mais no projeto como um todo.

A linguagem UML é amplamente utilizada no RUP para representar projetos.

1. **Verificação contínua da qualidade**

Não ter a preocupação com qualidade é uma falha muito comum em projetos computacionais. As pessoas envolvidas em um projeto estão preocupadas com a qualidade do software após o término dos projetos, ou acham que a qualidade é responsabilidade de grupo especifico, diferente da equipe de desenvolvimento. O RUP visa auxiliar no controle do planejamento da qualidade, analisando a construção de todo o processo e envolvendo todos os membros da equipe de desenvolvimento.

1. **Controle de mudanças**

Em todos os projetos de software a existência de mudanças é inevitável. O RUP define métodos para controlar e monitorar mudanças. Como uma pequena mudança pode afetar aplicações de formas inteiramente imprevisíveis, o controle de mudanças é essencial para o sucesso de um projeto.

O RUP também define áreas de trabalho seguras, garantindo a um programador que as mudanças efetuadas em outro sistema não afetarão o seu sistema.

* + 1. Disciplinas

Uma disciplina no RUP expressa todas as atividades que devem ser realizadas para produzir um conjunto de artefatos.

O RUP organiza suas disciplinas da seguinte forma:

* Fluxos de processo correspondem às atividades de desenvolvimento: modelagem de negócios, requisitos, análise e projeto, implementação, testes e implantação.
* Fluxos de suporte correspondem às atividades de gerenciamento e infra-estrutura: gerenciamento de configuração e mudanças, gerenciamento de projeto e ambiente.

As disciplinas Fluxos de processo são divididas em 9, que são elas:

* **Modelagem do negócio**: Os objetivos desta disciplina são de compreender a estrutura e a dinâmica da organização-alvo; entender os problemas da organização; fazer com que os clientes, desenvolvedores tenham entendimento comum em relação à organização e obter os requisitos necessários do sistema para sustentar a organização.

A disciplina de Modelagem e Negócios é baseada em modelos de casos de uso de negócios e um modelo de objeto de negócios. O modelo de caso de uso de negócio descreve uma sequência de atividades necessárias para fornecer um resultado para o ator de negócio e o modelo de objetos de negócio descreve os casos de uso de negócios.

* **Requisitos:** Na disciplina de requisitos o foco é o de manter concordância entre os clientes e as partes interessadas no que diz respeito aquilo que o sistema deve fazer; fornecer aos desenvolvedores uma melhor compreensão dos requisitos do sistema; ter uma base para estimar custos e prazos de desenvolvimento do sistema e definir uma interface para o usuário ter um entendimento de como o sistema funcionará de acordo com as suas necessidades.

Os modelos de caso de uso de negócios e objetos de negócio utilizado na disciplina de Modelagem e Negócio servirão de informações importantes para esse fluxo. Além desses documentos, serão criados um documento de visão, modelo de caso de uso, casos de uso e especificação suplementar, isso para descrever o que o sistema fará, onde todos os envolvidos são fontes de informações. Um documento de visão descreve qual o entendimento que as partes envolvidas do sistema têm dele. O Modelo de casos de uso descreve as funções fornecidas pelo sistema. Um Caso de uso representa uma sequência de ações que resultam em valor para um ator específico. Especificação suplementar captura os requisitos que não são capturados pelos casos de uso nos modelo de casos de uso. Como exemplos de especificação suplementar podem citar: atributos de qualidade do sistema: confiabilidade, usabilidade e etc; sistemas operacionais e etc.

* **Análise e Projeto:** Os objetivos desse fluxo é o de mudar os requisitos em um projeto do que o sistema será; Obter uma arquitetura robusta do sistema e adaptar e adaptar o projeto de acordo com o ambiente de implementação. O modelo de análise e projeto é o principal objetivo dessa disciplina. O modelo de análise e projeto possui a realização dos casos de uso. A realização de casos de uso expressa o que o design espera de um caso de uso, um exemplo seria os diagramas de classes das classes e dos subsistemas participantes.
* **Implementação:** o objetivo é construir um sistema, produzindo o código executável e organizando o código em termos de subsistemas de implementação organizados em camadas; implementar classes e objetos em termos de componentes (arquivo fonte, executável, binários e etc). A disciplina de implementação limita-se ao escopo do sistema e como as classes individuais devem ser testadas em unidades.
* **Teste:** a disciplina de teste oferece serviços a outras disciplinas. O teste foca principalmente na avaliação da qualidade do produto, realizada através de algumas práticas: encontrar e documentar erros na qualidade do software, validar as funções da maneira que foi projetada, verificar se os requisitos foram implementados de forma correta...Uma diferença interessante entre o teste e as outras disciplinas do RUP é que a sua principal finalidade é localizar e expor os pontos fracos do software.A diferença é enquanto as outras disciplinas enfocam na abrangência,o teste enfatiza na deficiência.
* **Implantação:** Descreve as atividades que possibilitem que o software esteja disponibilizado para o usuário final. A ênfase da disciplina de implantação é testar o produto no ambiente de implantação,onde são realizado testes beta,antes de ser entregue ao usuário final.

As disciplinas de fluxo de suporte correspondem a 3, que são elas: ambiente, gerência de configuração e mudanças e gerência de projetos.

* **Ambiente:** A disciplina de ambiente oferece o ambiente de suporte para um projeto.Ao fazer isso, ela também serve de suporte a todas as outras disciplinas.A meta das atividades dessa disciplina é oferecer à organização o ambiente de desenvolvimento de software,processos e ferramentas que dará suporte à equipe de desenvolvimento.
* **Gerência de configuração e mudanças:** O gerenciamento de configuração e mudanças envolve a identificação de itens de configuração, restrições a mudanças nestes itens, verificação de mudanças feitas nos itens e definição e gerenciamento das configurações dos mesmos através do processo de desenvolvimento. Os métodos,processos ferramentas que são usados para fazer esse controle em uma organização pode ser considerada como um sistema CM. O sistema CM manuseia as informações importantes sobre o processo de desenvolvimento, a implantação e a manutenção,além de conservar o acervo base de artefatos potencialmente reusáveis, resultando da execução destes processos.

É essencial para o controle de numerosos artefatos produzidos pelas várias pessoas que trabalham em um projeto. O controle ajuda a evitar confusões dispendiosas e garante que os artefatos resultantes não são conflitantes em relação a questões como: atualização simultânea, notificação limitada e múltiplas versões.

* **Gerência de projetos:** O gerenciamento de projetos de software é a arte de balancear os objetivos, gerenciamento de riscos e restrições para entregar, com sucesso, um produto que esteja de acordo com as necessidades dos clientes e usuários.A meta do gerenciamento de projetos do RUP é tornar esta tarefa mais fácil. O propósito do fluxo de gerenciamento de projetos é fornecer um framework para gerenciamento de projetos de software; providenciar guidelines para o planejamento, recrutamento de pessoal, execução e monitoração dos projetos e providenciar um framework para o gerenciamento de riscos.

# OpenUp

Originalmente o OpenUp era conhecido como BUP (Basic Unified Process) pela IBM,que em 2005 foi liberado para a fundação Eclipse e renomeado para OpenUp [DOURADO,2009].

O OpenUp é considerado um processo híbrido,pois foi desenvolvido pela IBM com base nos processos RUP e XP, tendo como principal objetivo reunir as melhores características de cada uma dessas metodologias. Apesar dessa visão, o OpenUp possui princípios e práticas que são baseados em processos definidos (que se não seguidos, não acontecerá de maneira correta), assim como RUP. Por isso, que esse processo está sendo citado como uma metodologia tradicional, devido às suas práticas serem baseadas em

O OpenUP surgiu a partir das diferentes necessidades encontradas em vários projetos. Também existe grande influência da atual dinamicidade nos negócios, as quais refletem diretamente nos requisitos dos softwares. Para tanto valoriza a colaboração entre a equipe e os benefícios aos interessados, ao invés da formalidade e entregáveis desnecessários. O conteúdo fornecido é considerado completo, pois são encontradas definições de papéis, artefatos e processos necessários para o desenvolvimento de um projeto de pequeno porte, com equipe co-localizada.

**1.3.1 OpenUp e suas características**

O OpenUP (*Open Unified Process*) é um processo de desenvolvimento de software de código aberto*,*projetado para equipes pequenas e centralizadas, que éatualmente mantido pelo Projeto Eclipse, que define um *framework* de processo de desenvolvimento de *software*. Assim, este processo unificado aplica uma abordagem iterativa e incremental dentro de um ciclo de vida estruturado. Contudo, abraça uma filosofia pragmática e ágil que foca na natureza colaborativa do desenvolvimento de *software* [ MONTEIRO e YBANEZ,2009].

Além disso, é um processo independente de ferramenta e de pouca cerimônia que pode ser estendido para direcionar uma grande variedade de tipos de projeto.

O OpenUP é um processo: **mínimo**: apenas conteúdos fundamentais do processo são definidos; **completo**: o processo abrange todas as fases do ciclo de vida de desenvolvimento; **extensível**: o processo pode ser utilizado na forma como foi definido, mas permite que novos conteúdos de processos sejam acrescentados para atender de forma mais completa as características de um determinado projeto. [CUNHA e VASCONCELOS ]

* + 1. **Visão Geral do OpenUp**

O processo pode ser facilmente entendido através das 3 camadas abaixo descritas:



**Figura 1.9 Visão Geral do OpenUp**

**Ciclo de Vida do Projeto** – Fases com foco nas necessidades dos Stakeholders.  
  
O OpenUP apresenta a mesma distribuição de fases já conhecidas no RUP, onde o critério de saída de cada fase é no mínimo atender as seguintes respostas:

**Iniciação**:Todos os *stakeholders* concordam com o escopo e objetivos do projeto?  
**Elaboração**: Todos concordam com a arquitetura proposta e o valor entregue ao cliente considerando os riscos levantados?

**Construção**:Existe uma aplicação que está quase pronta rodando bem próxima a ser finalizada?

**Transição**: A aplicação está finalizada e o cliente satisfeito?

**Ciclo de Vida da iteração com foco no time:**

Além da divisão por fases já conhecida, o OpenUP divide o projeto em iterações (também conhecidas como Sprints segundo a metodologia SCRUM) planejadas que podem variar de alguns dias a algumas semanas (a media recomendada é de 4 semanas podendo ser reduzida ou aumentada em até aproximadamente 6 semanas). Ao final de cada iteração deve ser gerado um incremento ao Produto (Build executável ou demo). Ao final de cada iteração geralmente é realizada uma retrospectiva e avaliação onde são discutidas as lições aprendidas e a saúde do projeto. Vale mencionar que o principal objetivo da retrospectiva é aprender com erros e acertos e não apontar culpados.

**Micro incrementos** com foco individual: Um micro incremento é a execução de um pequeno passo que deve ser mensurável para alcançar os objetivos de uma iteração. Este pode representar o resultado de alguns dias ou horas de trabalho de uma pessoa ou um grupo determinado.

**1.3.3 Fa ses do OpenUp**

O OpenUp é dividido em quatro fases:concepção,elaboração,construção e transição. Cada fase pode ter quantas iterações forem necessárias, dependendo do grau de incerteza do domínio de negócio, da tecnologia a ser utilizada, da complexidade arquitetural, do tamanho do projeto, etc. Este processo de entrega define um processo de desenvolvimento de software que suporta os princípios fundamentais do OpenUP. Foi projetado para suportar equipes pequenas e co-localizadas, com de 3 a 6 membros e que trabalhe em um projeto que irá durar de 3 e 6 meses.

**Concepção**

A fase de concepção no OpenUp é responsável por conseguir que se tenha um entendimento simultâneo entre todos os *stakeholders* dos objetivos do ciclo de vida para o projeto.

Há quatro objetivos que clarificam o escopo,os objetivos do projeto e a viabilidade da solução pretendida: **Entender o que construir** a partir da visão dos stakeholders a respeito do produto a ser desenvolvido,quem é o stakeholder do sistema e porque; **Definir** o escopo dos sistemas e seus limites; **Identificar** quais os requisitos mais críticos;Definir pelo menos uma **arquitetura candidata** e sua aplicação prática e **Entender** o custo,cronograma e os riscos associados ao projeto.

**Elaboração**

Com a fase de elaboração o propósito é o de estabelecer uma linha de base da arquitetura do sistema para o volume de esforço de desenvolvimento na próxima fase.Algumas finalidades para a fase de elaboração podem ser citadas,como:

Obter um entendimento mais detalhado dos requisitos permite ao usuário criar um plano mais detalhado e obter comprometimento dos *stakeholder;* Projete,implementee teste um esqueleto da estrutura do sistema; Apesar da funcionalidade não estar completa ainda, a maior parte das interfaces entre os blocos sendo construídos é implementada e testada. Isto é conhecido como uma **arquitetura executável**; **Mitigue os riscos essenciais e** produza **um cronograma e uma estimativa de custos precisos.** Muitos riscos técnicos são resolvidos como resultado do detalhamento dos requisitos e do projeto, implementação e teste da arquitetura e Refine e detalhe o plano de projeto de alto nível.

**Construção**

A finalidade da fase de construção é de terminar o desenvolvimento do sistema baseado na arquitetura colocada na linha de base. Existem objetivos para a fase de Construção que nos ajudam a ter o desenvolvimento com custo eficiente de um produto completo,como por exemplo:

**Desenvolver de forma iterativa** um produto completo com a descrição dos requisitos que faltam, preencher os detalhes do projeto, terminar a implementação e testar o software; Liberar a primeira versão beta do sistema e determinar se os usuários já estão prontos para que a aplicação possa ser implantada; **Minimizar os custos de desenvolvimento e conseguir algum grau de paralelismo** entre os desenvolvedores ou as equipes de desenvolvimento, como por exemplo, atribuindo os componentes que podem ser desenvolvidos independentemente para desenvolvedores distintos.

**Transição**

A fase de Transição possui alguns objetivos que ajudam a fazer um ajuste na funcionalidade,desempenho e na qualidade total do produto beta proveniente da fase anterior,como por exemplo:

**Executar o teste Beta** para validar se as expectativas dos usuários foram atendidas**;** Realizar algumas atividades de ajuste fino,tais como reparação de erros e melhorias no desempenho e na usabilidade; Obter a concordância dos *stakeholders* de que a distribuição está completa,incluindo vários níveis de testes para aceitação do produto,como testes formais,informais e beta; **Melhorar o desempenho de projetos futuros com documentação das lições aprendidas** e Melhorar o ambiente de processos e ferramentas para o projeto.

* + 1. **Princípios básicos do OpenUp**

O OpenUP é baseado em 4 princípios básicos que são:

1. **Equilíbrar as prioridades concorrentes para maximizar o valor para os stakeholders**

Para que se alcance o sucesso,*stakeholders* e participantes do projeto devem convergir para um claro entendimento e concordar com três fatores: Problema a ser resolvido,Restrições impostas à equipe de desenvolvimento (custo, cronograma, recursos, regulamentos) e Restrições impostas à solução. Coletivamente, estes três itens representam os requisitos para o desenvolvimento do sistema. O desafio de todos os participantes do projeto é criar uma solução que maximize o seu valor para os stakeholders, mesmo que sujeitos a restrições. O equilíbrio está em fazer uma análise crítica do custo-benefício entre características desejadas e as decisões de projeto subsequentes que definem a arquitetura do sistema.

1. **Focar na evidenciação da arquitetura**

Sem uma base arquitetural,um sistema não evoluirá de forma eficiente,será difícil organizar a equipe ou comunicar idéias sem foco técnico comum que a arquitetura fornece. Dessa forma,utilize a arquitetura como um ponto focal,para que desenvolvedores possam alinhar seus interesses e idéias ao se tornar evidentes as decisões técnicas essenciais tomadas em cada evolução arquitetural.

1. **Colaborar para alinhar os interesses e compartilhar o entendimento**

Em uma equipe cada membro tem seus próprios conhecimentos, habilidades e maneiras de fazer as coisas. Este princípio tem a finalidade de alinhar essas diferenças de forma que o projeto seja beneficiado bem como fazer com que todos os membros da equipe tenham um entendimento sobre o projeto. O continuo aprendizado também é estimulado por este principio fazendo com que cada membro da equipe desenvolva mais habilidades e incremente seus conhecimentos.

1. **Evoluir para continuamente obter feedback e promover melhorias**

O objetivo deste princípio é fazer com que através de feedbacks, obtenha-se modos de melhorar o produto e também o processo da equipe envolvida. Através de feedbacks, pode-se identificar potenciais riscos e tratá-los mais cedo durante o projeto.

É importante ter o objetivo do projeto de maneira clara ao próprio entendimento para que seja possível fazer uma medição do progresso e identificar possíveis melhorias no processo.

* + 1. **Disciplinas**

As disciplinas do OpenUP são divididas em 6. Elas serão detalhadas logo abaixo:

1. **Análise e Projeto**

Os propósitos da Análise e Projeto são: transformar os requisitos em um projeto do que será o sistema,desenvolver uma arquitetura robusta para o sistema e adaptar o projeto para corresponder com ambiente de implementação. A disciplina de Análise e Projeto está relacionada a outras disciplinas, como por exemplo: a disciplina de Requisitos provê a primeira entrada para a Análise e Projeto;a disciplina de Implementação implementa o projeto; a disciplina de Teste testa o projeto do sistema durante a Análise e Projeto e a disciplina de Gestão de Projetos planeja o projeto e cada iteração.

1. **Gerência de Configuração e Mudança**

A finalidade desta disciplina é: Manter um conjunto de produtos de trabalho consistente a medida que evolui, manter construções de software consistentes,fornecer meios eficientes para se adaptar às mudanças, re-planejando o trabalho adequadamente, fornecer dados para a medição do progresso e está relacionada ao controle de mudanças de artefatos pela configuração e a habilidade de manter versões e configurações consistentes dos artefatos.

1. **Implementação**

A disciplina de implementação tem como propósitos: Construir o sistema de forma incremental e verificar que as unidades técnicas usadas para construir o sistema funcionem como especificado.

Esta disciplina se relaciona com as outras disciplinas da seguinte forma: A disciplina de Requisitos define o que será implementado; a de Análise e Projeto organiza e define o escopo da implementação;a de Teste valida que a construção do sistema atende aos requisitos;a de Gerência de Configuração e Mudança fornece mecanismos para controlar mudanças no sistema em construção e a disciplina de Gerência de Projeto planeja quais funcionalidades serão implementadas em cada iteração.

1. **Gerência de Projetos**

A disciplina de Gerência de Projetos está focada em:manter a equipe focalizada na entrega contínua do produto de software testado para a avaliação dos *Stakeholders;* ajudar a priorizar à seqüência de trabalho;ajudar a criar um ambiente de trabalho eficaz para maximizar a produtividade da equipe;manter os *Stakeholders* e a equipe informados sobre o progresso do projeto e fornecer uma estrutura para controlar o risco do projeto e para adaptar-se continuamente às mudanças

O gerenciamento de projeto age como um elo entre os *Stakeholders* e a equipe de desenvolvimento. É interessante que as atividades da gerenciamento de projeto adicionem valor ao criar um ambiente de trabalho de elevado desempenho onde os

*Stakeholders* tenham confiança na habilidade da equipe de conhecer as capacidades e restrições da plataforma técnica e de entregar com sucesso algo valoroso e que os membros da equipe de projeto entendam as necessidades dos *Stakeholders,*produzindo continuamente um produto de software para avaliação.

1. **Requisitos**

A disciplina de requisitos tem como atividades: entender o problema a ser resolvido; entender as necessidades dos *Stakeholders*;definir os requisitos para a solução;definir o escopo do sistema;identificar interfaces externas ao sistema;identificar restrições técnicas na solução;fornecer a base para o planejamento das iterações e fornecer a base inicial para a estimativa de custo e cronograma. Para realizar essas atividades, é importante que compreendam a definição e o escopo do problema que estamos tentando resolver. A disciplina de Requisitos é relacionada às outras disciplinas das seguintes maneiras: A disciplina de Análise e Projeto obtém suas entradas primárias a partir da disciplina de Requisitos; a de Teste valida o sistema de acordo com os requisitos; a de Gerência de Configuração e Mudança fornece os mecanismos para controlar as mudanças nos requisitos; a de Gerência de Projeto planeja o projeto e atribui os requisitos a cada iteração analisando os requisitos priorizados e atribuindo o trabalho.

1. **Teste**

O propósito desta disciplina é: Encontrar e documentar defeitos,validar e provar as suposições feitas no projeto e requisitos especificados através de demonstrações concretas,validar que o produto de software foi feito como projetado e validar que os requisitos estão apropriadamente implementados.

A disciplina de Teste está relacionada às outras disciplinas das seguintes maneiras: A disciplina de Requisitos captura requisitos para o produto de software, que é um das contribuições primárias para identificar que testes executar; A de Análise e Projeto determina o projeto apropriado para o produto de software, que é outra contribuição importante por identificar que testes executar; A de Implementação produz construções do produto de software que é validado pela disciplina de Teste;A de Gerência de Projeto planeja o projeto e o trabalho necessário em cada iteração e a de Gerência de Configuração e Mudança controla mudanças dentro do projeto. O esforço de teste verifica se cada mudança foi completada adequadamente. Ativos de teste são mantidos abaixo da gerência de configuração.

# MSF

O Microsoft Solutions Framework (MSF) surgiu em 1994, a partir da análise de como a Microsoft desenvolve seus produtos. Basicamente o MSF é uma compilação das boas práticas utilizadas pela empresa, que foi criado tanto para uso interno como para uso de seus clientes. Porém, apesar de ter sido criado pela Microsoft, o MSF aborda basicamente o processo de construção de soluções, não se prendendo ao uso de produtos desta empresa.[ CARDIM 2006]. Inicialmente, surgiu o MSF 3.0 que era composto por quatro elementos básicos: princípios fundamentais,modelo de processo,modelo de equipe e disciplinas. Dentro da totalidade da popularização dos processos ágeis,a versão 4.0 do MSF foi lançada para atender duas extremidades especificas que são: MSF Agile Software Development(MSF4ASD) abordando processos ágeis e na outra vertente o MSF for CMMI Process Improvement (MSF4CMMI) que aborda os processos tradicionais. No entanto,mesmo assim,existe muito semelhança entre as duas versões.

Ao longo do tempo o MSF já se chamou: MDF (Microsoft Development Framework) e PCM (Product Cycle Model). Consolidando-se como MSF após o livro Microsoft Secrets de Michael Cusumano (1995). A história da própria Microsoft e do MSF se misturam, podemos então considerar o MSF como a “atitude mental da Microsoft” para seu modelo de negócios.

Microsoft

Grupo de produtos worldwide

Microsoft

Serviços de consultoria

Microsoft

Grupo de tecnologia da informação

Microsoft

Parceiros

**Microsoft Solução de**

**framework**

F**igura 1.10 Microsoft Solução de framework**

**1.4.1 MSF e suas características**

O MSF é um processo iterativo e incremental,onde as atividades são realizadas de forma cíclica, resultando em produtos incrementais;assim como o RUP e OpenUp. O MSF permite uma fácil compreensão tanto por parte da equipe como do cliente,além de ser bem flexível em sua aplicação; Há o aprendizado contínuo do projeto,onde cada lição é uma grande lição para equipe envolvida no processo de desenvolvimento; Pode-se perceber a evolução da curva de produtividade; Acompanhar a autonomia técnica crescente dos colaboradores a cada dia dentro do projeto e Possui uma teoria de controle,onde pequenas iterações permitem calibrar com mais precisão e agilidade estimativas e reduzir margens de erro.

**1.4.2 Visão Geral do MSF**

Um projeto MSF é regido por iterações ou ciclos. A ciclo, cada componente da equipe executa suas funções e atualiza o resultado do seu trabalho conforme a necessidade. Os ciclos se repetem até que o projeto seja concluído ou cada versão seja lançada. Cada componente da equipe será responsável por um ou mais papéis, dependendo do tamanho ou da complexidade do projeto. O MSF é dividido em dois modelos: Modelo de time(Team Model) e seus respectivos processos e Modelo de processo(Process Model).

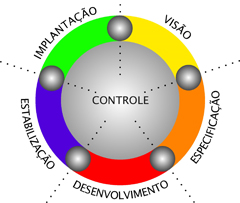
O modelo de time habilita a escalabilidade do projeto,identifica quem vai trabalhar durante o projeto e definir cada time com um responsável.

O modelo de processo será falado na próxima seção.Este é um modelo que trabalha em conjunto com o modelo de time organizando o processo em fases distintas.

**Fig 1.11. Modelo de times**

O modelo de time conjuntamente com o modelo de processos possuem objetivos e metas da equipe. A fase de **Gerenciamento de programa** atua nas áreas de gerenciamento de projeto, soluções em arquitetura,garantia de processos e serviços administrativos;**Desenvolvimento** atua nas áreas de consultoria tecnológica, implementação da arquitetura e design, desenvolvimento da aplicação e desenvolvimento da infra-estrutura; **Teste** atua nas áreas de plano de teste,engenharia de teste e reporte de teste; **Gerenciamento de release** está relacionada aos processos de infra-estrutura, suporte, operações,logística e gerenciamento comercial das publicações; **Experiência do usuário** atua nas áreas de acessibilidade, internacionalização, treinamento/material e suporte, pesquisa de usabilidade e teste,advogado do usuário e design de interface; **Gerenciamento do produto** satisfaz os clientes com valor de negócio,marketing,advogado do cliente e planejamento do produto. O princípio básico deste modelo é que cada um desses papéis aborda um objetivo importante para o projeto. Por isso todos os papéis devem estar representados e poder comunicar-se entre si, além de participar das decisões de projeto.

**1.4.3 Fases do MSF**

****

**1.12. Fases do MSF**

O modelo de processos do MSF em dividido em 6 fases: Visão,Especificação,Desenvolvimento,Estabilização,Implantação e Controle. Cada fase descreve um conjunto de subprodutos que devem ser entregues,assim como marcos que devem ser atingidos e os respectivos critérios de aceitação. Essas fases serão descritas a seguir:

1. **Visão:** Esta fase tem como foco fazer com que a equipe tenha uma visão comum do projeto. As atividades realizadas nessa fase são: formação de equipe,elaboração do projeto(visão mais ampla do que projeto deve fazer) e a definição do escopo do projeto.
2. **Especificação:** Durante essa fase a equipe estima custos,prepara plano de trabalho e programa os deliverables.É nesta fase que a equipe lista os requisitos do projeto(requisitos de negócio,do usuário,operacionais e de sistema).Esses são os requisitos que o MSF reconhece.
3. **Desenvolvimento:** nessa fase a equipe implementa a maioria dos componentes da solução. Os principais artefatos gerados nessa fase são:código fonte e executável, scripts de instalação e configuração, especificação funcional, elementos de suporte a performance, especificação e casos de teste.
4. **Estabilização:** esta fase tem como objetivo testar o que foi implementado na fase anterior. Esses testes enfatizam o uso do sistema simulando o ambiente de funcionamento. A equipe se preocupa em resolver erros encontrados nos testes e prepara a solução para liberação. Esta fase só termina quando todos os erros são corrigidos.
5. **Implantação:** nessa fase, a equipe estabiliza o produto e obtém a aprovação do cliente final. As atividades de estabilização do produto podem continuar durante este período enquanto os componentes do projeto são transferidos do ambiente de teste para o ambiente de produção. O aspecto relacionado à satisfação do cliente deve ser coletada em todas as fases.
6. **Controle:** A fase de controle está presente em todas as fases do projeto. É responsável por gerar o cronograma, para que as pessoas envolvidas possam se programar e concluir as suas atividades no prazo estipulado. E um plano de tarefas é construído, para que cada membro da equipe possa ser gerenciado e acompanhado semanalmente.

**1.4.4 Princípios básicos MSF**

Nessa seção, iremos apresentar os princípios básicos do MSF,que se compõem na filosofia e comportamento para as equipes que utilizam o conceito do MSF no desenvolvimento dos seus projetos de software.

1. **Parceria com o cliente**

A idéia é ter o cliente como membro do time para fazer validações constantes.Dessa forma está comprometido com o projeto e possa entender o que está sendo feito.Assim,os riscos do projeto podem ser mitigados.

1. **Qualidade é trabalho de todos**

O termo qualidade é muito subjetivo. Promover qualidade significa investir em pessoas,ferramentas e processos. Requer tanto prevenção de “bugs/problemas” quanto verificação de possíveis soluções.

1. **Trabalho em direção a uma visão compartilhada**

Segundo Steve McConnel,uma pesquisa feita com 75 times mostra que em todos os casos em que o time funciona eficazmente,todos os membros conheciam os seus objetivos! Através da visão do projeto os membros do time podem definir prioridades, tomar decisões e garantir que os esforços estejam alinhados aos resultados que se esperam.

1. **Manter-se ágil, adaptar-se às mudanças**

Quanto mais uma organização procura maximizar o impacto no negócio de um investimento em tecnologia, mais ela descobre novos ambientes e desafios.

Os projetos de tecnologia têm uma característica relevante: mudanças constantes. Alterações no projeto devem ser esperadas e é impossível isolar a entrega do projeto. Com isso, o MSF foi desenvolvido para gerenciar e antecipar a mudanças. Todas as alterações são aprovadas pela equipe, dessa forma melhora o impacto das mudanças e mitigar os impactos negativos.

1. **Encorajar comunicação aberta**

Para desenvolver um bom trabalho, é necessário que todos os membros da equipe tenham conhecimento prévio do que está sendo feito. Isso minimizar o desconhecimento, incertezas e retrabalho.

1. **Autorização dos membros da equipe**

Dar poder aos membros da equipe é um grande diferencial do MSF, pelo fato de pregar um modelo em rede hierárquica, onde cada membro é responsável pela entrega do produto.

1. **Estabelecer a responsabilidade desobstruída e responsabilidade compartilhada**

A definição clara dos papéis e das responsabilidades de cada membro da equipe é um dos principais fatores de sucesso. Sem isso implica em um retrabalho duplicado e certa insegurança em relação a função. Um estudo mostrou que esse principio diminui as incertezas quanto “o que”, “quem”, “quando” e “por que” com os resultados, tornando o trabalho mais eficiente e compensador.

1. **Foco em entregar um valor de negócio**

Os projetos de tecnologia não devem focar em “entregas de tecnologia”,mas em

“entregas com valor tangível ao negócio”.O projeto tem que possuir uma ligação intima com o negócio,se não existir essa ligação pode resultar em entregas com atraso e projetos cancelados.

1. **Aprender com todas as experiências**

Estatísticas mostram repetições de falhas em projetos. De fato, isso mostra que as pessoas não estão aprendendo com os erros para mudar esse quadro.E esse quadro piora mais ainda por que estamos diante prazos curtos e recursos limitados.O MSF recomenda revisões,coleta de lições aprendidas no projeto e criação de um documento no final do projeto,para que os erros não repitam.

1. **Criar sempre possibilidade de serem entregues produtos**

O time deve crer que o produto deve estar pronto para ser entregue a qualquer momento, mesmo no contexto de desenvolvimento de soluções.

* + 1. **Disciplinas do MSF**

As disciplinas são necessárias durante o ciclo de vida dos projetos e são guias constantes para cada modelo. O MSF assume três disciplinas que são:

**a) Gerenciamento de projeto**: é uma disciplina que incorpora atividades de diversas áreas de conhecimento; a maioria das responsabilidades sabidas da área de “gerência de projeto” são atribuídas ao individuo responsável pelo papel de gerente de projeto. A disciplina de gerenciamento de projeto ajuda o time a obter sucesso sem perder performance com recursos adicionais que não fornece valor suficiente aos recursos investidos.

**b) Gerenciamento de risco:** é o gerenciamento pró-ativo,compreensivo,visando o sucesso e diminuindo fatores negativos que impactariam no fracasso do projeto.A gerência de riscos é uma resposta à incerteza intrínseca em projetos de tecnologia.

**c) Gerenciamento de Aprendizado:**. A disciplina de gerenciamento de aprendizado identifica habilidades exigidas pelo time, alocando desse modo, recursos que o projeto necessita e criando oportunidades de aprendizado e crescimento.

# Considerações finais

Em virtude dos processos citados no capítulo, a definição dos processos tradicionais ajudará ao leitor a ter senso critico em escolher o processo que mais se adequar a sua empresa/organização.

Espero que o capítulo possa contribuir para o entendimento das metodologias e possa dar um embasamento das mesmas e que se há um processo bem definido e pessoas competentes para funções específicas, qualquer processo estará fardado a ter sucesso;independentemente da metodologia escolhida. Portanto, fica em aberto os possíveis questionamentos e propostas para novas pesquisas na área.

# Tópicos de Pesquisa

* **Aderência de um processo pesado(RUP) a um modelo de Qualidade(MPS.BR)**

Sabemos que uma empresa que vai fornecer um produto a um cliente deve ter alguns requisitos de qualidade, ele deve mostrar responsável e eficaz na construção do produto,caso contrário o processo de desenvolvimento está fadado ao fracasso.O uso de modelos de qualidade juntamente com processos tradicionais fornece segurança as informações necessárias para que o processo ocorra em conformidade.

* **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**

Ao analisarmos os processos tradicionais,iremos deparar com comparações com as metodologias ágeis.Para analisarmos assim, a fiel necessidade de uma metodologia ser usada em um determinado projeto e outra não.

* **Um estudo sobre Relações entre Papéis Funcionais do RUP e o Comportamento Pessoal no Trabalho em Equipe em Fábricas de Software**

Tem se visto que times de software têm buscado modelos consistentes para a montagem de times com melhor desempenho.O casamento correto entre o comportamento do indivíduo no trabalho em grupo e sua função no processo de desenvolvimento,tem tido uma atenção especial.

# Sugestão de Leitura

Para o leitor que queira conhecer o processo RUP, é indicado ler os seguintes livros: Introdução ao RUP: Rational Unified Process do autor Phillippe Kruchten, Conheça o Rational Unified Process do autor Mauro Viana.

Se quiser saber sobre aspectos gerenciais do RUP e o seu uso na implantação de normas, leia as monografia que se encontram nos seguintes site: <http://www.de9.ime.eb.br/~tssouza/eng_soft/Trabalho%20RUP/Mono_RUP.pdf>, <http://inf.unisul.br/~vera/egs/Rup_iso9000.pdf>

Para um conhecimento sobre os aspectos gerais da metodologia OpenUp,é ideal ler o artigo <http://www.eclipse.org/epf/general/OpenUP.pdf>, que se encontra na versão em inglês.

Para conhecer os dois mundos sobre os processos tradicionais e ágeis,recomenda-se ler este artigo <http://www.projectkoach.com/papers/OpenUP_2_worlds.pdf>, de Bjorn Gustafsson,em inglês.

Para ter um entendimento sobre os modelos de processos,MSF *agile* e CMMI está disponível o artigo <http://msdn.microsoft.com/pt-br/magazine/dd221363.aspx>, de Brian A.Randell,2008, o livro Microsoft Solutions Framework (MSF) de Marley Keeton Poderes,Jeff Carter, Geof Lory,Andrew McMurray e o site <http://imasters.uol.com.br/artigo/9324/dotnet/uma_entrevista_sobre_o_msf/>, onde tem se um artigo sobre o MSF,de Fábio Camara,2008.

# Exercícios

1. Por que o RUP é considerado um processo tradicional? Justifique a sua afirmação. Com base em leituras complementares,cite alguns exemplos de empresas que utilizaram/utilizam o RUP.
2. O que faz do processo RUP um modelo iterativo e incremental?
3. As fases do RUP são definidas em 4. Fale de forma resumida sobre os objetivos primordiais de cada fase.
4. Quais são as disciplinas do processo RUP? Qual o papel de cada uma?
5. Quais são as diferenças relevantes do RUP comparado ao OpenUp?
6. Fale sobre os princípios básicos do OpenUp.
7. Qual as disciplinas existentes no RUP que não se encontram no OpenUp. Fale sobre elas.
8. Quais são as fases do MSF? Detalhe cada uma.
9. Fale um pouco sobre o modelo de time do MSF.
10. Explique os princípios do MSF. Selecione os que você acha mais importantes para sua empresa.
    1. Referências Bibliográficas

*Rational Unified Process:Visão Geral.* Acessado em: 09/08/2009.Disponível em: em:[**http://www.wthreex.com/rup/portugues/index.htm**](http://www.wthreex.com/rup/portugues/index.htm).

Vianna, M.(2002). ***Conheça o Rational Unified Process (RUP).*Acessado em:** 09/08/2009. Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=79>.

Luiz, R.R.V(2009). *Obtendo Qualidade de Software com RUP*. Acessado em:12/08/2009. Disponível em: <http://javafree.uol.com.br/artigo/871455/Obtendo-Qualidade-de-Software-com-o-RUP.html>.

Perrelli, H.(2009). *Visão Geral do RUP*. Acessado em: 14/08/2009.Disponível em: <http://www.cin.ufpe.br/~if717/slides/3-visao-geral-do-rup.pdf>.

Cardim, I.C(2006). *Avaliando o Microsoft Solutions Framework for Agile Software Development em relação ao Extrem Programming.* Acessado em:05/09/2009.Disponível em:http://www.cin.ufpe.br/~tg/2005-2/icc2.pdf.

Camara,F(2007).*Um pouco da história do MSF*. Acessado em:26/09/2009. Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=1471>.

**Broering,E(2009).** *RUP-Rational Unified Process*. Acessado em:03/10/2009. Disponível em: [**http://www.devmedia.com.br/articles/viewcomp.asp?comp=4574**](http://www.devmedia.com.br/articles/viewcomp.asp?comp=4574)

Pires,Y. *Processo Unificado de desenvolvimento de software- RUP.* Acessado em:08/10/2009. Disponível em: [**http://www.laps.ufpa.br/yomara/paginav2/aps/processo%20unificado%20rup.pdf**](http://www.laps.ufpa.br/yomara/paginav2/aps/processo%20unificado%20rup.pdf).

Nogueira,G.L e Capra,S.P.M(2003). *Adaptação do RUP para Projetos de Software E-Commerce.* Acessado em:15/10/2009. Disponível em: <http://www.cci.unama.br/margalho/portaltcc/tcc2003/d2615.pdf>.

Martins, C.E.S, e Cabral, A.Y. *Gestão do conhecimento em um processo de desenvolvimento de software.* Acessado em:15/10/2009. Disponível em:

[**http://guaiba.ulbra.tche.br/pesquisas/2008/artigos/sistemas/328.pdf**](http://guaiba.ulbra.tche.br/pesquisas/2008/artigos/sistemas/328.pdf).

Werneck,V(2006). *Análise e Projeto de Sistemas.* Acessado em:17/10/2009.Disponível em:[**http://www.ime.uerj.br/~vera/projeto/apostila.pdf**](http://www.ime.uerj.br/~vera/projeto/apostila.pdf).

Tokuno, D.B(2004). *Modelos de Ciclo de Vida: Por que precisamos deles no desenvolvimento.* Acessado em:20/10/2009. Disponível em: [**http://imasters.uol.com.br/noticia/1861/gerencia/modelos\_de\_ciclo\_de\_vida\_por\_que\_precisamos\_deles\_no\_desenvolvimento/**](http://imasters.uol.com.br/noticia/1861/gerencia/modelos_de_ciclo_de_vida_por_que_precisamos_deles_no_desenvolvimento/).

Cordeiro, E.S. *Introdução ao ciclo de vida de software.* Acessado em:23/10/2009.Disponível:[**http://www.cordeiro.pro.br/aulas/engenharia/processoDeSoftware/ciclos.pdf**](http://www.cordeiro.pro.br/aulas/engenharia/processoDeSoftware/ciclos.pdf).

Balduino, R.(2007) *Introduction to OpenUP (Open Unified Process)*. Acessado em:03/11/2009. Disponível em: [**http://www.eclipse.org/epf/general/OpenUP.pdf**](http://www.eclipse.org/epf/general/OpenUP.pdf).

Monteiro,J.M e Ybanez,M(2009). *Integrando Metodologias Ágeis e Modelos de Maturidade de Software: Um Estudo de Caso.* Acessado em:05/11/2009. Disponível em:**<http://www.infobrasil.inf.br/iConstructor/Custom/anais2009/Integrando%20Metodologias%20%C3%81geis%20e%20Modelos%20de%20Maturidade%20de%20Software%20Um%20Estudo%20de%20Caso.pdf>**.

Kroll,P. e Royce,W.(2005). *Key principles for business-driven development.*

Acessado em:05/11/2009. Disponível em: [**http://www.ibm.com/developerworks/rational/library/oct05/kroll/**](http://www.ibm.com/developerworks/rational/library/oct05/kroll/).

Moraes,T(2006). *Aplicação de padrões ao processo de desenvolvimento de software RUP.* Acessado em:08/11/2009. Disponível em:http://dsc.upe.br/~tcc/20062/Monografia\_TiagoMoraes.pdf.

Piske,O.T(2003). RUP-Rational Unified Process. Acessado em:16/11/2009.Disponível em:http://www.angusyoung.org/arquivos/artigos/trabalho\_rup.pdf.

*Metodologia*. Acessado em:21/11/2009. Disponível em: [**http://www.architettura.com.br/Architettura1/NossosDiferenciais/Metodologia/Default.aspx**](http://www.architettura.com.br/Architettura1/NossosDiferenciais/Metodologia/Default.aspx).

*Metodologias ágeis X Metodologias tradicionais.* Acessado em:22/11/2009.Disponível em: http://tisimples.wordpress.com/2009/04/18/metodologias-ageis-x-metodologias-tradicionais/.

1. Um conjunto de ações seqüências que objetivam na conclusão de uma meta. [↑](#footnote-ref-1)