



Introdução ao RUP

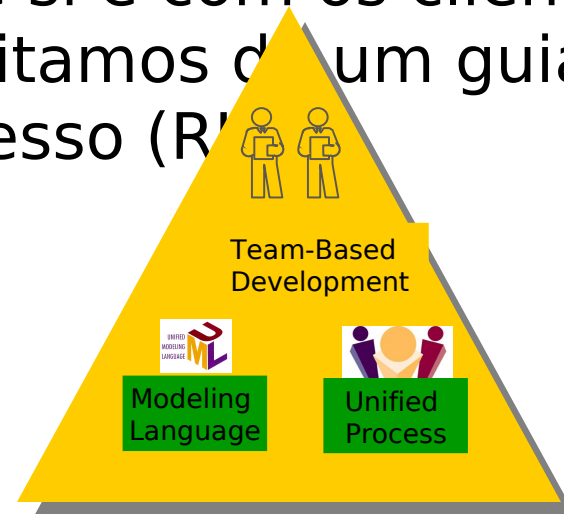
Rational Unified Process



Contexto

- Não é suficiente apenas a presença de desenvolvedores altamente treinados:

Precisamos de uma linguagem para a equipe poder se comunicar entre si e com os clientes (UML), além disso necessitamos de um guia organizacional: Um processo (RUP)



O que é UML (Unified Modeling Language) ?

- A UML (Unified Modeling Language) é o sucessor de um conjunto de métodos de análise e projeto orientados a objeto (OOA&D).
- A UML é um modelo de linguagem, não um método. Um método pressupõe um modelo de linguagem e um processo. O modelo de linguagem é a notação que o método usa para descrever o projeto. O processo são os passos que devem ser seguidos para se construir o projeto.
- O modelo de linguagem corresponde ao ponto principal da comunicação. Se uma pessoa quer conversar sobre o projeto, como outra pessoa, é através do modelo de linguagem que elas se entendem. Nessa hora, o processo não é utilizado.
- A UML define uma notação e um meta-modelo. A notação são todos os elementos de representação gráfica vistos no modelo (retângulo, setas, o texto, etc.), é a sintaxe do modelo de linguagem. Um meta-modelo é um diagrama de classe que define de maneira mais rigorosa a notação.
- A UML (Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software.
- Pode ser empregada para a visualização, especificação, construção e documentação de artefatos que fazem uso de sistemas complexos de software.

“UML é uma linguagem de modelagem, não uma metodologia.”

O que é RUP (Rational Unified Process) ?



- É um processo configurável de Engenharia de Software.
- O RUP é um guia para como usar efetivamente a UML

RUP e CMM (Capability Maturity Model)



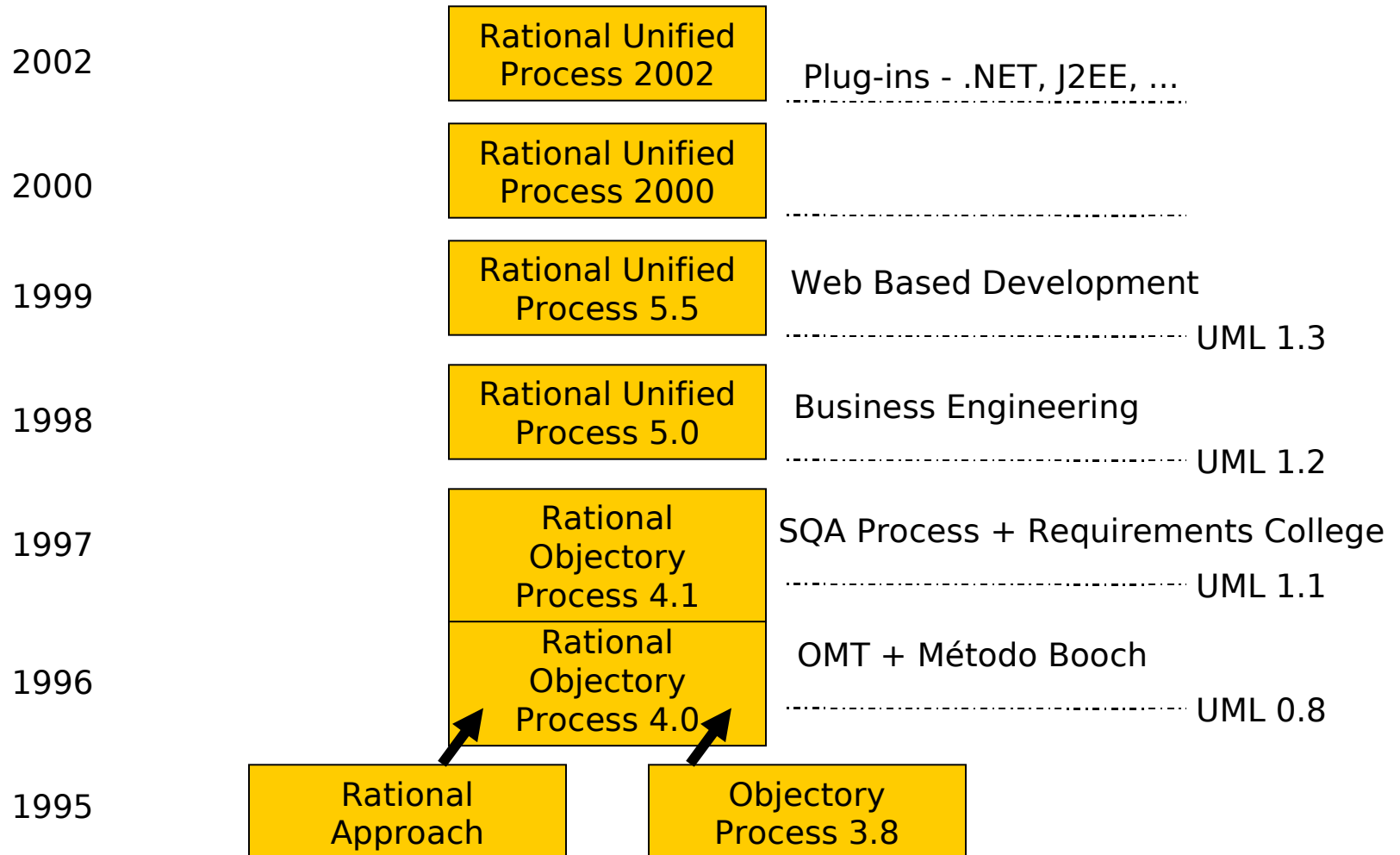
- O objetivo do RUP é assegurar uma produção de alta qualidade de software, que realiza a necessidade do usuário seguindo prazos e o orçamento.
- Com o advento do CMM as organizações focalizam a qualidade em primeiro plano e o RUP pode ser bastante útil quando se quer atingir níveis maiores.

Utilidade



- O RUP, como processo de desenvolvimento de software, tem 4 regras:
 - ◆ servir de guia;
 - ◆ especificar quais artefatos devem ser desenvolvidos e quando devem ser desenvolvidos;
 - ◆ dirigir as tarefas individuais e do time como um todo;
 - ◆ oferecer critérios para monitorar e medir os produtos e atividades do projeto.

Histórico RUP



Processo Unificado



- Dirigido por casos de uso
- Baseado em componentes
- Centrado em arquitetura
- Iterativo e Incremental
- Framework genérico de um processo de desenvolvimento

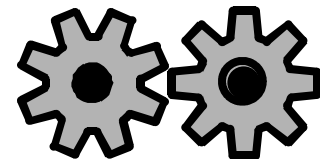
Dirigido por casos de uso



- Os casos de uso são utilizados como o principal recurso para o estabelecimento do comportamento desejado do sistema (especificação) e para sua verificação e validação (testes).

Baseado em componentes

- A característica principal do RUP é ser baseado em componente (parte física e atualizável do sistema), ou seja, o software desenvolvido é formado por componentes de software que se comunicam através de interfaces bem definidas.



Centrado na arquitetura



- No RUP os use cases e a arquitetura vão sendo desenvolvidos em paralelo. O conceito de arquitetura engloba os aspectos mais relevantes, tanto estáticos, como dinâmicos, do sistema.

Iterativo e Incremental

- O RUP utiliza pequenos ciclos de projeto (mini-projetos) que correspondem à uma iteração e que resultam em um incremento no software. Iterações referem-se a passos e incrementos a evoluções do produto.
- UP repete vários ciclos até o término do sistema
- Cada ciclo de vida possui 4 fases:
 - ◆ Concepção,
 - ◆ Elaboração,
 - ◆ Construção, e
 - ◆ Entrega

Fases do Ciclo de Vida




tempo →

- **Concepção** Define o escopo do projeto e sua viabilidade
- **Elaboração** Plano do Projeto, especificação de características e definição de baseline da arquitetura
- **Construção** Construção do produto
- **Transição** Substituição do antigo sistema e implantação do novo

Ciclo de Vida - Concepção


- Fase de compreensão do problema e da tecnologia através da definição dos use cases mais críticos. No final desta fase deve-se ter definido o escopo do produto e ter demonstrado que o projeto é viável do ponto de vista do negócio da organização.
- Após essa fase temos noção de quanto o sistema custará e quanto ele trará de retorno. Teremos também o plano de negócios definido.
 - ◆ O que o sistema deve fazer ?
 - ◆ Qual poderia ser a sua arquitetura ?
 - ◆ Qual o prazo e custo do desenvolvimento ?

Ciclo de Vida - Elaboração



- Fase de descrição da arquitetura do software na qual os requisitos que mais impactam na arquitetura são capturados em forma de use cases.
- Identificação dos riscos do projeto (requisitos, tecnológicos, especialização e políticos)
- No final da fase de elaboração deve ser possível estimar custos, elaborar o cronograma e o plano de construção do sistema.

Ciclo de Vida - Construção



- Fase na qual o software é construído e preparado para a transição para os usuários. Além do código, propriamente dito, também são produzidos os casos de teste e a documentação.

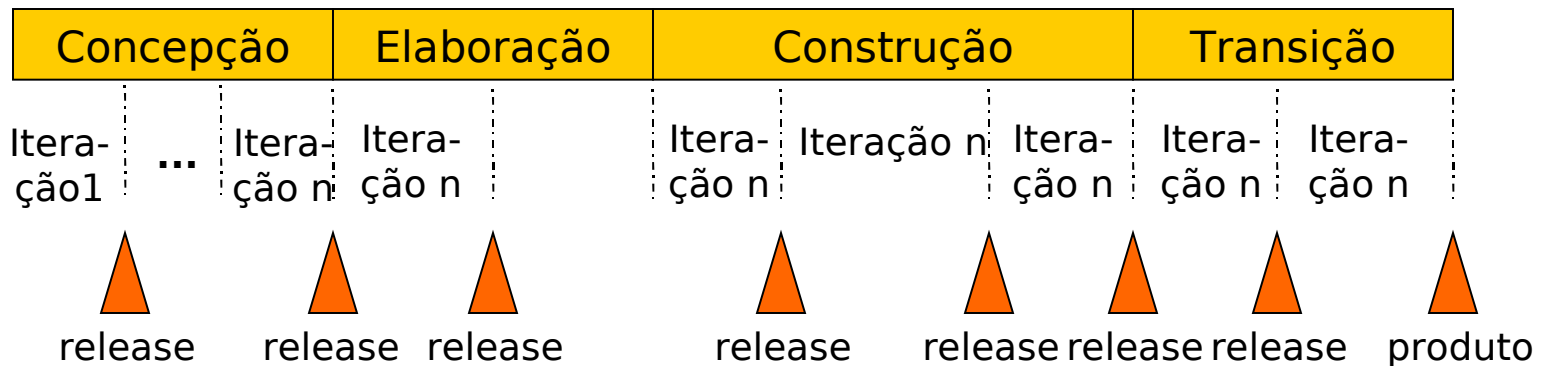
Ciclo de Vida - Transição



- É a transição entre o sistema antigo e o novo.
- Fase de treinamento dos usuários e transição do produto para utilização.
- Ao final desta fase devemos ter:
 - ◆ Manuais revisados
 - ◆ Sistema de Informação implantado e monitorado

Iterativo e Incremental

- Cada fase possui iterações



- Cada iteração gera um artefato ou um conjunto deles (releases)

Qualquer tipo de informação criada, modificada ou usada por um processo.

Iteração e Workflow

Passos dentro de uma iteração

Core Workflows

Requisito

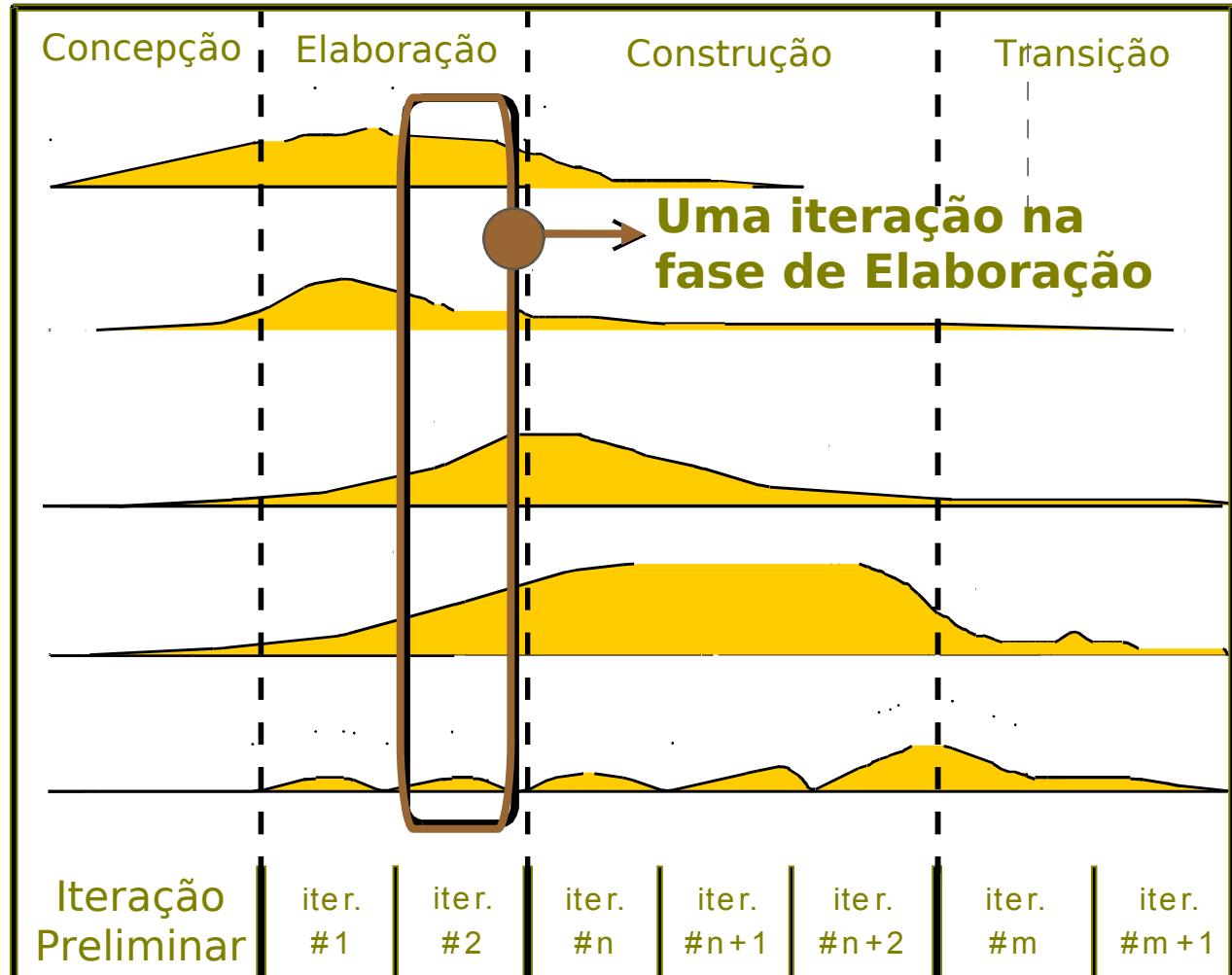
Análise

Desenho

Implementação

Teste

Fases



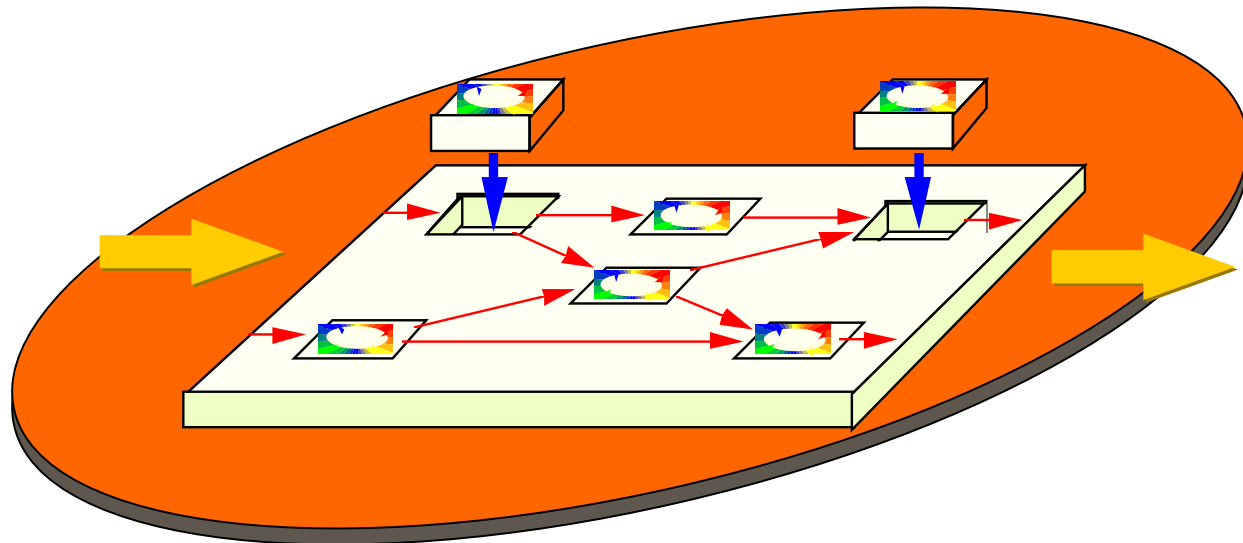
Iterativo e Incremental

- Não é tão eficaz entender todo o problema, desenhar toda a solução, e construir o sistema e então testar o produto em separado.

Um processo iterativo é requerido para permitir um entendimento crescente de todo o problema através de sucessivos refinamentos.

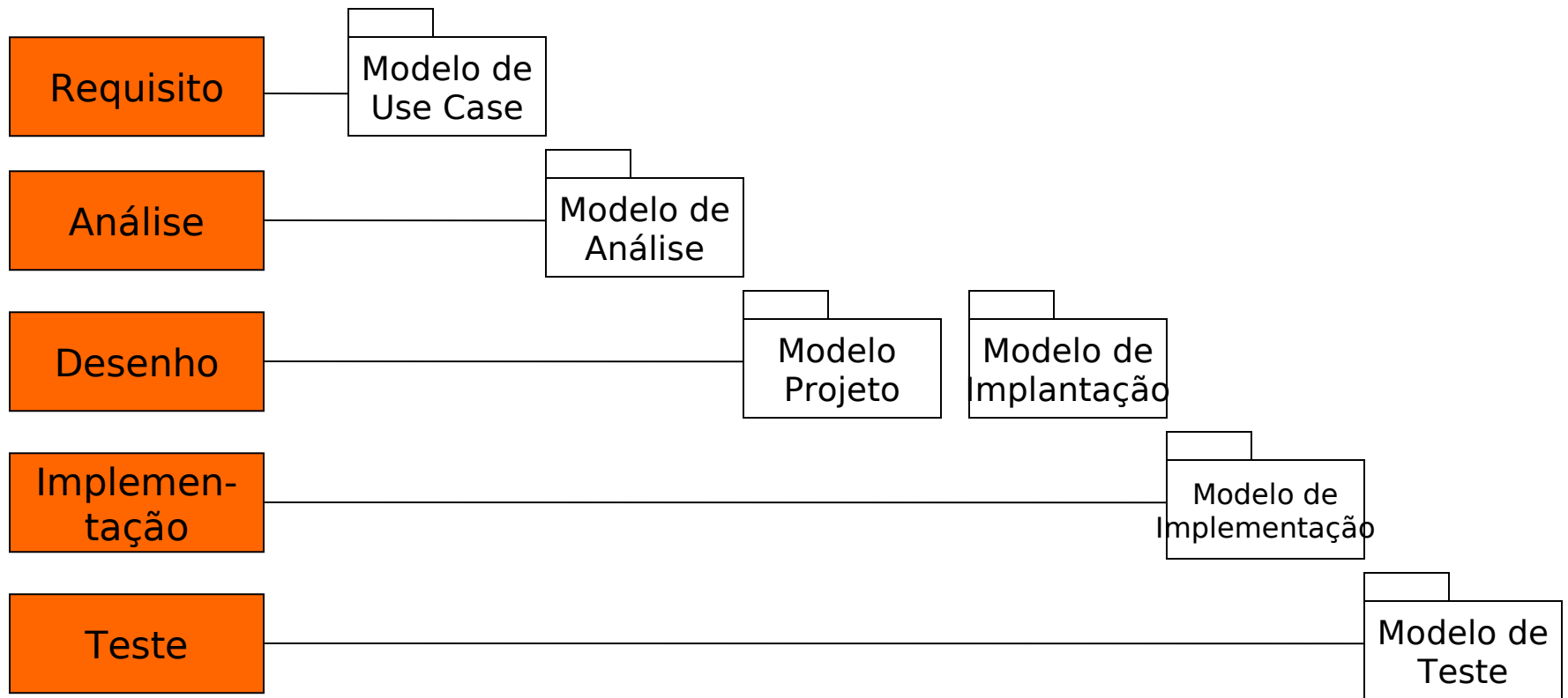


Framework



- O UP é desenhado para ser flexível e extensível
- Não existe Processo Universal

Modelos e Workflow




Atividades ou Workflows



- São atividades conduzidas em todas as fases de um ciclo, variando de intensidade conforme a fase.
- Dão origem aos artefatos do projeto.
- Em cada fase são desenvolvidas várias atividades do processo
 - ◆ modelagem de negócio
 - ◆ levantamento de requisitos
 - ◆ análise e projeto
 - ◆ implementação
 - ◆ testes...

Concepção - Produtos




- Modelo de negócio, que estabelece o contexto do sistema.
- Primeira versão simplificada do Modelo de Caso de Usos, Modelo de Análise e Modelo de Design; opcionalmente um esboço rudimentar dos Modelos de Implementação e Teste.
- Proposta inicial de uma descrição de arquitetura com base nas visões dos modelos descritos acima.
- Protótipo de prova-de-conceito, ou storyboards.
- Lista de riscos

Concepção - Workflows




- Requisitos: capturar os requisitos mais críticos (na forma de casos de uso) e definir o escopo do sistema.
- Análise: analisar os requisitos e montar uma proposta para o modelo de classes e objetos, com foco nas classes de negócio, mais o glossário.
- Desenho: preparar o Modelo de Design ou storyboard, apresentando um rascunho preliminar da arquitetura do sistema: identificar os primeiros componentes, interfaces e subsistemas, assim como o Modelo de Implantação.
- Implementação: pode ser necessário criar um protótipo descartável para demonstrar o caminho escolhido.
- Testes: criar primeiros esboços de teste com base nas informações já adquiridas.

Elaboração - Produtos




- Modelo de negócio completo descrevendo todo o contexto do sistema.
- Novas versões dos modelos: Casos de Uso, Análise, Design, Implementação e Implantação.
- Versão preliminar do manual de usuário (opcional).
- Previsão do custo de implementação.
- Cronograma de implementação.

Elaboração - Workflows




- Requisitos: encontrar, priorizar, detalhar e estruturar os Casos de Uso, capturando aproximadamente 80% dos requisitos.
- Análise: detalhar as classes de negócio, fazer o particionamento em pacotes, atualizar o glossário e refinar os Casos de Uso.
- Design: fazer o design dos Casos de Uso, classes e subsistemas para estabelecer uma estrutura básica do sistema. Pacotes de análise e subsistemas de design, são importantes. São considerados: sistema operacional, linguagem, banco de dados, distribuição de objetos, etc..
- Implementação: implementar e testar os componentes arquiteturalmente significantes. Eventualmente criar protótipos para testar alguma nova tecnologia.
- Testes: planejar e especificar os testes, definindo casos de teste e rotinas de teste.

Construção - Produtos




- Primeira versão do manual de usuário, juntamente com o help, manual de operação e instalação deverão ser esboçados. Material para curso e treinamento.
- Versão beta do sistema é disponibilizada.

Construção - Workflows




- Requisitos: capturar os requisitos remanescentes, refinando Casos de Uso e cenários
- Análise: capturar algum detalhe que passou despercebido nas classes pertinentes ao negócio.
- Design: refinar os casos de uso e cenários remanescentes com base na tecnologia utilizada.
- Implementação: codificar e integrar componentes, priorizando os casos de uso mais importantes.
- Testes: testar funcionalidades e performance do sistema. Se necessário testar novos casos e rotinas de teste.

Distribuição - Objetivos



- Verificar se os requisitos foram implementados corretamente
- Corrigir os problemas identificados pelos usuários na versão beta e tomar as ações necessárias para colocar o sistema em produção.
- Produzir e/ou atualizar:
 - ◆ Programas executáveis
 - ◆ Documentação legal (contrato, garantia...)
 - ◆ Especificação técnica
 - ◆ Manual do sistema, de administração do sistema e material de treinamento

Distribuição - Workflow



- Requisitos: eventual correção da documentação devido a bugs encontrados no sistema.
- Análise: eventual correção do modelo de análise devido a bugs encontrados no sistema.
- Design: eventual correção do modelo de design devido a bugs encontrados no sistema.
- Implementação: eventual correção do código devido a bugs encontrados no sistema.

FIM

RUP

Rational Unified Process

