

Disciplina

Engenharia de Software

Professor: Gilmar Luiz de Borba

2014 - 1

Conteúdo Programático

1-Conceitos

Engenharia de software, Ciência da Computação, Engenharia de Sistemas. Custos, desafios, principais atributos de um bom software. (SOMMERVILLE, 2001:2-17).

2- Modelos de Processos de Software (ou Modelos de ciclo de vida)

Modelo Codifica-Remenda, Modelo em Cascata, Modelo em Espiral, Modelo de Prototipagem evolutiva, Modelo de Entrega Evolutiva, Modelo Dirigido por Prazo, Modelo Dirigido por Ferramenta. (SOMMERVILLE, 2001:18-37) e (PÁDUA, 2005:11-15).

3- Processos de Software

Conceitos, UP (Unified Process), EUP (Enterprise Unified Process), PSP (Personal Software Process), TSP (Team Software Process), PRAXIS (PRocesso para Aplicativos eXtensíveis Interativos), ICONIX (ICONIX Software Engineering), XP (Extreme Programming), RUP (Rational Unified Process). (PÁDUA, 2005:16-51).

4- Rational Unified Process (RUP)

Modelo Dirigido por Ferramenta. Estrutura, Arquitetura, fluxos (disciplinas), fases, gráfico da baleia. Para cada fase e cada disciplina: conhecer quais são os trabalhadores envolvidos; quais são as atividades de cada trabalhador (papel); quais são os artefatos produzidos; entender como acessar a documentação do RUP e como esta é mantida. Kruchten, (2003).

5- Prática (RUP)

Desenvolvimento de um projeto de software (Laboratório) a partir das práticas do RUP. Entender a evolução do processo (RUP) no contexto de um projeto, aplicando a UML. Entender quais diagramas são aplicados em cada fase/disciplina e como estes se relacionam.

6-Modelagem Ágil de Software e Métodos ágeis: comparativo com o processo tradicional

Desenvolvimento Ágil, Extreme Programming, Valores, Princípios e Práticas. Discussão participativa em sala de aula. (SOMMERVILLE, 2001:38-51).

Bibliografia básica

FILHO, Wilson de Pádua. **Engenharia de Software – Fundamentos, Métodos e Padrões**. LTC, 2001.

KRUCHTEN, Philippe. **Introdução ao RUP - Rational Unified Process**. Rio de Janeiro: Ciência Moderna, 2003.

PRESSMAN, Roger S. **Engenharia de Software**. 6rd ed. McGraw-Hill, 2002.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª. Ed. São Paulo. Pearson, Prentice Hall, 2011.

Bibliografia complementar

MACHADO, Felipe Rodrigues Nery. **Tecnologia e Projeto de Data Warehouse** – São Paulo SP: Érica: 2004.

OLIVEIRA, Wilson José. **Data Warehouse** – Florianópolis SC: Visual Books: junho de 2002.

Introdução à Engenharia de Software

Engenharia de software

É uma disciplina de engenharia cujo o foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até a sua manutenção, quando o sistema já está sendo usado. *(SOMMERVILLE, 2011:5).*

Logo, percebemos que a Engenharia de Software envolve:

Os processos de software, a engenharia de requisitos, a modelagem de sistemas, os testes de software, o reuso de software, os componentes de software, a arquitetura, o gerenciamento de projetos, a qualidade de software, a melhoria dos processos etc.

Introdução à Engenharia de Software

Ciência da Computação x Engenharia de software

Ciência da computação foca a teoria e os fundamentos; a engenharia de software preocupa-se com o lado prático do desenvolvimento e entrega de software úteis. (*SOMMERVILLE, 2011:4*).

Introdução à Engenharia de Software

Engenharia de Sistemas x Engenharia de software

Engenharia de sistemas se preocupa com todos os aspectos do desenvolvimento de sistemas computacionais, incluindo engenharia de hardware, software e processo. Engenharia de software é uma parte mais específica desse processo mais genérico. (*SOMMERVILLE, 2011:4*).

Introdução à Engenharia de Software

Custos

Aproximadamente 60% dos custos de software são de desenvolvimento; 40% são custos de testes. Para software customizado, os custos de evolução freqüentemente superam os custos de desenvolvimento. (*SOMMERVILLE, 2011:4*).

Desafios

Lidar com o aumento de diversidade, demandas pela diminuição do tempo para a entrega e desenvolvimento de software confiável. (*SOMMERVILLE, 2011:4*).

Introdução à Engenharia de Software

Principais atributos de um bom software.

-Manutenibilidade

O software deve ser escrito de forma que possa evoluir para atender às necessidades dos clientes. Esse é um atributo crítico, porque a mudança de software é um requisito inevitável de um ambiente de negócio em mudança.

Confiança e Proteção

[...] Um software confiável não deve causar prejuízos físicos ou econômicos nos casos de falha de sistema. Usuários maliciosos não devem ser capazes de acessar ou prejudicar o sistema.

Eficiência

O software não deve desperdiçar os recursos do sistema, como memória e ciclos do processador. Portanto, eficiência inclui capacidade de resposta, tempo de processamento, uso de memória etc.

Aceitabilidade

O software deve ser aceitável para o tipo de usuário para o qual foi projetado. Isso significa que deve ser compreensível, usável e compatível com outros sistemas usados por ele.

(SOMMERVILLE, 2011:5).

O que é um Processo?

Um processo é um conjunto de passos particularmente ordenados, construídos por atividades, métodos, práticas e transformações, usado para atingir uma meta. Esta meta geralmente está associada a um ou mais resultados concretos finais, que são os produtos da execução do processo. (PÁDUA, 2005,11).

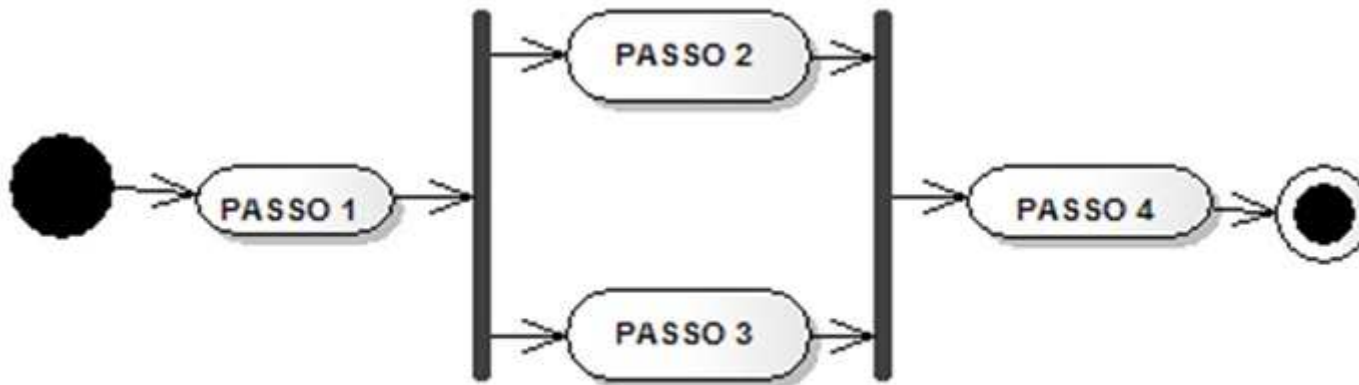
Processo

Na engenharia de software os processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contratação de software.

(PÁDUA, 2005,11).

Processos de desenvolvimento de software

Podem existir passos em paralelos em um processo.



Fonte: (PÁDUA, 2003, Página 11)

Processos de desenvolvimento de software

Todo processo deve possuir uma arquitetura bem definida e os passos iniciais pra definir essa arquitetura é o modelo do ciclo de vida.

Processos de desenvolvimento de software

A norma que estabelece uma estrutura comum para os processo de ciclo de vida de software é a ISO/IEC **12207**.

Algumas características:

- Estabelece uma arquitetura de alto nível do ciclo de vida de software;
- Descreve um conjunto de processos e seus inter-relacionamentos;
- Não possui nenhuma ligação com métodos, ferramentas, treinamentos, métricas ou tecnologias empregadas (*independência de cultura organizacional*);
- Flexibilidade, define "o que fazer" e não "como fazer";
- Modularidade são fortemente coesos e fracamente acoplados.

Processos de desenvolvimento de software

Uma evolução da ISO/IEC **12207** é a norma ISO/IEC **15504** que define os níveis de capacidade para cada processo assim como o CMMI. A ISO/IEC 15504 (conhecida como SPICE) define processo de desenvolvimento de software.

Características e Objetivos:

- Realização de avaliações de processos de software com o foco da melhoria dos processos;
- Identificar os pontos fracos e fortes, que serão utilizados para a elaboração de um plano de melhorias do processo;
- Determinar a capacidade dos processos viabilizando a avaliação de um fornecedor em potencial;
- Identificar e descrever um conjunto de processos considerados universais e fundamentais para a boa prática da engenharia de software;
- Definir os níveis de capacidade a serem utilizados para avaliar uma organização com relação a realização de determinado processo (e propor melhoria);
- Definir um guia para a orientação da melhoria de processo;

SPICE : Software Process Improvement and Capability Determination.

CMMI: Capability Maturity Model.

Processos de desenvolvimento de software

ISO/IEC 15504

Grupos de Processos definidos na ISO 15504:

- Cliente-Fornecedor;
- Engenharia;
- Suporte;
- Gerência;
- Organização.

Modelos de Ciclo de Vida de Processos de Desenvolvimento de Software.

Segundo BEZERRA (2007), o desenvolvimento de software é uma atividade complexa. Essa complexidade corresponde à sobreposição das complexidades relativas ao desenvolvimento dos seus diversos componentes: software, hardware, procedimentos etc. Isto reflete no alto número de projetos de software que não chegam ao final, ou que extrapolam recursos de tempo e dinheiro alocados. (BEZERRA, 2007:21).

Porque usar modelos de processos de desenvolvimento de software?

Dados levantados no Chaos Report (1994) destacam os vários problemas trazidos por esta complexidade. (CHAOS apud BEZERRA,2007:21)

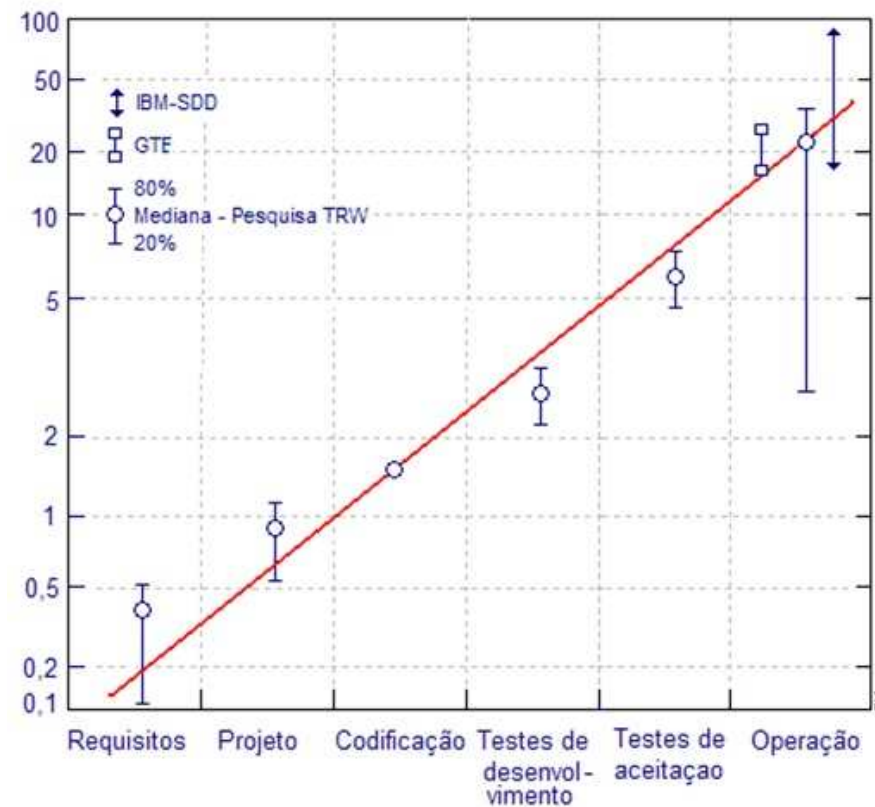
- . Projetos que terminam dentro do prazo estimado: 10%.
- . Projetos descontinuados antes de chegarem ao fim: 25%.
- . Projetos acima do custo esperado: 60%.
- . Atraso médio dos projetos: um ano.

Porque usar Processos de Desenvolvimento de Software?

- . Acompanhar o desenvolvimento de software.
- . Gerenciar requisitos.
- . Usar uma arquitetura adequada de desenvolvimento.
- . Usar modelos para desenvolver software.
- . Controlar continuamente a qualidade do software.
- . Acompanhar as mudanças.

Processos de Desenvolvimento de Software

Custo relativo para corrigir um erro durante o desenvolvimento do sistema

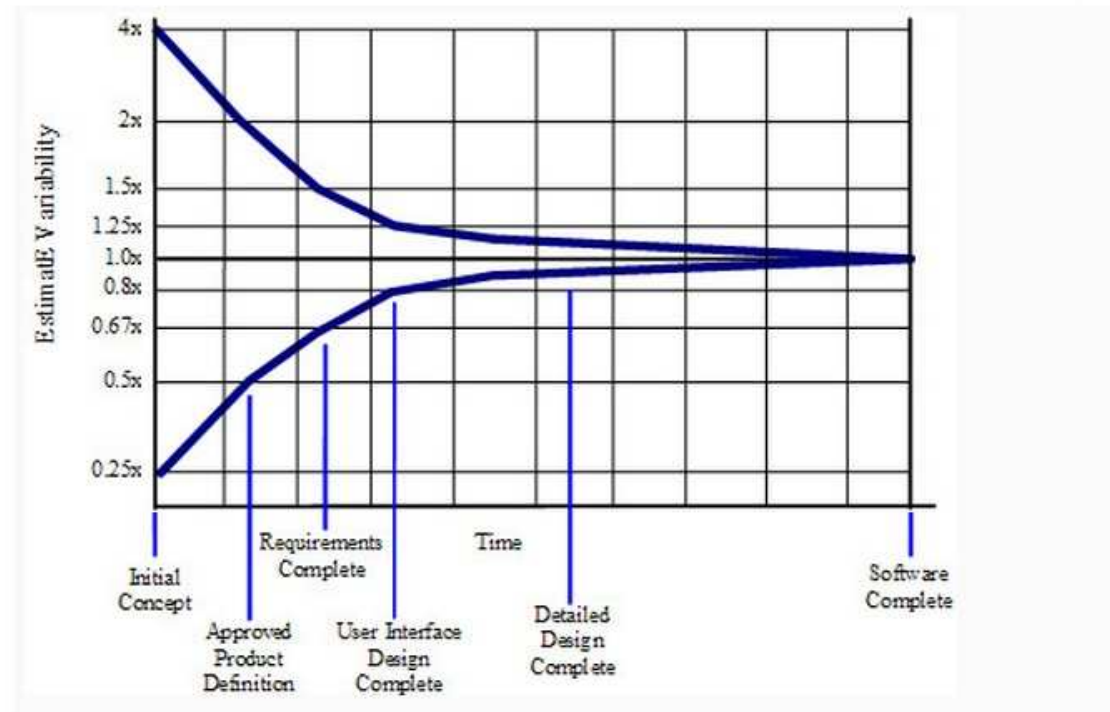


Fonte: (Gane, 1983, página 7)

Referências cruzadas: Barry W. Boehm, Engenheiro de Software americano - Departamento de Ciência da Computação da Universidade do Sul da Califórnia.

Processos de Desenvolvimento de Software

Cone da Incerteza



The Cone Of Uncertainty - Construx - Software Development Best Practices
 Disponível em: <http://www.construx.com/Page.aspx?cid=1648>.
 Acessado em: Dezembro de 2012.

Referências cruzadas: Barry W. Boehm, Engenheiro de Software americano - Departamento de Ciência da Computação da Universidade do Sul da Califórnia.

Modelos de ciclo de vida de processos de desenvolvimento de software

- Modelo Codifica-remenda.
- Modelo em Cascata.
- Modelo em Espiral.
- Modelo de Prototipagem evolutiva.
- Modelo de Entrega evolutiva.
- Modelo dirigido por prazo.
- Modelo dirigido por ferramenta.

Por que falar de Modelos de ciclo de vida?

Porque falar de Modelos de ciclo de vida?

O ponto de partida para a arquitetura de processo de software é a escolha de um modelo de um ciclo de vida.

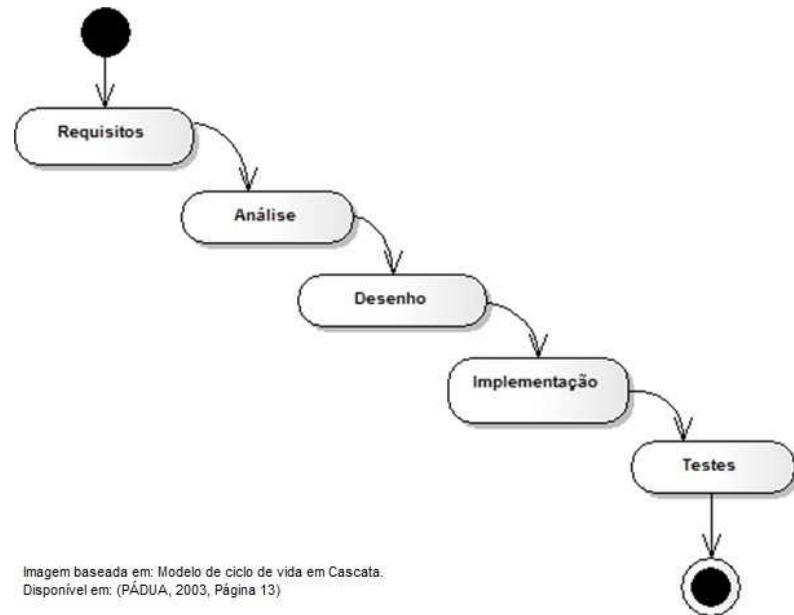
Modelo de ciclo de vida codifica-remenda.



Imagem baseada em: Modelo de ciclo de vida Codifica-Remenda.
Disponível em: (PÁDUA, 2003, Página 12)

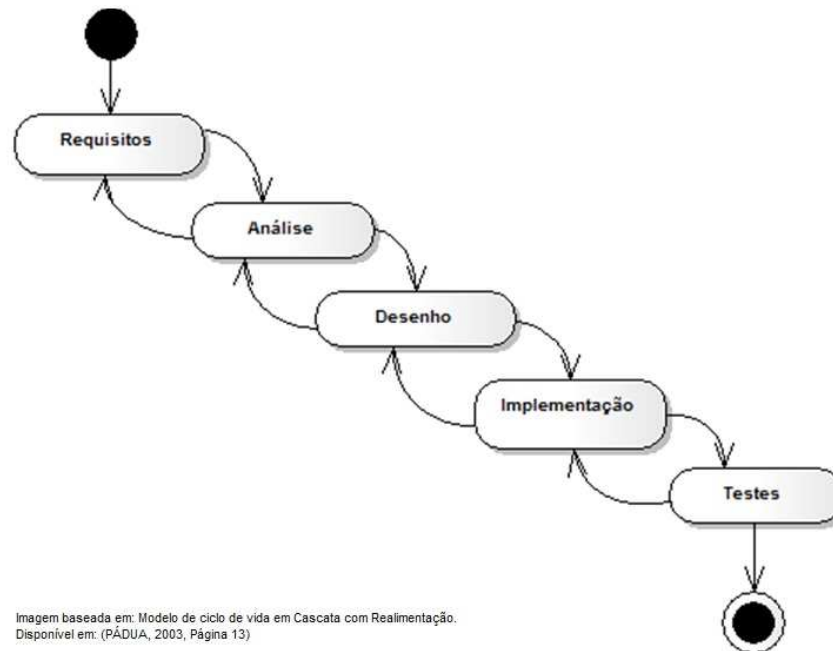
- Caótico.
- Normalmente não há processo formal definido.
- Os erros aparecem e são corrigidos.
- Não há muito critério.
- Pouco ou nenhum controle gerencial e técnico.

Modelo de ciclo de vida em cascata.



- Sequência pré-definida dos processos.
- Marcos.
- Rígido, lento e burocrático.
- Pouca transparência.

Modelo de ciclo de vida em cascata com realimentação.



- Seqüência pré-definida dos processos.
- Marcos.
- Revisão e alteração de resultados das fases anteriores.
- Modelo mais realista.

Modelo de ciclo de vida em espiral.

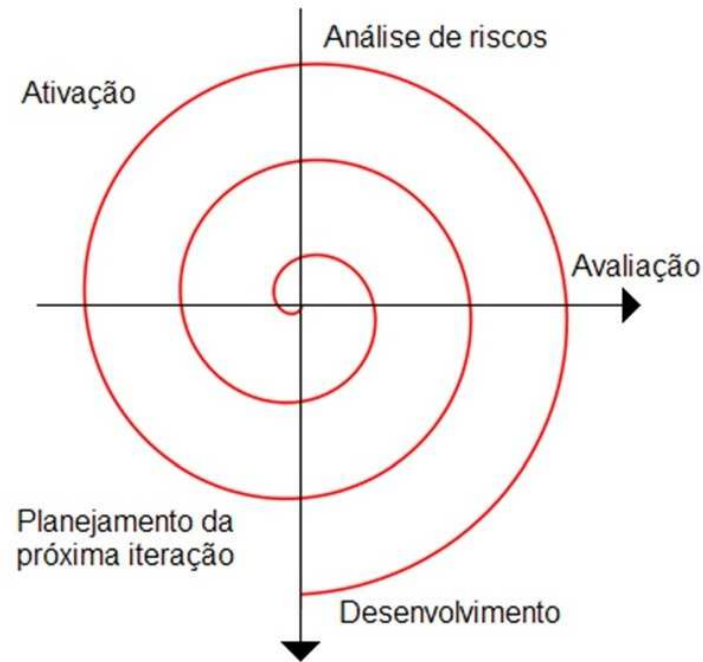


Imagem baseada em: Modelo de ciclo de vida em Espiral
Disponível em: (PÁDUA, 2003, Página 14)

GLB

- Iterações sucessivas.
- Cada iteração uma volta na espiral.
- Cada volta, novos dados para a nova iteração.
- Desenvolvimento de produtos a curto prazo.
- Dificuldade de gerenciamento do projeto.

Modelo de ciclo de vida prototipagem evolutiva.

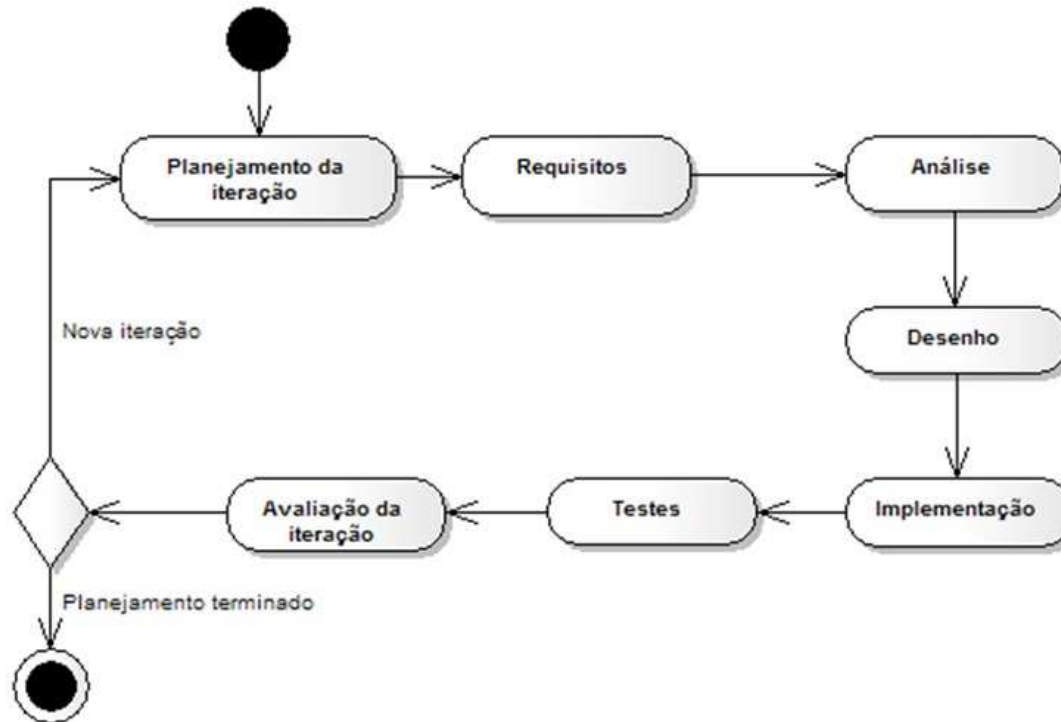
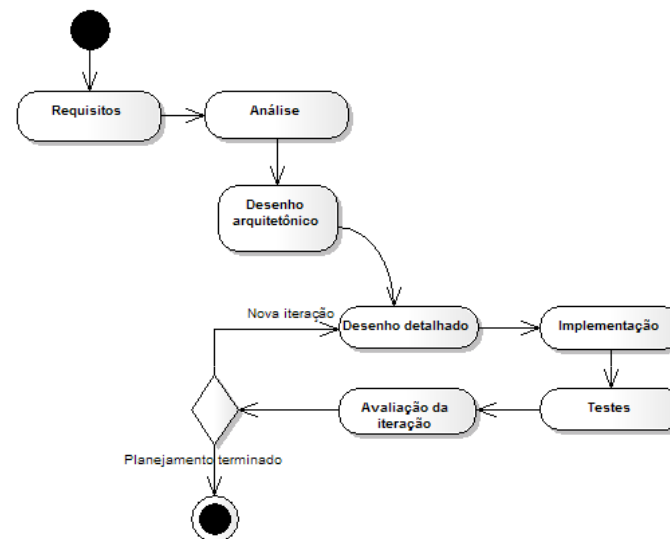


Imagem baseada em: Modelo de ciclo de Prototipagem Evolutiva
Disponível em: (PÁDUA, 2003, Página 14)

- Construção de uma série de versões (protótipos).
- Os protótipos cobrem novos requisitos.
- Flexibilidade.
- Iteratividade.
- Dificuldade de gerenciamento do projeto.

Modelo de ciclo de vida da entrega evolutiva.



- Características dos modelos em cascata e prototipagem evolutiva.
- Facilidade de avaliação do produto.
- Facilidade de gerenciamento do projeto.
- Dificuldade de manter a integridade do produto nas liberações parciais.

Modelo dirigido por prazo.

- O que se consegue fazer dentro de um prazo determinará o produto.
- Os prazos são determinados de acordo com os interesses.

Na prática, os prazos costumam ser definidos de forma política, e o “produto” que se entrega no final do prazo é apenas um resultado parcial. Ele será completado aos trancos e barrancos em desenvolvimento posterior, disfarçado de “manutenção”. (PÁDUA, 2005:15).

Modelo dirigido por ferramenta.

- Impõem processos rígidos adequados à ferramenta escolhida.
- A qualidade desses processos dependerá da qualidade da ferramenta.

Exemplos de Processos

- . **UP** (Unified Process)
- . **EUP** (Enterprise Unified Process)
- . **PSP** (Personal Software Process)
- . **TSP** (Team Software Process)
- . **PRAXIS** (PRocesso para Aplicativos eXtensíveis Interativos)
- . **ICONIX** (ICONIX Software Engeneering)
- . **XP** (Extreme Programming)
- . **RUP** (Rational Unified Process)

Exemplos de Processos

UP (Unified Process)

- Usa a UML.
- É dirigido por caso de uso.
- É centrado em uma arquitetura.
- É incremental e iterativo.
- Usa o modelo em espiral.
- É dividido no tempo a partir de fases.
- É dividido por fluxos (ou disciplinas).

Exemplos de Processos

UP (Unified Process)

Fases:

Concepção, Elaboração, Construção e Transição.

Fluxos:

Requisitos, Análise, Desenho, Implementação e testes.

Exemplos de Processos

EUP (Enterprise Unified Process)

É uma extensão do processo unificado, possui uma fase adicional de produção (produto em operação e manutenção) e a partir desta, permite implementar novos fluxos.

Exemplos de Processos

PSP (Personal Software Process)

Segundo PÁDUA (2005), é um processo baseado na necessidade dos desenvolvedores individuais para atingir os níveis superiores de maturidade no desenvolvimento de software (CMM). É baseado em projetos individuais com duração típica de cerca de 10 horas. (PADUA, 2005:16).

Exemplos de Processos

PSP (Personal Software Process)

- Incentiva a prática individual.
- Projetos com duração de aproximadamente 10 horas.
- Objetiva melhorar a previsibilidade e a qualidade.
- Permite melhorar a qualidade do ciclo de vida dos produtos.
- Focado na maturidade dos processos.

Exemplos de Processos

TSP (Team Software Process)

- Focado para a formação de equipes.
- Baseado no modelo em espiral.
- Três ciclos realizados em 15 dias.
- Direcionado em métricas e inspeções.
- Objetivos: criatividade, gerir planos e reduzir custos.

Exemplos de Processos

PRAXIS

(**PR**ocesso para **A**plicativos **eX**tensíveis **I**nterativo**S**)

Segundo PÁDUA (2005), o PRAXIS foi

projetado para suportar projetos realizados individualmente ou por pequenas equipes com duração de seis meses a um ano. Com isso pretende-se que ele seja utilizável para projetos de fim de curso de graduação ou similares, ou projetos de aplicação de disciplinas da engenharia de software. (PÁDUA, 2005:21).

Exemplos de Processos

PRAXIS

(**PR**ocesso para **Ap**licativos e**X**tensíveis **I**nterativo**S**)

- Projetado para fins didáticos.
- Suporta projetos individuais ou por pequenas equipes.
- Projetos com duração de seis meses a um ano.
- Abrange a área técnica e a área gerencial.

Exemplos de Processos

PRAXIS

(**PR**ocesso para **A**plicativos e **X**tensíveis **I**nterativo**S**)

- . Usa tecnologia orientada a objetos.
- . Usa notação de análise e desenho baseada na UML.
- . Usa padrões baseados no IEEE.
- . Reflete elementos do Processo Unificado.
- . Recebeu influências dos processos PSP e TST.
- . Abrange métodos técnicos.
- . Abrange métodos gerenciais.
- . Assim como o UP, Propõe um ciclo de vida composto por fases.

Exemplos de Processos

ICONIX (ICONIX Software Engeneering)

- . Não é tão burocrático quanto o RUP.
- . É um processo relativamente simples.
- . Usa a UML.
- . Usa obrigatoriamente rastreabilidade de requisitos.

Exemplos de Processos

XP (Extreme Programming)

Manifesto para desenvolvimento ágil de software, valores:

- . Comunicação.
- . Simplicidade.
- . Feedback.
- . Coragem.

Exemplos de Processos

XP (Extreme Programming)

Algumas práticas

- . Programação em pares (pair programming).
- . Reuniões em pé (stand-up meeting).
- . Desenvolvimento orientado a testes (test driven development).
- . Padrões de codificação (coding standards).
- . Pequenas versões (small releases).
- . Projeto simples (simple design).
- . Time coeso (whole team).
- . Testes de aceitação (customer tests).
- . Ritmo sustentável (sustainable pace).
- . Refatoração (refactoring).
- . Integração contínua (continuous integration).

Processos de Desenvolvimento de Software

RUP (Rational Unified Process)

Rational Unified Proccess é um processo de engenharia de software. Ele fornece uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Seu objetivo é assegurar a produção de software de alta qualidade que satisfaça as necessidades de seus usuários finais dentro de prazo e orçamento previsíveis. (KRUCHTEN, 2003: 15).

O Rational Unified Proccess como um produto

“Processos de software são softwares, também. “
(OSTERWEIL apud KRUCHTEN, 2003: 16).

O Rational Unified Process como um produto

O RUP é projetado, desenvolvido, entregue e mantido como qualquer ferramenta de software, o RUP mantém muitas características de um produto:

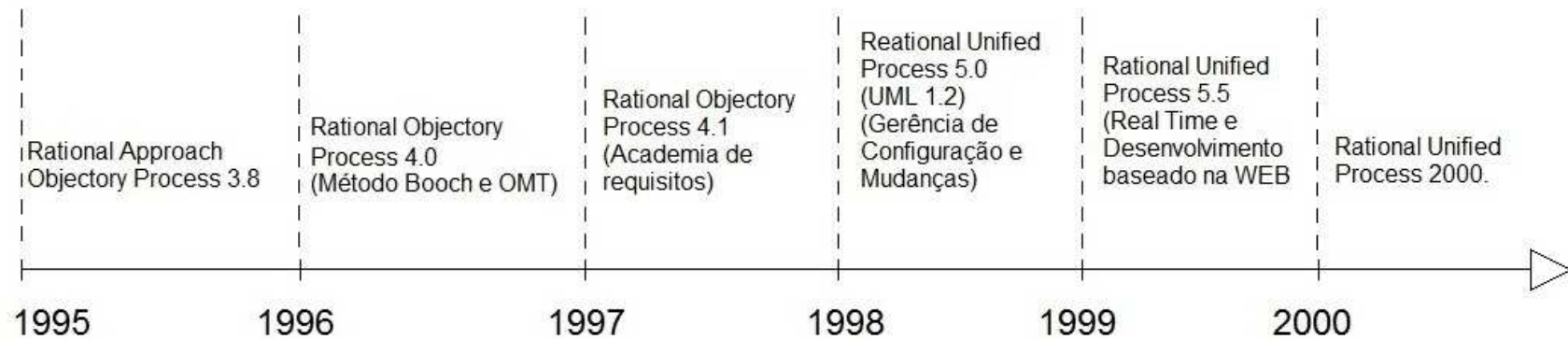
- . Atualizações regulares.
- . Entrega on-line.
- . Adaptabilidade.
- . Integração.

Processos de Desenvolvimento de Software

Benefícios desta abordagem

- . O processo nunca ficará obsoleto.
- . Acesso a versões do processo.
- . Facilidade de acesso a especificações do processo.
- . Uso de hiperlinks.
- . Facilidade de melhorias.
- . Facilidade de gerenciamento.

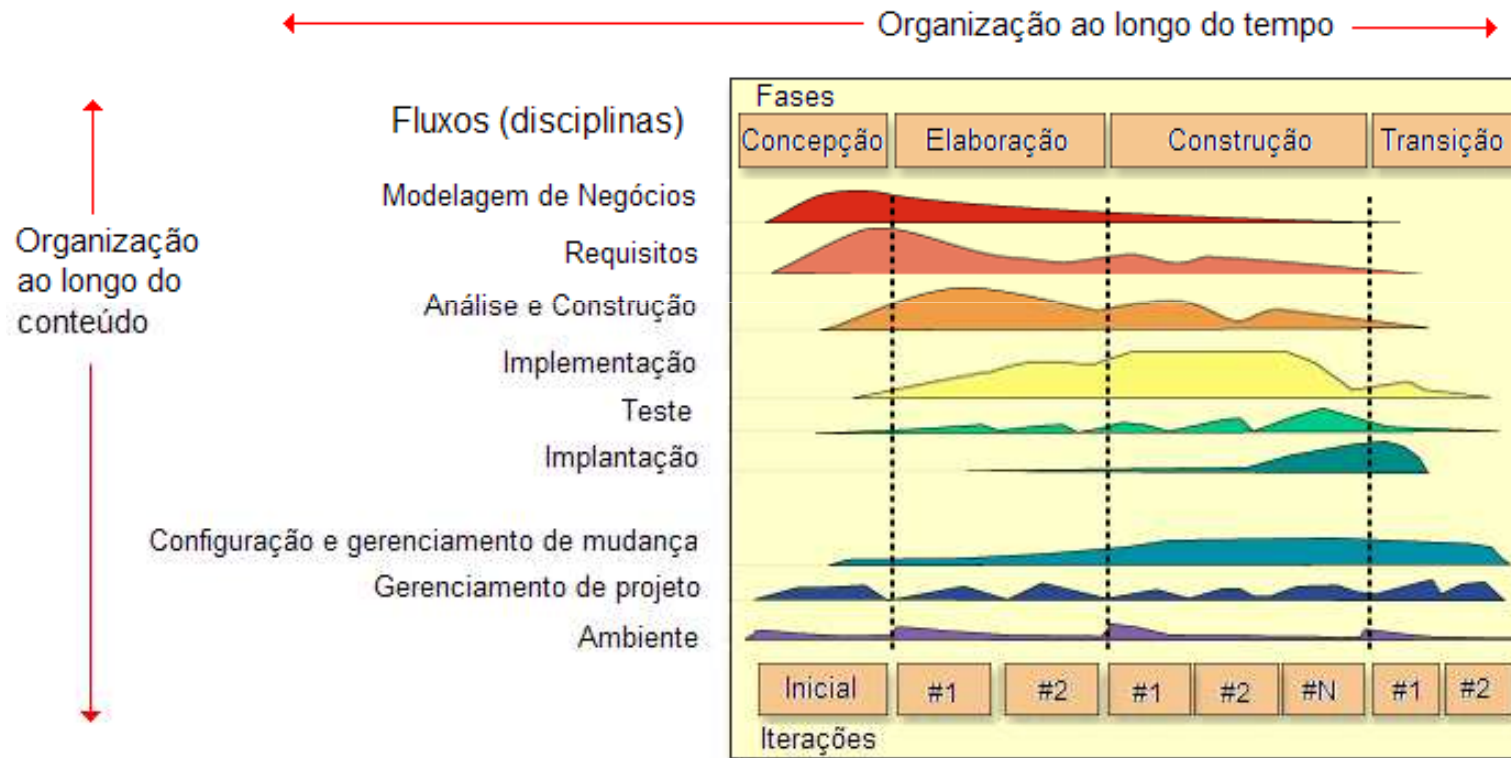
Processos de Desenvolvimento de Software



Referência: Figura baseada na Genealogia do Rational Unified Process.
(Kruchten, 2003, Página 27)

GLB

Estrutura do processo



Práticas e Características

- . Desenvolvimento iterativo.
 - . Gerenciamento de requisitos.
 - . Arquitetura e uso de componentes.
 - . Modelagem baseada na UML.
 - . Qualidade de processo e produto.
 - . Configuração e gerenciamento de mudança.
-
- . Desenvolvimento orientado a caso de uso.
 - . Configuração de processos.
 - . Suporte a ferramentas.

(Práticas)

Desenvolvimento iterativo.

- Leva em conta as mudanças de requisitos.
- Gerenciamento de riscos.
- Gerenciamento de mudanças.
- Correções a cada iteração.

(Práticas)

Desenvolvimento iterativo.

Em um desenvolvimento não-iterativo, as mesmas pessoas estariam esperando ao redor para começar o trabalho delas, fazendo plano depois de plano mas não tendo progresso concreto.
(KRUCTEN, 2003:21).

(Práticas)

Gerenciamento de requisitos.

- Controlar projetos complexos.
- Aumentar a qualidade.
- Reduzir custos.
- Melhorar a comunicação.

Arquitetura e uso de componentes

(Práticas)

Arquitetura e uso de componentes

O que é um componente de software?

Um componente de software pode ser definido com um pedaço não-específico de software, um módulo, um pacote ou um sistema que cumpre uma função clara, tem um limite claro e pode ser integrado uma arquitetura bem definida. (KRUCTEN, 2003:22).

(Práticas)

Arquitetura e uso de componentes

O que é um componente de software?

Um componente de software pode ser definido com um pedaço não-específico de software, um módulo, um pacote ou um sistema que cumpre uma função clara, tem um limite claro e pode ser integrado uma arquitetura bem definida. (KRUCTEN, 2003:22).

Segundo BOOCH (2006) é uma parte física e substituível de um sistema com o qual está em conformidade e proporciona a realização de um conjunto de interfaces. (BOOCH et al, 2006:453).

(Práticas)

Modelagem baseada na UML

Um modelo é uma simplificação da realidade que nos ajuda a dominar um sistema grande e complexo que não pode ser compreendido na sua totalidade. (KRUCTEN, 2003:23).

(Práticas)

Qualidade de processo e produto

- Qualidade do próprio software
- Grau de aceitação do processo, planejamento, testes etc.

(Práticas)

Configuração e gerenciamento de mudança

- Manter em foco os defeitos e mal-entendidos.
- Bases para gerenciamento de métricas de software.

(Características)

Desenvolvimento orientado a caso de uso

- Casos de uso atravessam todo o processo.
- Ligação entre os requisitos e outros artefatos.
- Usados mais intensamente nos requisitos e fase de projeto.

(Características)

Configuração de processos

- Otimizar os processos.
- Evitar retrabalho.
- Aumentar a qualidade.

(Características)

Suporte a ferramentas

- Criar e manter os artefatos.
- Gerenciar configuração em cada iteração.

Conceitos

- . Trabalhadores (workers).
- . Artefatos.
- . Atividades.
- . Fluxos.
- . Plano de Iteração

Conceitos

Trabalhadores (workers).

A light blue callout bubble with a black outline, containing the text 'Quem?'. It is connected to the 'Trabalhadores (workers)' text by a thin black line.

Quem?

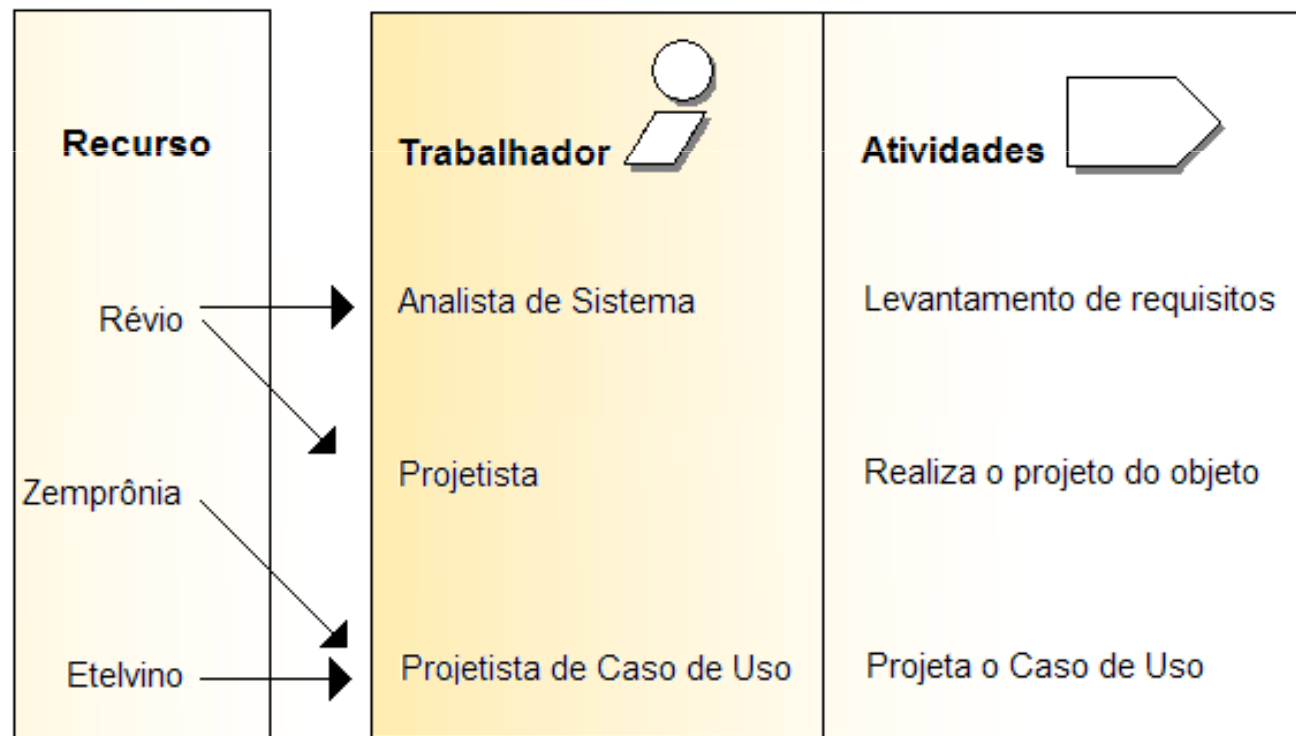
Um trabalhador define o comportamento e as responsabilidades de um indivíduo ou grupo de indivíduos que trabalham juntos com uma equipe. (KRUCTEN, 2003:30).

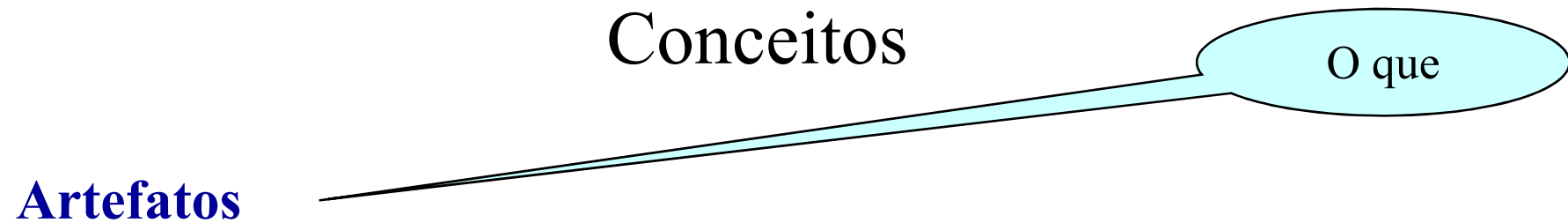
Esse comportamento é expresso por atividades que o trabalhador realiza, e cada trabalhador desempenha atividades específicas. O trabalhador representa o conceito central no processo RUP.

Renomeado para “papéis” no RUP 2001

Conceitos

Trabalhadores (workers).

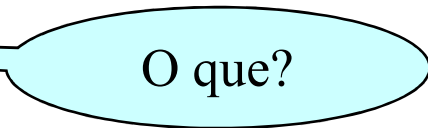




Um artefato é um pedaço de informação que é produzida, modificada ou usada por um processo. Os artefatos são produtos tangíveis do processo.(KRUCTEN, 2003:33).

Conceitos

Artefatos

A light blue oval callout bubble with a black outline, containing the text 'O que?'. A thin black line extends from the left side of the bubble towards the 'Artefatos' header.

O que?

Um artefato é um pedaço de informação que é produzida, modificada ou usada por um processo. Os artefatos são produtos tangíveis do processo.(KRUCTEN, 2003:33).

É uma peça física substituível de um sistema que contém informações físicas (“bits”). Em um sistema encontramos diversos tipos diferentes de artefatos de implantação com arquivos de código-fonte, executáveis e escripts. (BOOCH et al, 2006:21).

Os artefatos representam o produto das atividades de entrada e saída do RUP.

Conceitos

Exemplos de Artefatos



Conceitos

Atividades

Como?

Todo trabalhador no processo RUP possui atividades que são desempenhadas por estes. As atividades possuem propósitos claros que representam criações e atualizações de artefatos. As atividades normalmente são específicas de cada trabalhador.

É uma unidade do trabalho que um indivíduo naquele papel pode ser pedido a desempenhar e isso produz um resultado significativo no contexto do projeto. (KRUCTEN, 2003:32).

Conceitos

Etapas da atividade

- . Reflexão.
- . Desempenho.
- . Revisão.

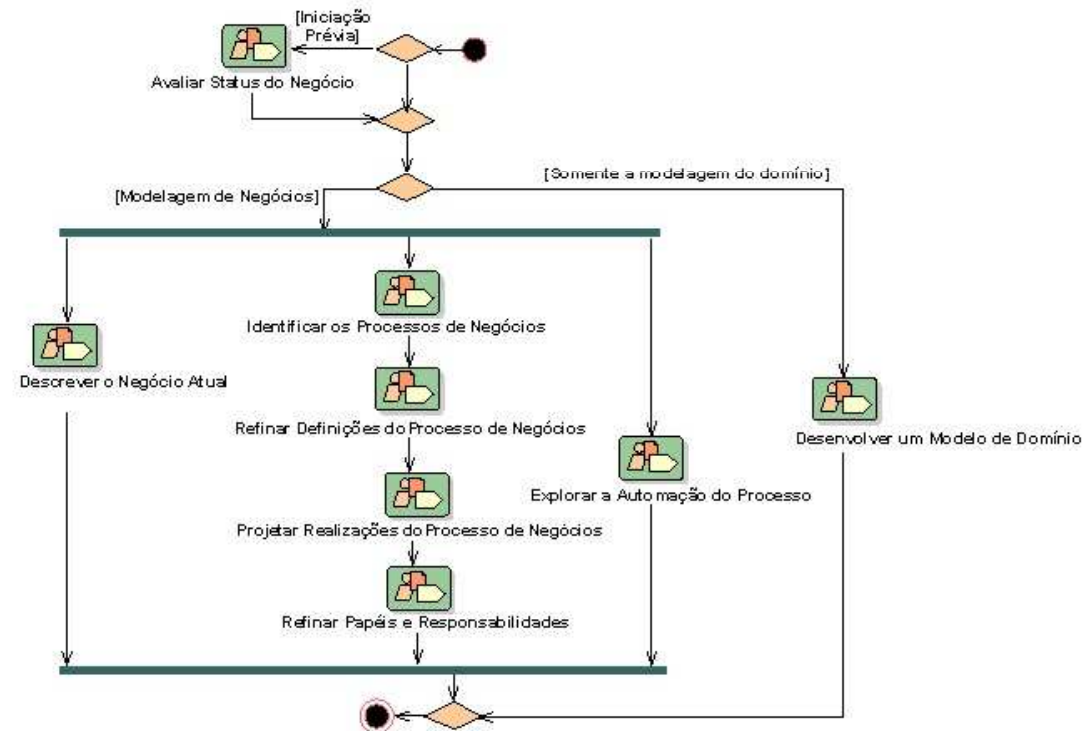
Conceitos

Fluxo de trabalho

Os fluxos de trabalho são representações dos caminhos percorridos das atividades para cada disciplina do RUP dentro do ciclo de vida do desenvolvimento. O fluxo de trabalho é representado por um diagrama de Atividades.

Conceitos – Fluxo de Trabalho

Fluxo de Trabalho da Modelagem de Negócios

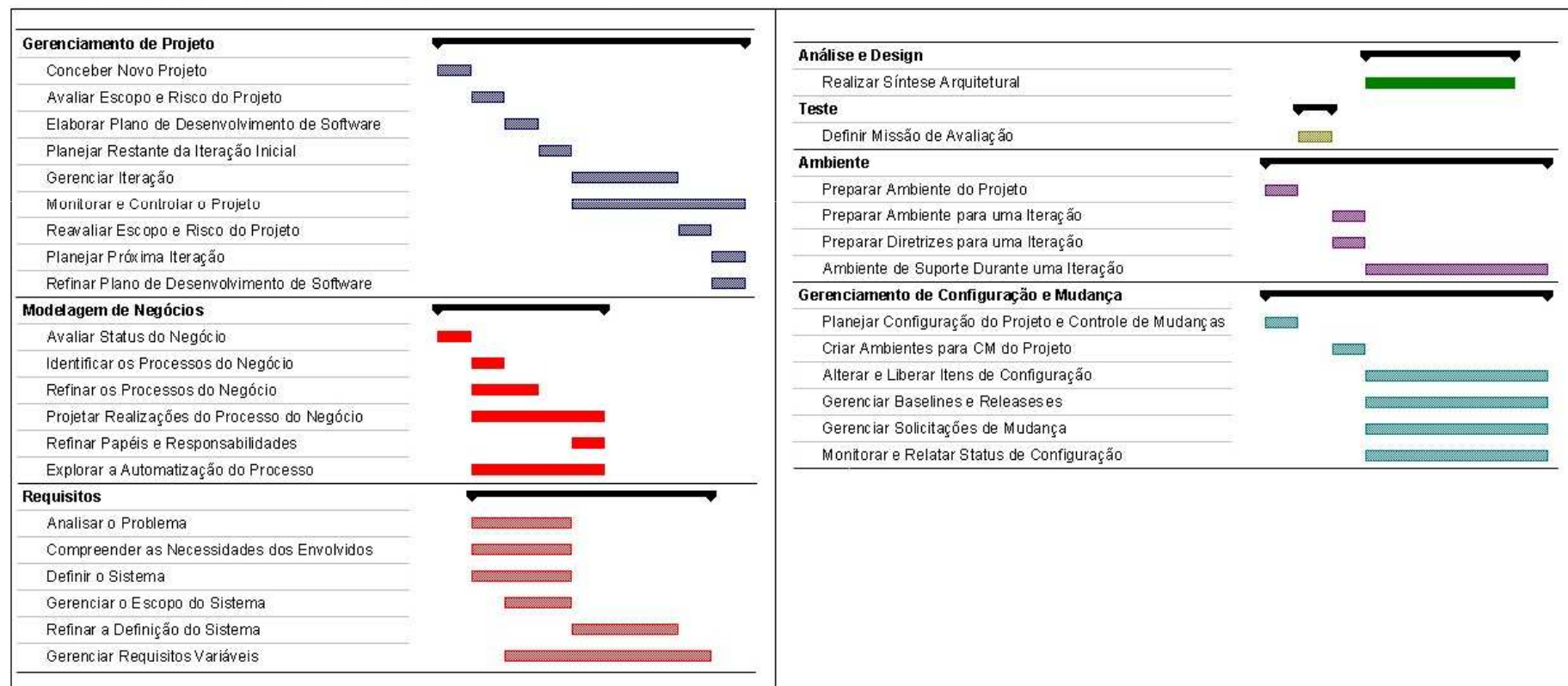


Conceitos – Plano de Iteração

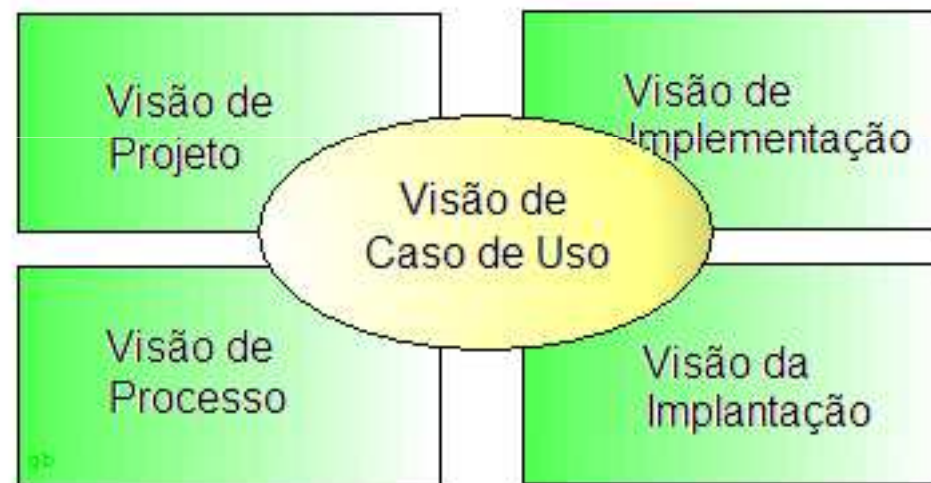
Mostra as iterações em cada uma das fases e as respectivas disciplinas envolvidas. Os planos de iteração são desenvolvidos a partir dos fluxos de trabalho de cada disciplina. O principal objetivo de um Plano de iteração é indicar as dependências e mostrar onde ocorrem fluxos de trabalho em paralelo. Não é a intenção do Plano de Iteração sugerir a aplicação de um nível de esforço uniforme no decorrer dos fluxos de trabalho.

Conceitos – Plano de Iteração

Plano de Iteração (Exemplo para a fase de Iniciação)



Arquitetura - O modelo de visão 4 + 1 da arquitetura



Fonte:
Desenvolvendo Aplicações com UML - Ana Cristina Melo - BrasPort - 1a. Edição

Visão de Caso de Uso

Segundo BOOCH (2006) é a visão de da arquitetura de um sistema, abrangendo os casos de uso que descrevem o comportamento do sistema, conforme é visto pelos seus usuários finais, analistas e pessoal de teste.
(BOOCH et al, 2006:462).

Visão de Implantação

Segundo BOOCH (2006) é a visão de arquitetura de um sistema, abrangendo os nós que formam a topologia de hardware, em que o sistema é executado; a visão de implantação abrange a distribuição, entrega e instalação das partes que constituem o sistema físico.
(BOOCH et al, 2006:462).

Visão de Implementação

Segundo BOOCH (2006) é a visão da arquitetura do sistema abrangendo os artefatos utilizados para montar e liberar o sistema físico, inclui o gerenciamento de configuração das versões do sistema, composta por artefatos de alguma forma independentes que podem ser reunidos de vários modos para produzir um sistema em execução .(BOOCH et al, 2006:462).

Visão de Processo

Possui enfoque no desempenho e na escalabilidade.

Elementos: classes ativas.

Diagramas: classes, objetos, interação e gráfico de estados.

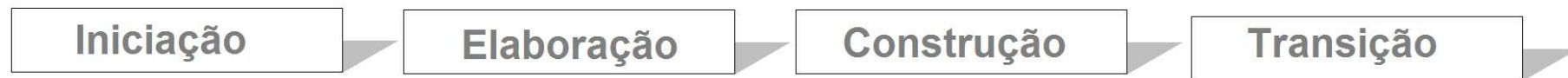
Visão de Projeto

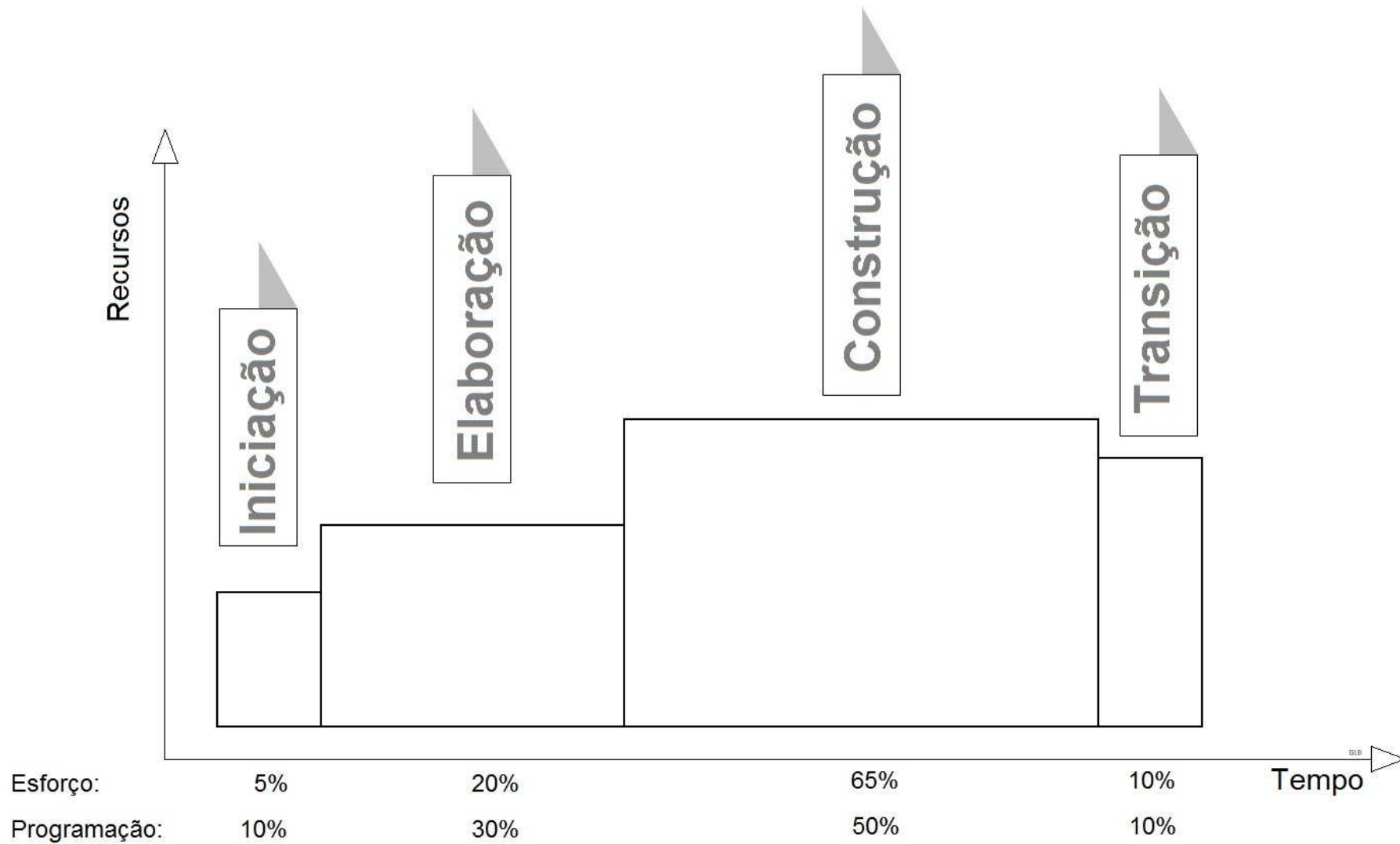
Segundo BOOCH (2006) é a visão da arquitetura de um sistema, abrangendo as classes, interfaces e colaborações que formam o vocabulário do problema e a sua solução; a visão de projeto abrange os requisitos funcionais do sistema. (BOOCH et al, 2006:462).

Fases

Uma fase é basicamente um intervalo de tempo entre dois marcos principais. São usadas para auxiliar no gerenciamento do ciclo de vida do processo RUP. Cada fase é concluída por um marco principal que é verificado no final, a partir da atividade Revisar Marcos do Ciclo de Vida, determinando assim se os objetivos da fase foram alcançados. Caso a avaliação seja satisfatória o próximo passo (próxima fase) será executado.

Fases

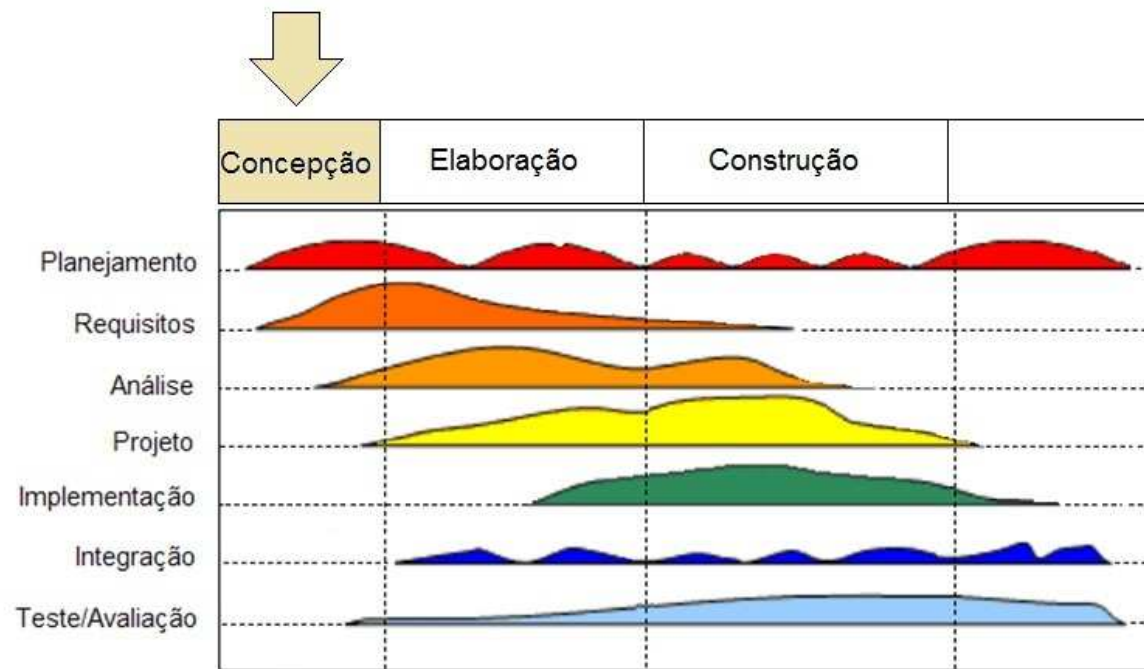




Fases do RUP

Detalhamento

Fases do RUP



GLE

Fases

Fase de Concepção – Atividades/Metas

- Idéia inicial do projeto.
- Estudos de viabilidade.
- Estimativa de riscos.
- Estimativa do custo global.
- Determina a continuidade do projeto.

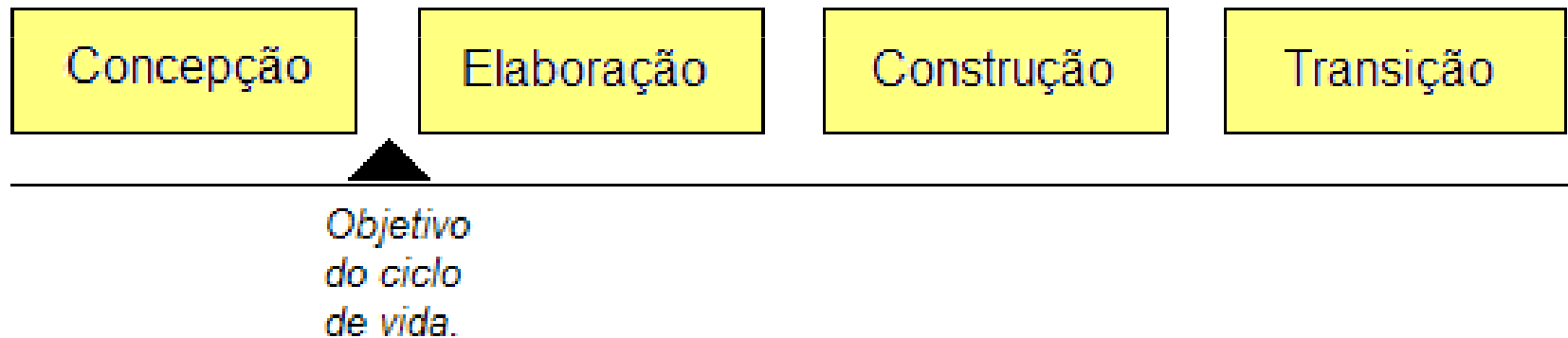
Fases

Fase de Concepção – Resultados

- Documento de visão.
- Lista de todos os casos de uso e atores.
- Modelo de domínio com previsões financeiras e riscos.

Fases

Fase de Concepção - Marco

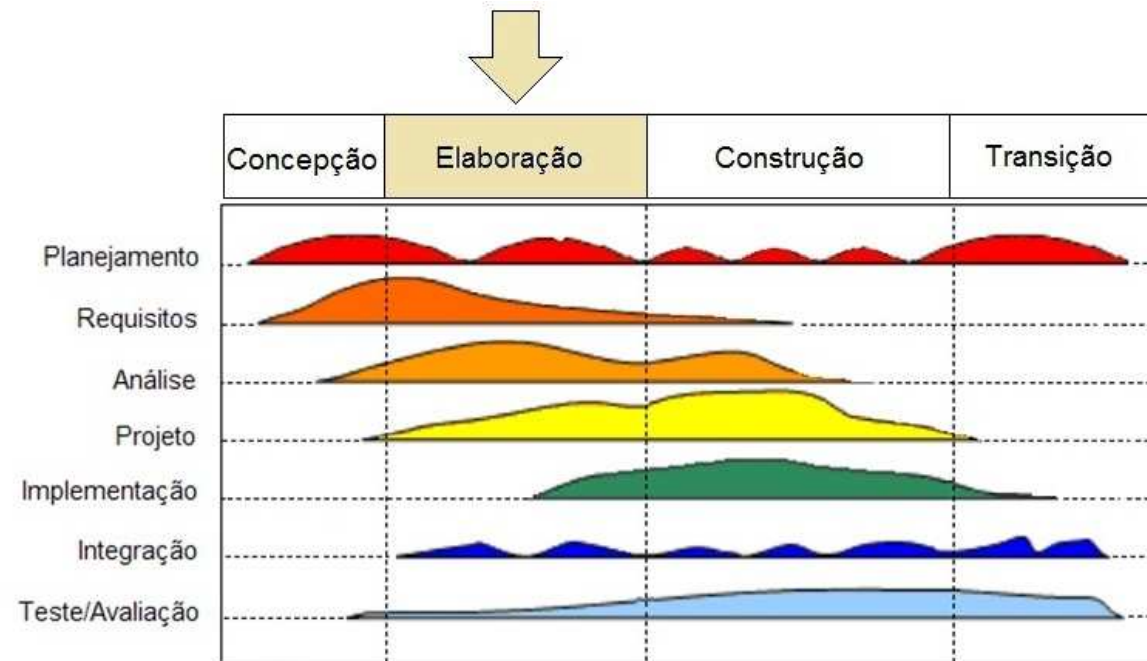


Fases

Fase de Concepção – Principais Trabalhadores e Artefatos

Fase	Trabalhador	Artefato
INICIAÇÃO	Analista de sistemas	Visão, glossário, modelos de casos de uso, guia de modelagem de casos de uso.
	Analista do processo de negócios	Modelo de Objetos de negócios.
	Gerente de projeto	Caso de Negócio, Lista de Riscos, Plano de Desenvolvimento de software, Plano de interação.
	Engenheiro de Processo	Caso de Desenvolvimento.
	Especialista em Ferramentas	Ferramentas que suportarão o esforço do Desenvolvimento de software.
	Gerente de configuração	Repositório de projeto

Fases do RUP



GLB

Fases

Fase de Elaboração – Atividades/Metas

- Determina o que e como será construído.
- Maior detalhamento (análise de riscos).
- Detalhamento dos diagramas de Caso de Uso e Classes.
- Para o caso de Workflow usar diagramas de Atividades.
- Definir um plano detalhado para fase de construção.

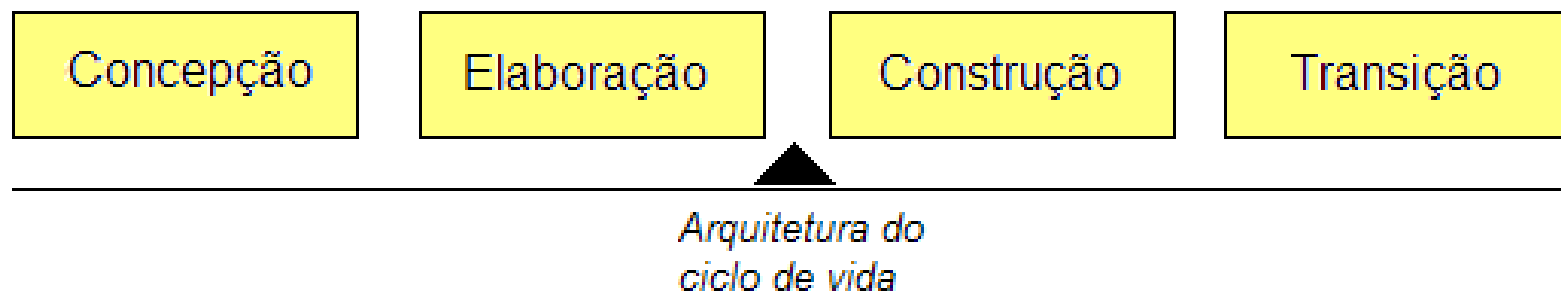
Fases

Fase de Elaboração – Resultados

- Modelos de caso de uso pelo menos 80% completo.
- Requisitos funcionais e não funcionais bem definidos.
- Descrição global da arquitetura do ciclo de vida.
- Um protótipo arquitetônico executável.

Fases

Fase de Elaboração – Marco

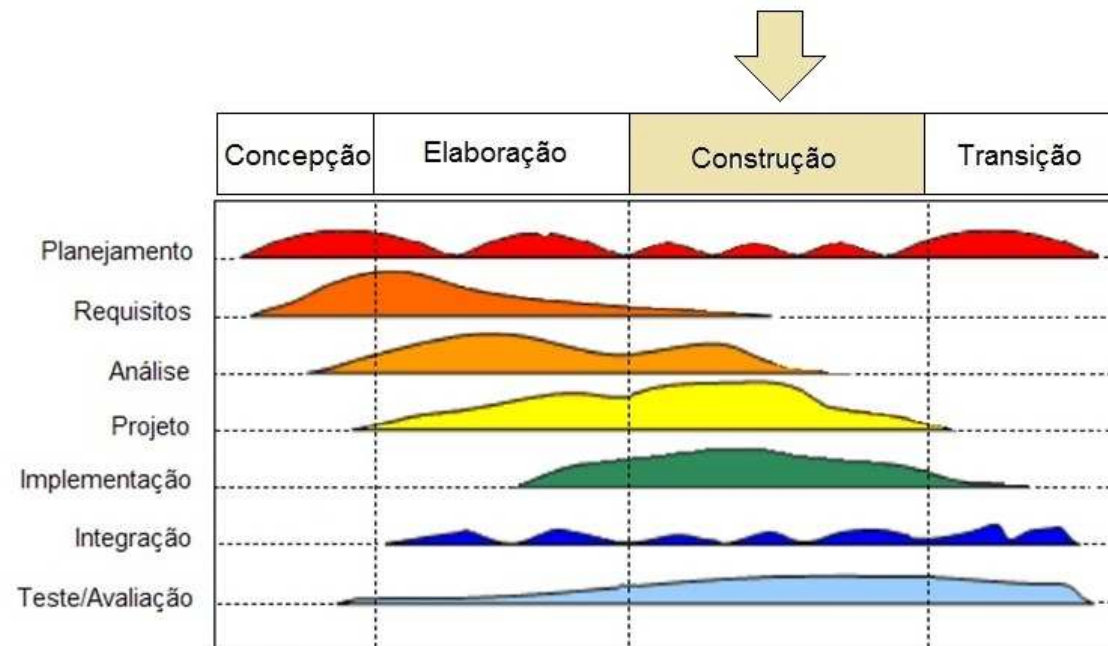


Fases

Fase de Elaboração – Principais Trabalhadores e Artefatos

Fase	Trabalhador	Artefato
ELABORAÇÃO	Arquiteto de Software	Documento de arquitetura de software, Modelo de Design , Guia de design, Modelo de Análise.
	Designer de Banco de dados	Modelo de dados.
	Designer de Testes	Conjunto de Testes.
	Engenheiro de Processo	Caso de Desenvolvimento.
	Especialista em Ferramentas	ferramentas que suportarão o esforço do Desenvolvimento de software.
	Desenvolvedor do curso	Materiais de Treinamento.

Fases do RUP



GLE

Fases

Fase de Construção – Atividades/Metas

- Fazer o planejamento prévio (usar o diagrama de Caso de Uso).
- Definir as prioridades (baseadas nos Casos de Uso).
- Desenvolvimento dos componentes e testes.
- Análise, projeto, codificação, testes e documentação.

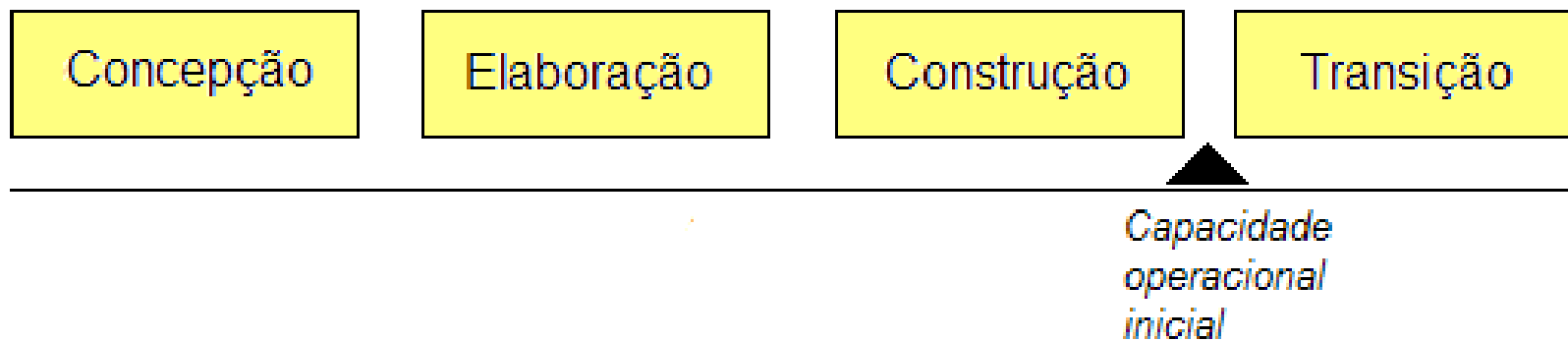
Fases

Fase de Construção – Resultados

- Produto acabado (com sua capacidade operacional inicial).
- Integração do produto com as plataformas adequadas.
- Manuais do usuário.

Fases

Fase de Construção – Marco

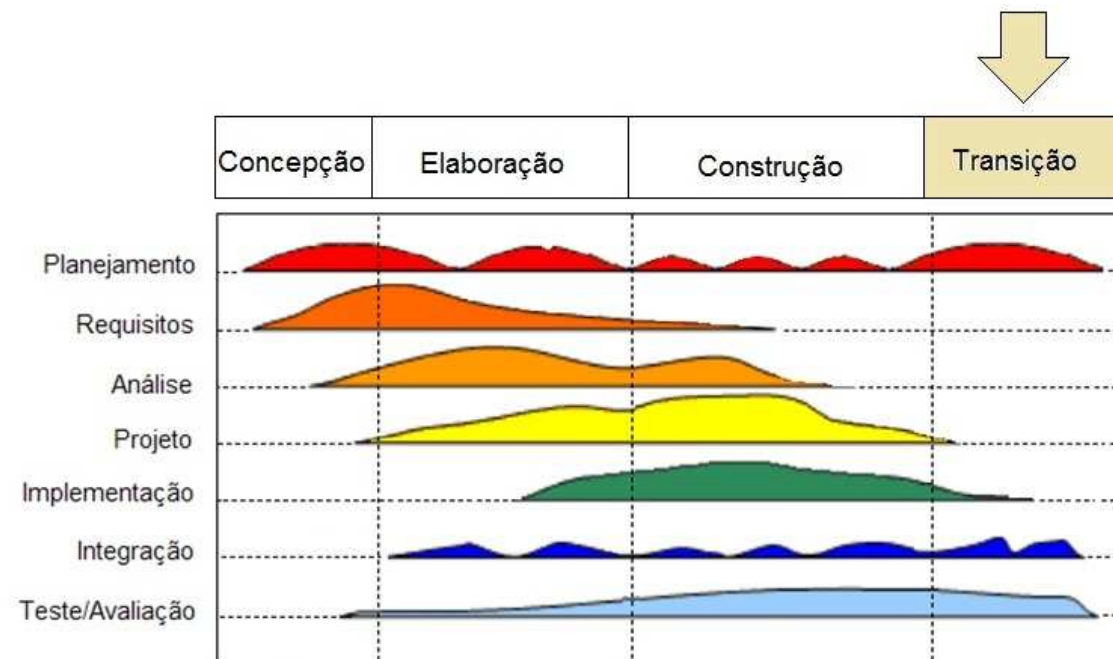


Fases

Fase de Construção – Principais Trabalhadores e Artefatos

Fase	Trabalhador	Artefato
CONSTRUÇÃO	Gerente de Implantação	Plano de Implantação
	Arquiteto de Software	Modelo de implementação, Modelo de Design.
	Designer de Testes	Conjunto de Testes.
	O gerente de projeto	Plano de interação.
	Engenheiro de Processo	Caso de Desenvolvimento.
	Designer de Banco de dados	Modelo de dados.
	Analista de sistemas	Modelos de casos de uso.

Fases do RUP



GLB

Fases

Fase de Transição – Atividades/Metas

- Otimização.
- Correção de erros finais.
- Treinamento de usuários.
- Aceite oficial do usuário (o mais rápido possível).
- Operação paralela com o sistema legado.
- Passagem da versão beta para versão de entrega.

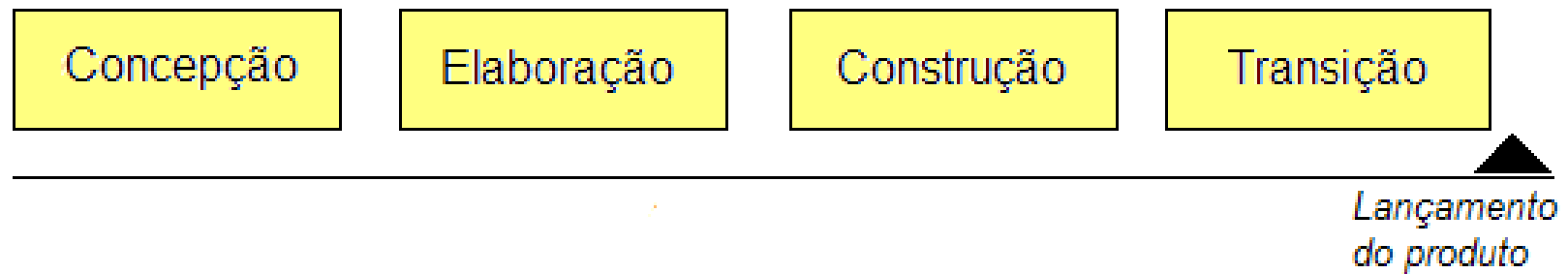
Fases

Fase de Transição – Resultados

- Início da fase de concepção do próximo ciclo.
- Lançamento do produto;

Fases

Fase de Transição – Marco



Fases

Fase de Transição – Principais Trabalhadores e Artefatos

Fase	Trabalhador	Artefato
TRANSIÇÃO	Gerente de implantação	Produto, Notas de Release.
	Implementador	Artefatos de instalação.
	Redator técnico	Material de suporte para usuário.
	Designer de Testes	Conjunto de Testes.
	Desenvolvedor do curso	Materiais de Treinamento.

Fluxos

É uma seqüência de atividades que produz um resultado de valor observável. (KRUCTEN, 2003:36).

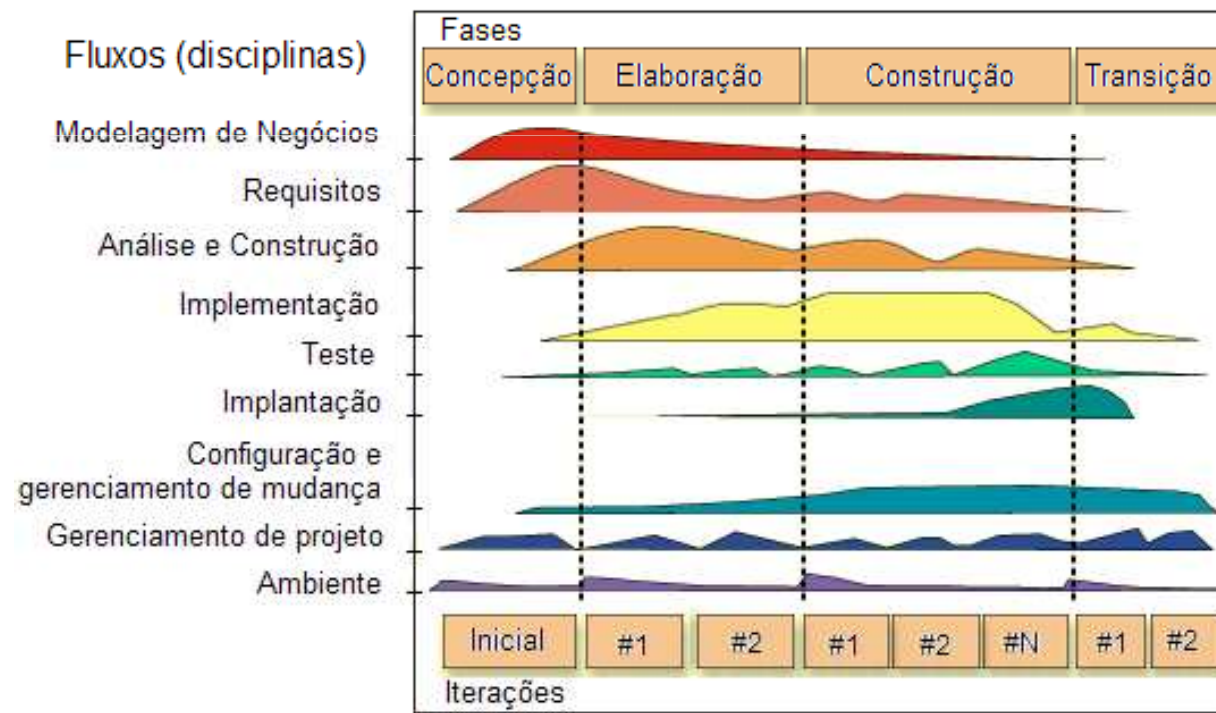
Fluxos

Fluxos centrados no processo

1. Modelagem de negócio.
2. Requisitos.
3. Análise e projeto.
4. Implementação.
5. Testes.
6. Distribuição e configuração.
7. Gerenciamento de mudança.
8. Gerenciamento de projeto.
9. Ambiente.

Fluxos

Fluxos centrados no processo

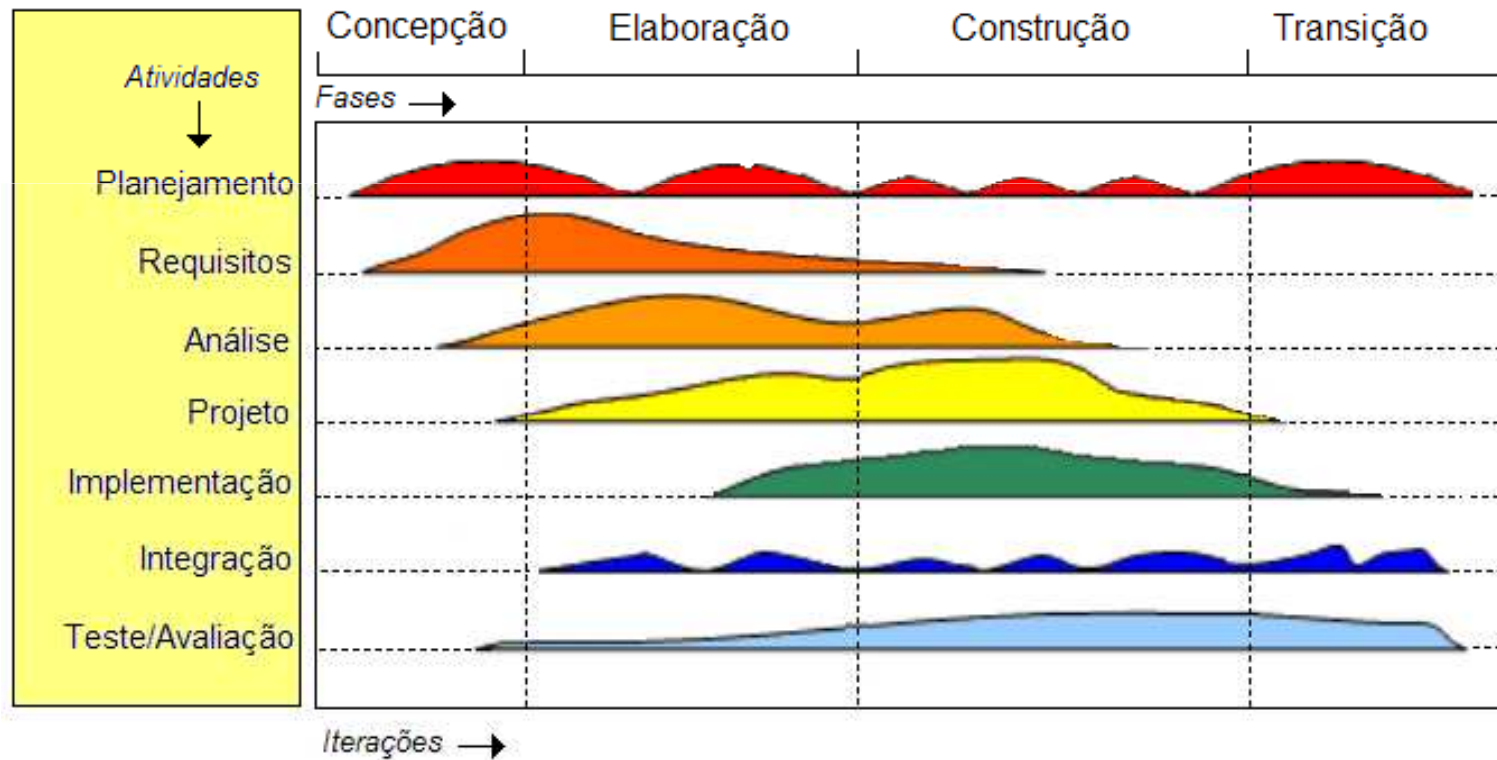


Fluxos

Atividades – Foco através da mudança no ciclo

1. Atividade de planejamento.
2. Atividade de requisitos.
3. Atividade de análise.
4. Atividade de projeto.
5. Atividade de Implementação.
6. Atividade de distribuição.
7. Atividade de Integração.
8. Atividade de Teste e avaliação.

Atividades – Foco através da mudança no ciclo



RUP (Rational Unified Process) - Casos de Uso

O Rational Unified Process é um processo de desenvolvimento dirigido a Caso de Uso.

RUP (Rational Unified Process) - Casos de Uso

Definição para caso de uso segundo o Rational Unified Process

Um caso de uso é uma sucessão de ações executadas por um sistema, que rende um resultado observável de valor a um ator em particular.
(RATIONAL apud KRUCHTEN, 2003: 81).

Definição para caso de uso

O diagrama de Caso de Uso procura, por meio de uma linguagem simples, possibilitar a compreensão do **comportamento externo** do sistema por qualquer pessoa, tentando apresentar o sistema por intermédio de uma **perspectiva do usuário**. [...] tem por objetivo apresentar uma **visão externa geral da funções e serviços** que o sistema deverá oferecer aos usuários, sem se preocupar como tais funções serão implementadas. [...] é de grande auxílio na etapa de **levantamento de requisitos**. (BEZERRA, 2007:48).

Identificação de Caso de Uso

- Quais são as necessidades e objetivos de cada ator?
- Que informações o sistema deve produzir?
- O sistema realiza alguma ação regular e periódica?
- Quais são os requisitos funcionais do sistema?

RUP (Rational Unified Process) - Casos de Uso

Segundo KRUCHTEN, (2003)

- São meios de expressar requisitos funcionais.
- São compreensíveis por uma ampla gama de usuários.
- Ajudam a sincronizar o conteúdo de vários modelos.
- Os cenários descrevem os casos de uso.
- **Conduzem várias atividades no RUP:**
 - . Criar e validar modelo.
 - . Definir casos de teste.
 - . Planejar iterações.
 - . Distribuir o sistema.

(KRUCHTEN, 2003: 91).

Referências Principais

FILHO, Wilson de Pádua. **Engenharia de Software – Fundamentos, Métodos e Padrões**. LTC, 2001.

KRUCHTEN, Philippe. **Introdução ao RUP - Rational Unified Process**. Rio de Janeiro: Ciência Moderna, 2003.

PRESSMAN, Roger S. **Engenharia de Software**. 6rd ed. McGraw-Hill, 2002.

SOMMERVILLE, Ian. **Engenharia de Software**. 8ª edição, São Paulo : Pearson Addison-Wesley, 2007.

Referências Relevantes

BEZERRA, E. **Princípios de Análise e projeto de sistemas com UML**. Rio de Janeiro: Campus, 2002.

MELO, Ana Cristina. **Desenvolvendo aplicações com UML**. Rio de Janeiro: Brasport, 2002.

TOM, PENDER. **UML: a Bíblia**. Rio de Janeiro : Elsevier , 2004.

GANE, CHRIS. **Análise Estruturada de Sistemas**. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S/A., 1983.