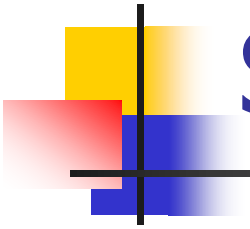
# Processo Unificado

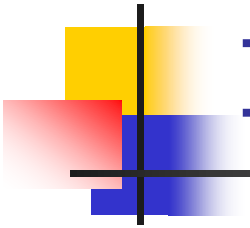
**Universidade Federal do Maranhão - UFMA**
**Departamento de Informática**
**Processo de Desenvolvimento de Software**
Profª.MSc Simara Rocha
simararocha@gmail.com/simara@deinf.ufma.br

Referências: Medeiros, E. Desenvolvendo Software com UML 2.0: Definitivo, Makron Books, 2006.
Pressman, R. S. Engenharia de Software, McGraw-Hill, 6ª. Edição, 2006
Sommerville, I. Engenharia de Software, 8ª edição, 2007.



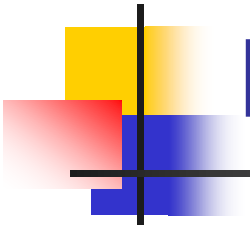
# Sumário

- Histórico
- Características
- Ciclo de Vida
- Fluxos de Trabalho
- Os artefatos do PU
- RUP



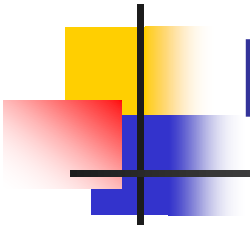
# Introdução

- O processo unificado encaixa-se na definição de processo: Um conjunto de atividades executadas para transformar um conjunto de requisitos do cliente em um sistema de software.
- O PU também é uma estrutura genérica de processo que pode ser customizado adicionando-se ou removendo-se atividades com base nas necessidades específicas e nos recursos disponíveis para o projeto.



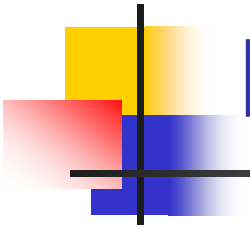
# Histórico

- O PU tem suas raízes no trabalho feito por Ivar Jacobson na decada de 60;
- Em 87 Jacobson deixou a empresa Ericsson, em que trabalhava, iniciando uma companhia chamada Objectory AB;
- Desenvolveram o Objectory, que era semelhante em estrutura com(Processo e Protudo) ao que hoje é o RUP;
- Seu livro Object-Oriented Software Engineering foi um marco na comunidade OO;



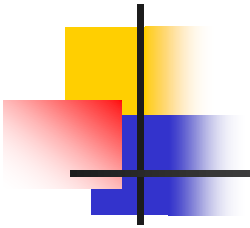
# Histórico

- Alguns anos apos a Rational comprou a Objectory AB;
- Em 94 foi construido o Processo Objectory da Rational (ROP) em paralelo com o Método Unificado, que depois foi chamado de UML;
- Em 98 a Rational mudou o nome do produto-processo para RUP.



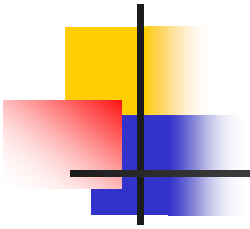
# Histórico

- Apesar do PU utilizar a UML para atividades de modelagem, eles são intrinsecamente separados
- A UML é independente de processo.
- Qualquer que seja o processo usado no desenvolvimento de um projeto, a UML poderá ser utilizada para registrar as decisões de análise e de projeto resultantes



# Características do PU

- É um framework genérico de um processo de desenvolvimento
- É baseado em componentes que realizam as interfaces
- Utiliza toda a definição da UML



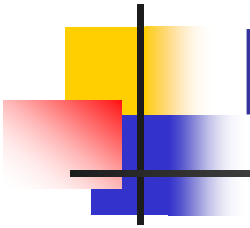
# Características do PU

- Alicerces:
  - Dirigido por Casos de Uso
  - Centrado na Arquitetura
  - Desenvolvimento iterativo e incremental
- Os 4 Ps:**pessoal, projeto, produto** e **processo.**

# P4 = Pessoa, Projeto, Produto, Processo

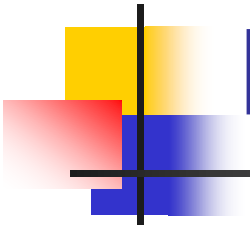
- **PESSOAS** financiam, escolhem, desenvolvem, gerenciam, testam, usam e são beneficiadas por produtos
- **PROJETOS** sofrem alterações. Determinam os tipos de *pessoas* que irão trabalhar no projeto e os *artefatos* que serão usados

# P4 = Pessoa, Projeto, Produto, Processo



- **PRODUTO** código fonte, código de máquina, subsistemas, classes, diagramas: interação, de estados e outros artefatos
  - **ARTEFATO** é qualquer tipo de informação criada por uma pessoa (diagramas UML, textos, modelos de interfaces)

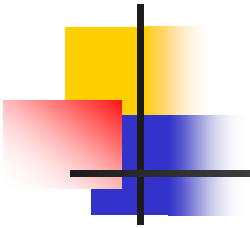
# P4 = Pessoa, Projeto, Produto, Processo



- **PROCESSO** define **quem (papel)** faz **o que (artefato), quando (disciplina)** e **como (atividades)**
- **PU** é um processo. Considera fatores *organizacionais*, do *domínio*, *ciclo de vida* e *técnicos*

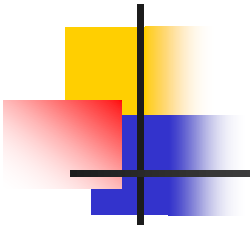
# Trabalhadores e Atividades

- Trabalhadores
  - Um **trabalhador é alguém que desempenha um papel e é responsável pela realização de atividades para produzir ou modificar um artefato.** Exemplos: analista de sistemas, programador, testador etc.
- Atividades:
  - tarefa que um trabalhador executa a fim de produzir ou modificar um artefato.



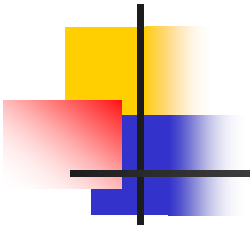
# Dirigido por Casos de Uso

- Um caso de uso é uma seqüência de ações, executadas por um ou mais atores, que produz um ou mais resultados de valor para um ou mais atores.
- Um ator é uma pessoa ou outro sistema.
- O PU é dirigido por casos de uso, pois utiliza-os para dirigir todo o trabalho de desenvolvimento, desde a captação inicial e negociação dos requisitos até a aceitação do código (testes)



# Dirigido por Casos de Uso

- Os casos de uso possibilitam que os requisitos funcionais possam ser capturados na perspectiva de cada um dos usuários do sistema, e com isso, definir o comportamento, as respostas esperadas e os casos de testes que devem validar a implementação do sistema.

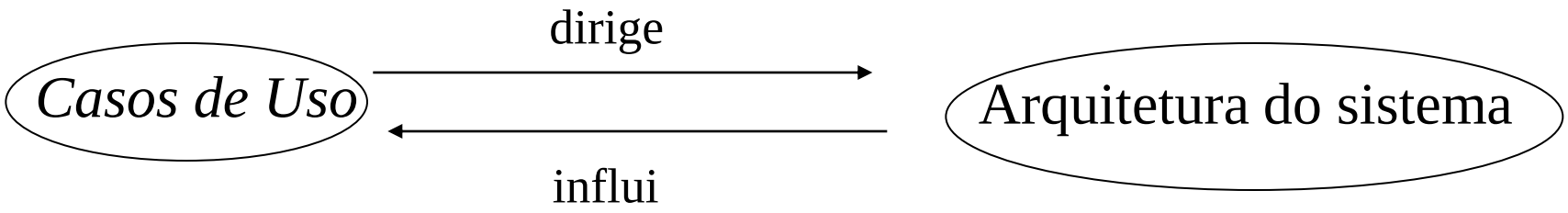


# Dirigido por Casos de Uso

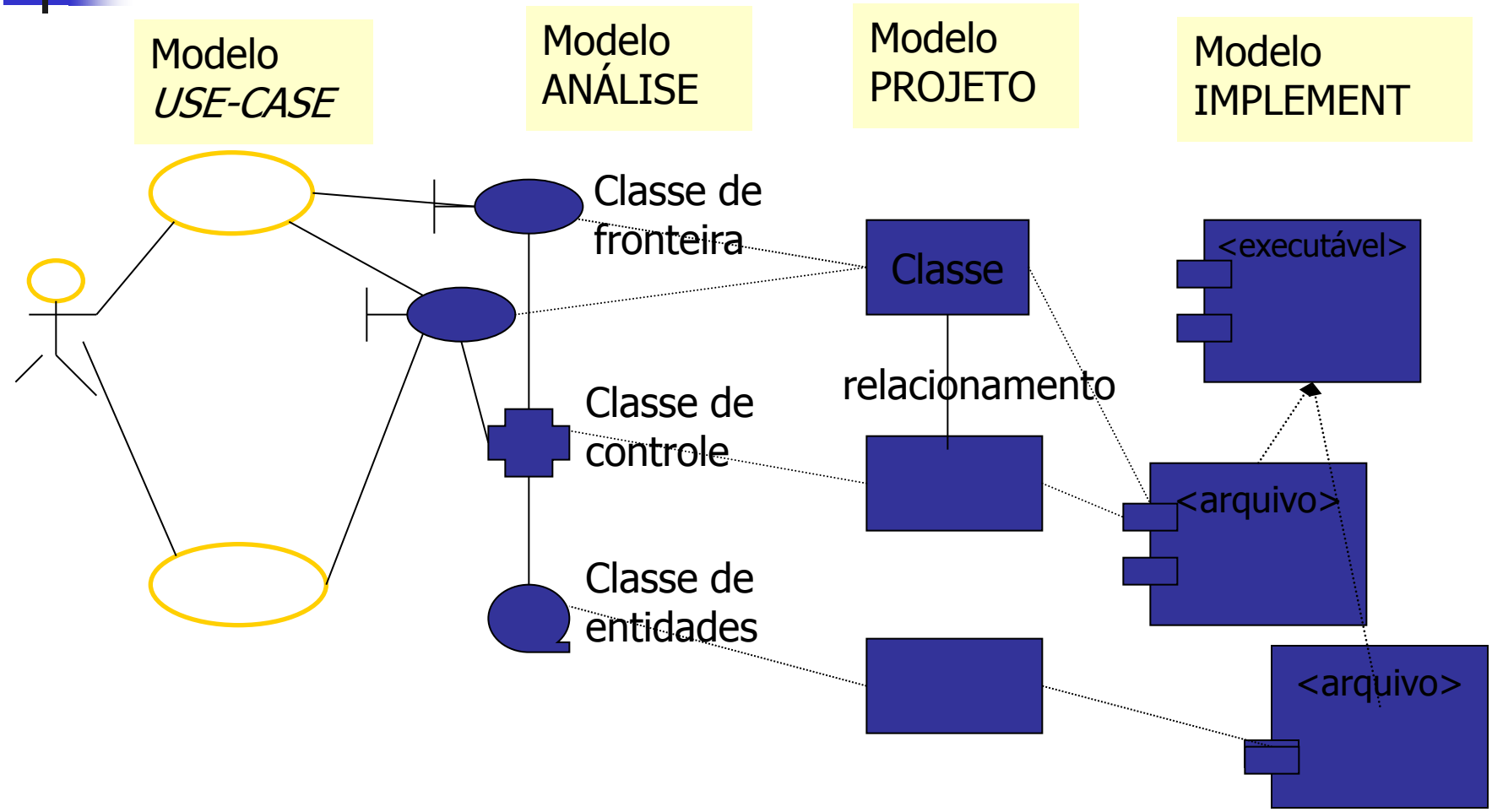
- Os casos de uso acompanham todo o processo de desenvolvimento: especificação de requisitos, análise, projeto, implementação e testes.
- Modelo casos de uso $= \Sigma$ casos de uso = funcionalidade do sistema

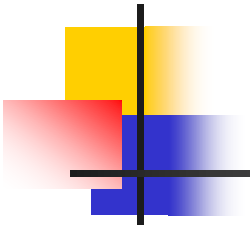
# Por que Casos de Uso?

- Os requisitos funcionais são registrados preferencialmente por meio deles
- Eles ajudam a planejar as iterações
- Eles podem conduzir o projeto
- O teste é baseado neles.



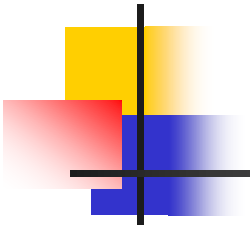
# Dirigido a Casos de Uso





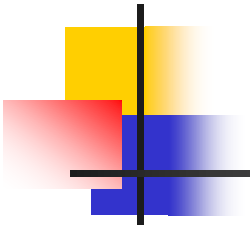
# Centrado na arquitetura

- Uma arquitetura é um "mapa" do sistema que define as diferentes partes do mesmo, seus relacionamentos, interações e mecanismos de interconexão, uma vez que englobam tanto os aspectos estáticos quanto os dinâmicos de um sistema.



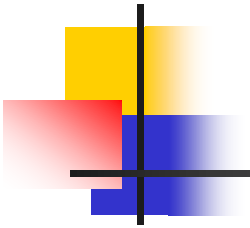
# Centrado na arquitetura

- A arquitetura também se refere a questões como desempenho, escalabilidade, reuso e restrições econômicas e tecnológicas.
- Por isso, é importante definir uma arquitetura cedo e refiná-la com o tempo.



# Centrado na arquitetura

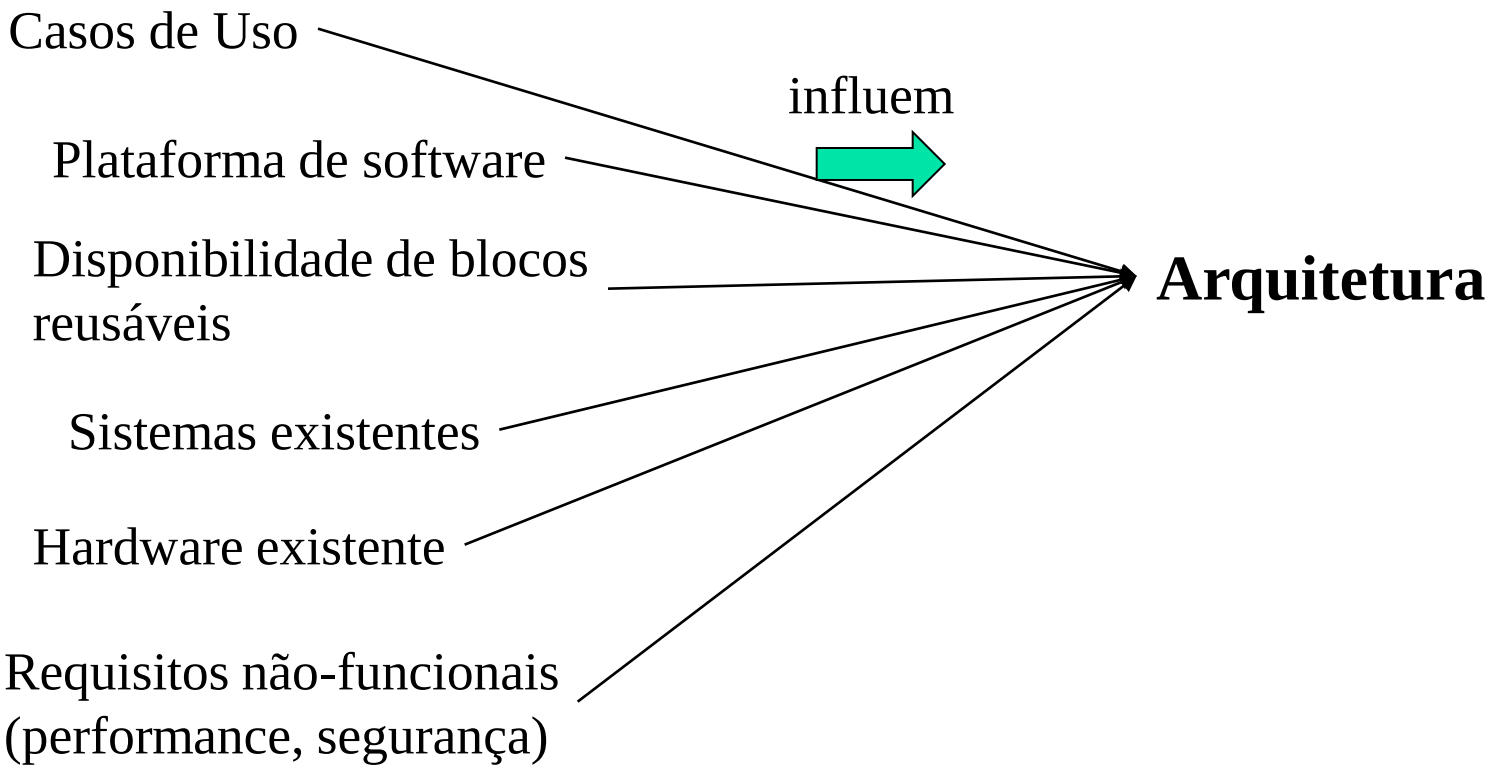
- O Processo Unificado é centrado na arquitetura e esta é descrita como diferentes visões do sistema, sendo influenciada pelos casos de uso, sistemas existentes, plataforma de desenvolvimento, framework, etc.



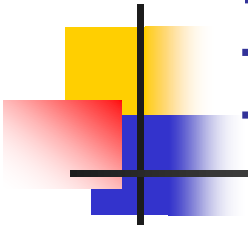
# Centrado na arquitetura

- A arquitetura é importante porque:
  - Ajuda a entender a visão global
  - Ajuda a organizar o esforço de desenvolvimento
  - Facilita as possibilidades de reuso
  - Facilita a evolução do sistema
  - Tem base nos casos de uso especificados.

# Centrado na arquitetura

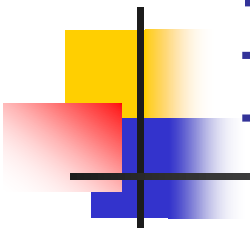


# Desenvolvimento Iterativo e Incremental



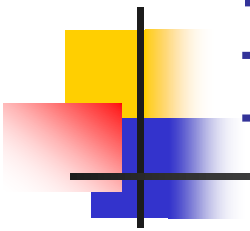
- O PU é basicamente um processo de desenvolvimento iterativo e incremental, no qual o software não é implementado a partir de um instante no fim do projeto, mas ao contrário, o ciclo de vida no PU é desenvolvido e implementado em iterações.
- Uma iteração é um miniprojeto que resulta em uma versão do sistema liberada interna ou extermamente.

# Desenvolvimento Iterativo e Incremental



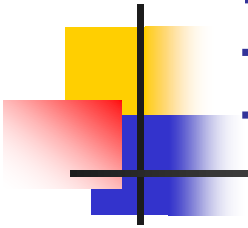
- Essa versão oferece uma melhora incremental sobre a iteração anterior.
- Os desenvolvedores terão que identificar e ordenar logicamente as iterações necessárias para atingir os objetivos do projeto.

# Desenvolvimento Iterativo e Incremental

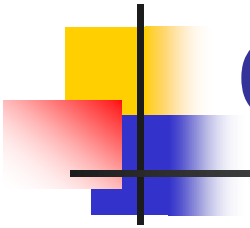


- Para cada iteração os desenvolvedores terão que identificar e especificar os casos de usos relevantes, desenvolver a atividade usando a arquitetura escolhida como guia, implementar o modelo de design em componentes e verificar se estes satisfazem os casos de uso.

# Desenvolvimento Iterativo e Incremental

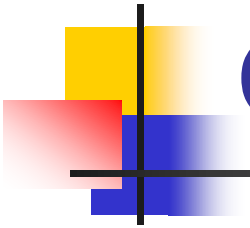


- Benefícios:
  - Identificação de riscos é adiantada
  - Preparação do sistema para requisitos que mudam
  - Integração contínua (facilita testes) e aprendizado facilitado
- Resultado de uma iteração: incremento.



# Ciclo de Vida

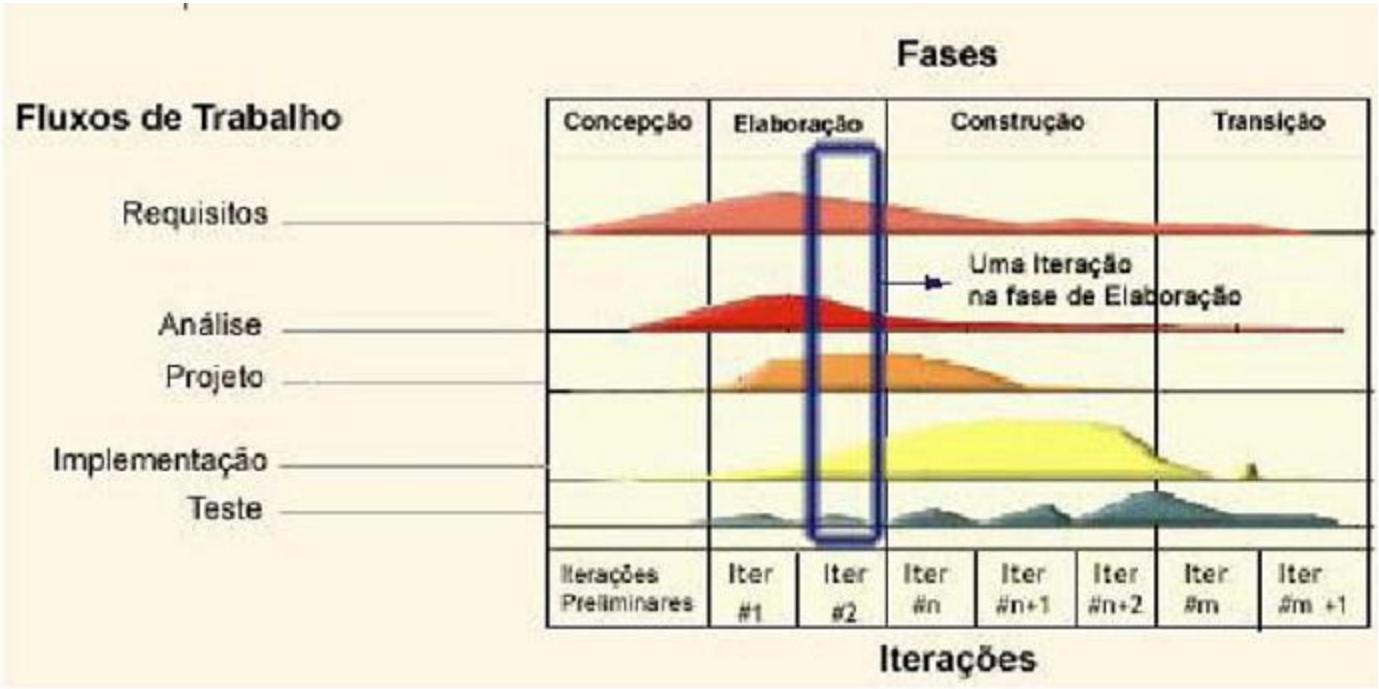
- O Processo Unificado consiste da repetição de uma série de ciclos durante a vida de um sistema.
- Cada ciclo é formado por quatro fases:
  - Concepção
  - Elaboração
  - Construção
  - Transição

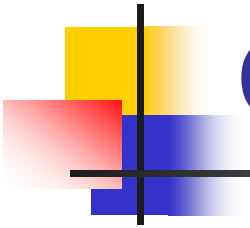


# Ciclo de Vida

- Cada fase, por sua vez, é subdivida em iterações que passam por todos os cincos fluxos do trabalho do processo:
  - requisitos, análise, projeto, implementação e teste.

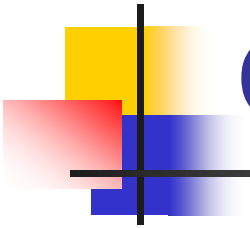
# Ciclo de Vida





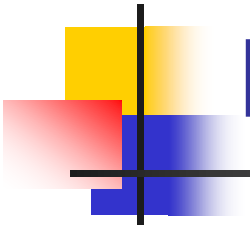
# Concepção

- O objetivo é estabelecer a viabilidade do sistema proposto.
- Nessa fase se faz:
  - Definição do escopo do sistema.
  - Estimativas de custos e cronograma
  - Identificação dos potenciais riscos que devem ser gerenciados ao longo do projeto
  - Esboço da arquitetura do sistema, que servirá como alicerce para a sua construção.



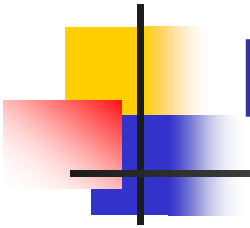
# Concepção

- Cada iteração está voltada para a produção de casos de uso de negócio e da arquitetura preliminar



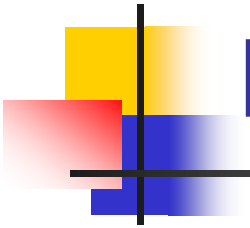
# Elaboração

- O objetivo é capturar o que pode ser feito no sistema com as restrições financeiras e outras questões levantadas...



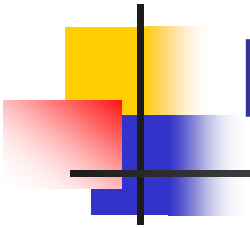
# Elaboração

- Nessa fase se faz:
  - Captura dos requisitos funcionais da aplicação (que não foram capturados na fase de concepção).
  - Expansão do esboço da arquitetura para uma arquitetura base para o desenvolvimento do sistema.
  - Abordagem dos riscos significativos.
  - Elaboração de um plano com questões econômicas indicando como o projeto vai evoluir na fase de construção.



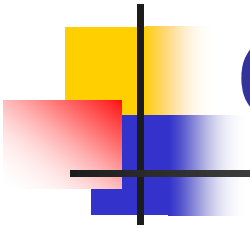
# Elaboração

- Nesta fase, as iterações estão voltadas para a produção da arquitetura básica
- Vários dos casos de uso são especificados com detalhes;
  - a arquitetura do sistema é projetada



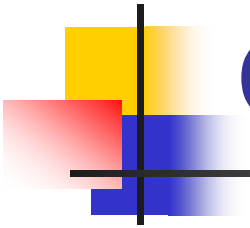
# Elaboração

- A arquitetura é expressa em visões dos modelos do sistema:
  - modelo de casos de uso
  - modelo de análise
  - modelo de projeto
  - modelo de implementação
  - modelo de distribuição
- O resultado da fase é o conjunto dos modelos acrescidos de elementos de implementação



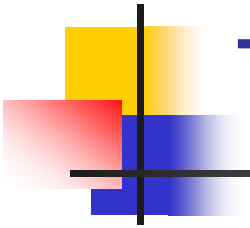
# Construção

- Nesta fase, as iterações estão voltadas para a produção de módulos executáveis, culminando com o desenvolvimento de um produto pronto para entrega à comunidade de usuários
- Durante esta fase o produto é construído
- O sistema torna-se um produto pronto para ser posto à disposição da comunidade de usuários



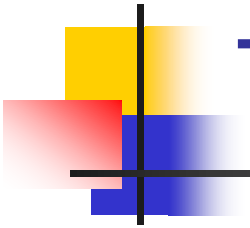
# Construção

- A arquitetura é estável, ainda que possa ser aperfeiçoada
- Ao final da fase, o sistema conterá todos os casos de uso que gerência e usuários concordaram em desenvolver para esta versão do produto
- Erros poderão ser encontrados e serão corrigidos



# Transição

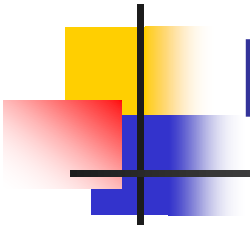
- Compreende o período em que o produto passa para beta-teste
  - a primeira versão é entregue aos usuários
- A fase envolve atividades de treinamento de usuários, assistência de uso do produto e correção de defeitos encontrados após a entrega



# Transição

- Um pequeno número de usuários experientes utiliza o produto e reporta os erros e inadequações encontradas
- Os desenvolvedores corrigem os erros e incorporam melhorias

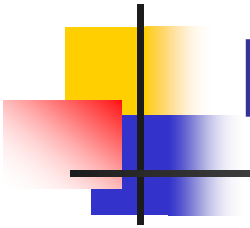
# Fluxos de Trabalho



- Avaliando-se as fases do PU, pode-se ter a impressão de que cada ciclo de iteração comporta-se como o modelo em cascata.
- Mas isso não é verdade
  - paralelamente às fases do PU, os fluxos de trabalho (requisitos, análise, projeto, implementação, testes), denominados disciplinas do PU, são realizados a qualquer momento durante o ciclo de desenvolvimento.

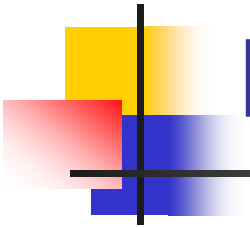
# Fluxos de Trabalho

- As disciplinas entrecortam todas as fases do PU, podendo ter maior ênfase durante certas fases e menor ênfase em outras, mas podendo ocorrer em qualquer uma delas.



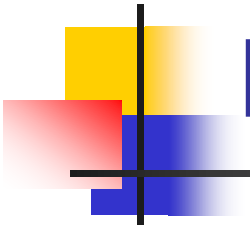
# Requisitos

- Durante o fluxo de trabalho, os requisitos do sistema são especificados através da identificação das necessidades de usuários e clientes
- Estes requisitos funcionais são expressos em casos de uso através do modelo de casos de uso
- O modelo de casos de uso é desenvolvido em vários incrementos, onde as iterações irão adicionar novos casos de uso e/ou adicionar detalhes as descrições dos casos de uso existentes.



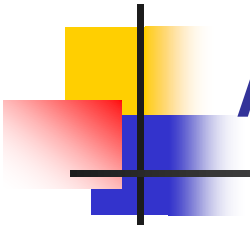
# Requisitos

- Durante a fase de concepção, os casos de uso mais importantes são identificados, delimitando o domínio do sistema.
- Durante a fase de elaboração, a maioria dos requisitos remanescentes é capturada, assim desenvolvedores poderão saber o quão deverão se empenhar para desenvolver o sistema



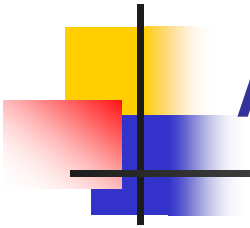
# Requisitos

- Ao final da fase de elaboração, devem ter sido capturados e descritos cerca de 80% dos requisitos do sistema
- Porém, apenas 5% a 10% destes requisitos são implementados nesta fase
- Os requisitos que sobram são capturados e implementados durante a fase de construção.
- Na fase de transição quase não há requisitos a serem capturados, a menos que ocorram mudanças nos mesmos.



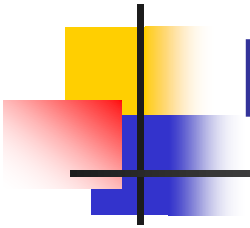
# Análise

- O produto do fluxo de análise é o modelo de análise.
- O modelo tem a função de refinar os requisitos especificados no fluxo anterior (fluxo de requisitos) através da construção de diagramas de classes conceituais permitindo, desta forma, argumentação a respeito do funcionamento interno do sistema.



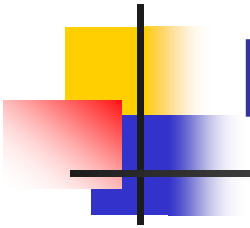
# Análise

- O modelo de análise fornece mais poder expressivo e formalismo como diagramas de interações e diagrama de gráficos de estados que representam a dinâmica do sistema.
- O fluxo de análise tem maior importância durante fase de elaboração
- Isso contribui para a definição de uma arquitetura estável e facilita o entendimento detalhado dos requisitos.



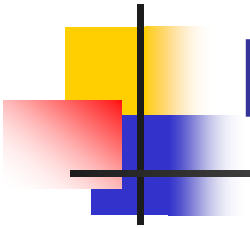
# Projeto

- No fluxo de projeto o sistema é moldado e sua forma é definida de maneira a suprir as necessidades especificadas pelos requisitos.
- Define um modelo de projeto que é construído com base no modelo de análise definido no fluxo anterior.
- O fluxo de projeto desenvolve o modelo de projeto que descreve o sistema em um nível físico.



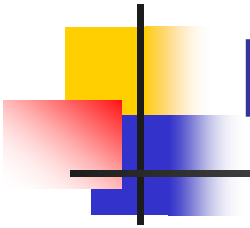
# Projeto

- A função principal deste fluxo é obter uma compreensão detalhada dos requisitos do sistema, levando em consideração fatores como linguagens de programação, SO, tecnologias de banco de dados, interface com o usuário, etc.
- O fluxo de projeto possui seu enfoque entre o fim da fase de elaboração e o início da fase de construção.
- Este fluxo serve de base para o fluxo de implementação



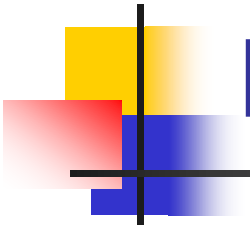
# Projeto

- Através do modelo de projeto, casos de uso são realizados por artefatos de projeto representados por ferramentas de modelagem também utilizadas no fluxo de análise como diagramas de classes, diagramas de interação e diagramas de estados, agora com o intuito de capturar os requisitos de implementação.



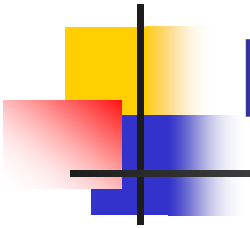
# Projeto

- Os mesmos diagramas citados são construídos em um nível mais físico que conceitual
- O fluxo de projeto também pode utilizar o diagrama de objetos



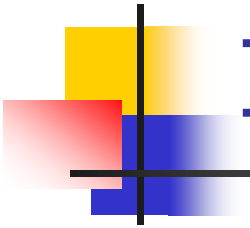
# Projeto

- O modelo de projeto descreve as realizações físicas de casos de uso considerando como requisitos, e outros detalhes relacionados ao ambiente de implementação, causam impacto ao sistema sob consideração.



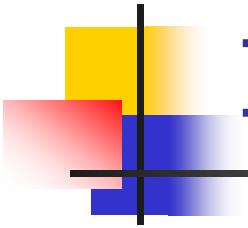
# Projeto

- O modelo de projeto funciona como uma abstração da implementação do sistema.
- Enquanto que o fluxo de análise se interessa por *o que* o sistema de fazer, o fluxo de projeto diz respeito a *como* os requisitos serão implementados



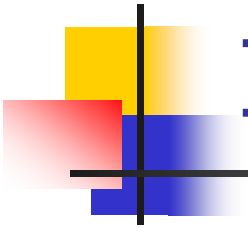
# Implementação

- O fluxo de implementação é baseado no modelo de projeto
- Implementa o sistema em termos de componentes (código fonte, executável, etc).
- A maior parte da arquitetura do sistema é definida durante o projeto.



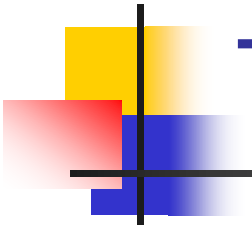
# Implementação

- O modelo de implementação se limita:
  1. Planejar as integrações do sistema em cada iteração. O resultado é um sistema que é implementado como uma sucessão de etapas pequenas e gerenciáveis
  2. Implementar os subsistemas encontrados durante o projeto
  3. Testar as implementações e integrá-los, compilando-as em um ou mais arquivos executáveis, antes de envia-los ao fluxo de teste.



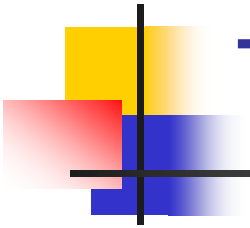
# Implementação

- Este fluxo tem maior importância na fase de construção e apesar de ter suas características próprias, a maior parte de suas atividades é realizada de forma quase mecânica, pelo fato das decisões mais difíceis terem sido tomadas durante o fluxo de projeto.
- O código gerado durantes a implementação, deve ser uma simples tradução das decisões de projeto em uma linguagem específica.



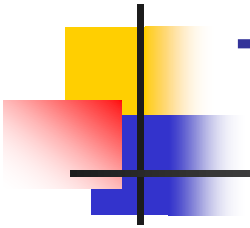
# Teste

- O fluxo de teste é desenvolvido com base no produto do fluxo de implementação.
- Os componentes executáveis são testados exaustivamente no fluxo de teste para então ser disponibilizados aos usuários finais.
- O principal propósito do fluxo de teste é realizar vários testes e sistematicamente analisar os resultados de cada teste.



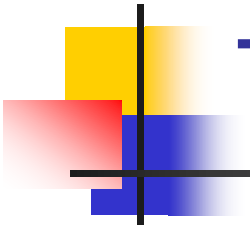
# Teste

- Componentes que possuírem defeitos retornarão a fluxos anteriores como os fluxos de projeto e implementação, onde os problemas encontrados poderão ser corrigidos.
- Um planejamento inicial de testes pode ser feito já na fase concepção.



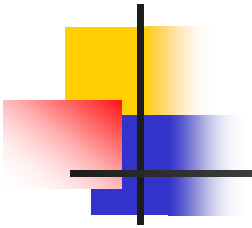
# Teste

- A maior parte dos testes são feitos durante a fase de elaboração, quando a arquitetura do sistema é definida, e durante a fase de construção quando o sistema é implementado.
- Na fase de transição, o fluxo de testes se limita ao conserto de defeitos encontrados durante a utilização inicial do sistema.



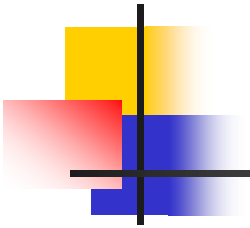
# Teste

- O produto do fluxo de teste é o modelo de teste
- O modelo de teste pode descrever:
  - como componentes executáveis, provenientes do fluxo de implementação, são testados.
  - como aspectos específicos do sistema testados, como por exemplo, se a interface do usuário é útil e consistente ou se o manual do usuário cumpre seu objetivo.



# Os Artefatos

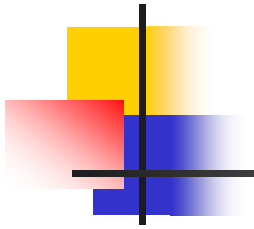
- Cada uma das disciplinas do PU pode gerar um ou mais artefatos, que devem ser controlados e administrados corretamente durante o desenvolvimento do sistema.
- Artefatos são quaisquer dos documentos produzidos durante o desenvolvimento, tais como modelos, diagramas, documentos de especificação de requisitos, código fonte ou executável, planos de teste, etc.



# Os Artefatos

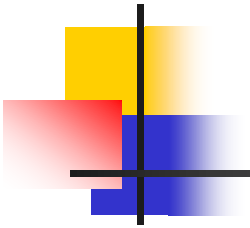
- Muitos dos artefatos são opcionais, produzidos de acordo com as necessidades específicas de cada projeto.
- Exemplos:
  - Diagrama de casos de uso
  - Diagrama de Classes
  - Diagrama de Sequência
  - Código fonte
  - Etc.

# RUP – Rational Unified Process

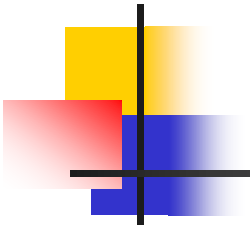


- É uma instância específica e detalhada do Processo Unificado.
- Processo proprietário de Engenharia de Software criado pela da RationalSoftware Corporation, adquirida pela IBM (atualmente IRUP)
- Fornece técnicas a serem seguidas pelos membros da equipe de desenvolvimento de software com o objetivo de aumentar a sua produtividade

# RUP – Rational Unified Process

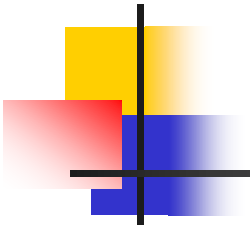


- É um processo considerado pesado e preferencialmente aplicável a grandes projetos.
- O fato de ser amplamente customizável torna possível que seja adaptado para projetos de qualquer escala
- O RUP é, por si só, um produto de software. É modular e automatizado, e toda a sua metodologia é apoiada por diversas ferramentas de desenvolvimento integradas e vendidas pela IBM através de seus "RationalSuites".



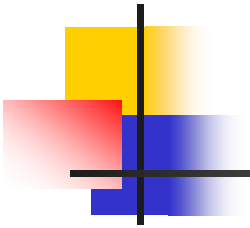
# RUP – Rational Unified Process

- Princípios e melhores práticas
  - Gestão e Controle de Mudanças do Software
  - Gerenciar requisitos
  - Usar arquiteturas baseada em componentes
  - Modelar o software visualmente (diagramas)
  - Verificar a qualidade do software.



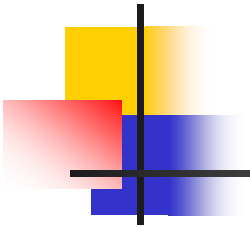
# RUP – Caraterísticas

- Semelhante ao PU:
  - Baseado em casos de uso
  - Orientado a arquitetura
  - Iterativo e incremental
  - Etc.



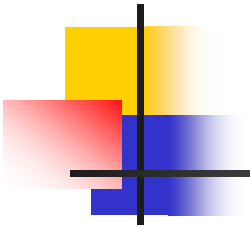
# RUP – Fases

- Semelhante ao PU:
  - Concepção
  - Elaboração
  - Construção
  - Transição



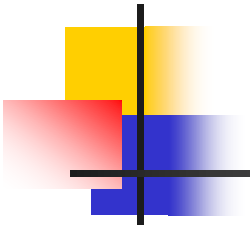
# RUP – Disciplinas

- Gerencia de Projeto
- Modelagem de Negócio
- Requisitos
- Análise e Projeto (união de A/P do PU)
- Teste Configuração e gerência de alterações
- Ambiente
- Instalação



# RUP – Artefatos

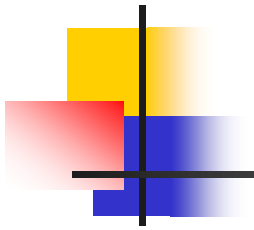
- Conjunto de Gerência
- Conjunto de Requisitos
- Conjunto de Projeto
- Conjunto de implementação
- Conjunto de Instalação



# RUP – Métodos Concorrentes

- Cleanroom (considerado pesado)
- E os MetodosÁgeis (leves) como a ***Programação Extrema*** (XP-ExtremeProgramming), e outros

# Semelhança entre RUP e XP



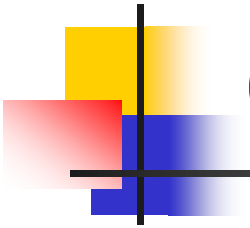
- Procuram facilitar a comunicação entre os vários interessados em um projeto
  - RUP -> UML
  - XP -> modelagem informal
- Integração contínua (escalas e ordens diferentes): “analisar um pouco, projetar um pouco, codificar um pouco, testar um pouco ...”
- Objetivam a simplicidade

# Diferença entre RUP e XP

- RUP: enfoque descendente de construção do software.
- XP: enfoque ascendente –“ descobrir o projeto em resposta à codificação e fatoração contínua”
- XP: foge da documentação formal, confia mais na documentação oral. O RUP não.
- XP: concebido para equipes pequenas e médias
- RUP: concebido para grandes projetos.

# Diferença entre RUP e XP

- RUP: enfoque descendente de construção do software.
- XP: enfoque ascendente
  - " descobrir o projeto em resposta à codificação e fatoração contínua"



# Conclusão

- O PU é um processo de desenvolvimento dirigido por casos de uso, centrado na arquitetura e iterativo e incremental
- É composto por 5 fluxos de trabalho
- Produz um conjunto de Artefatos
- O RUP é uma instância do PU voltado para projetos mais complexos.