Processo de Desenvolvimento de Software – RUP

Prof. Ingrid Nunes

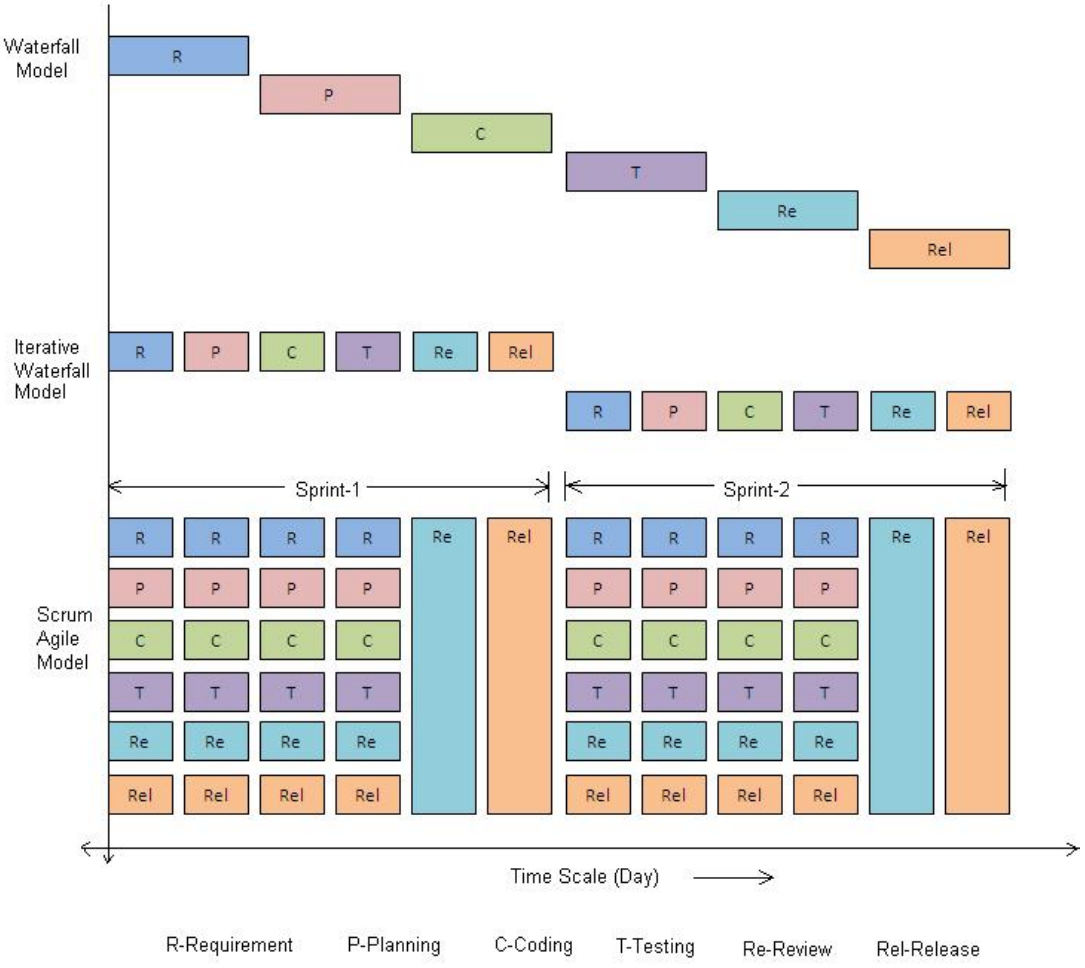
INF01127 - Engenharia de Software N

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

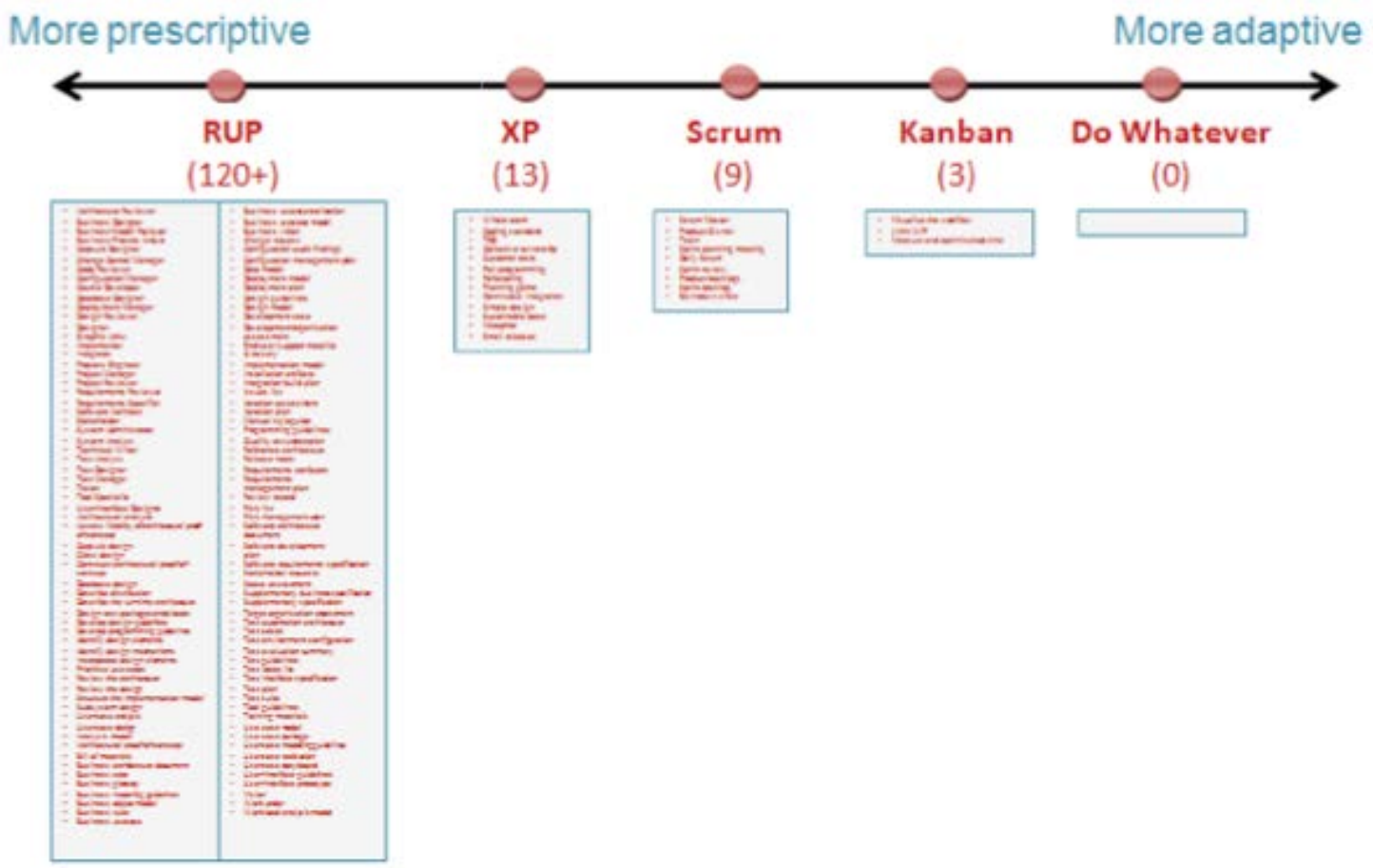
Prosoft

.inf
INSTITUTO DE INFORMÁTICA
UFRGS

Modelo Iterativo Incremental



Modelo Iterativo Incremental



Processo de Desenvolvimento de Software

RATIONAL UNIFIED PROCESS (RUP)

Rational Unified Process (RUP)

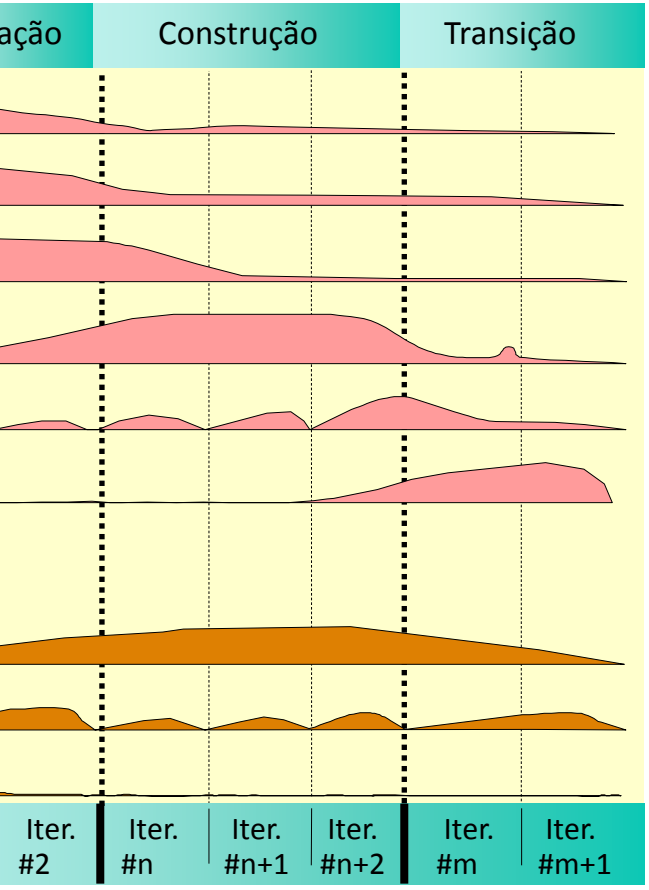
"Conjunto de atividades executadas para transformar requisitos do cliente em um sistema de software"
(Scott, K. 2003)

- Modelo híbrido de processo, i.e. traz elementos de todos os modelos genéricos
- Faz uso extensivo da UML e enfatiza o desenvolvimento em time
- Documentação na IBM
  - http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Perspectivas do RUP

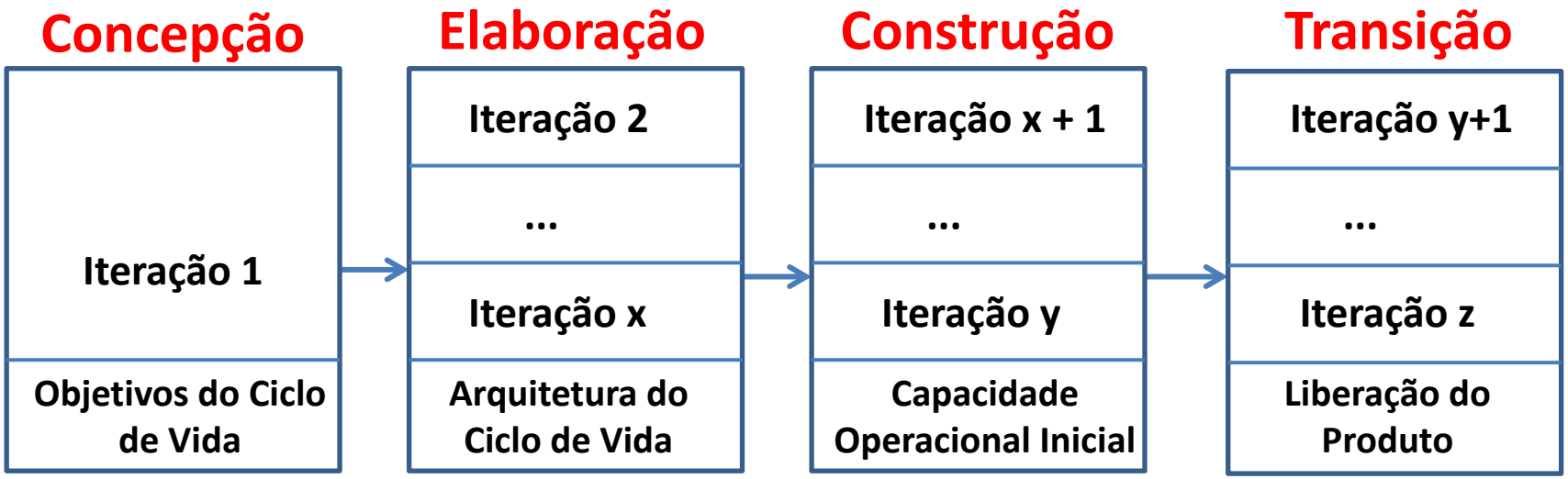
- **Dinâmica**
  - Mostra as **fases** do modelo ao longo do tempo
- **Estática**
  - Mostra as atividades realizadas no processo
    - **Workflows**
- **Prática**
  - Sugere as boas práticas a serem usadas ao longo do processo

Rational Unified Process (RUP)



Perspectiva: Dinâmica

- Ciclo de vida do software é dividido em ciclos
  - Cada ciclo tem 4 fases consecutivas



Importante: cada etapa tem um objetivo!

Dinâmica (1): Concepção

- Objetivos
  - Definir *business case* e escopo
    - Ex.: melhor performance aumenta a satisfação do cliente
  - Identificar casos de uso
    - Todas as entidades externas (pessoas e sistemas) e respectivas interações com o sistema
  - Avaliar a contribuição do sistema com o negócio
  - Importante
    - **Projeto pode ser cancelado nesta fase**

Dinâmica (1): Concepção

- Resultados
  - Documento com visão geral dos requisitos do projeto
  - Modelo de caso de uso inicial
    - 20-30% concluído
  - Glossário inicial do projeto
  - Modelo de negócio inicial, incluindo o contexto do negócio, critérios de sucesso, etc.
  - Plano do projeto com fases e iterações
  - Modelo de negócios
  - Um ou mais protótipos

Dinâmica (2): Elaboração

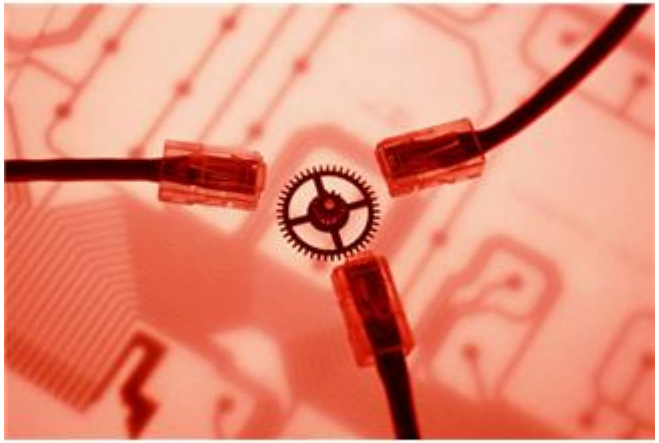
- Objetivos
  - Entender o domínio do problema
  - Criar um framework de arquitetura do sistema
  - Desenvolver o plano de projeto
    - Documenta a evolução do projeto à medida que ele vai sendo executado e novas informações vão sendo disponibilizadas
  - Identificar os principais riscos do projeto

Dinâmica (2): Elaboração

- Resultado
  - Modelo de caso de uso
    - Pelo menos 80% concluído
  - Descrição **dos requisitos não-funcionais**
  - Descrição da arquitetura de software
  - Protótipo executável da arquitetura de software
  - Lista de riscos revisada
  - Manual do usuário preliminar

Dinâmica (3): Construção

- Objetivos
  - Projeto, programação e teste do sistema
    - Partes do sistema são desenvolvidas e integradas



Dinâmica (3): Construção

- Resultados
  - Produto de software integrado na plataforma adequada
  - Manuais de usuário
  - Descrição da *release* corrente

Dinâmica (4): Transição

- Objetivos
  - **Transferência do sistema** da comunidade de desenvolvimento para a comunidade dos usuários
  - Entrada do sistema em funcionamento
- Resultados
  - Versão beta
  - Execução paralela com um sistema legado que esteja sendo substituído
  - Treinamento de usuários e manutenção

Perspectiva Estática

- Enfoca atividades que ocorrem durante o processo de desenvolvimento
  - Workflows
- Descrição dos workflows orientada em termos de modelos UML

Perspectiva Estática

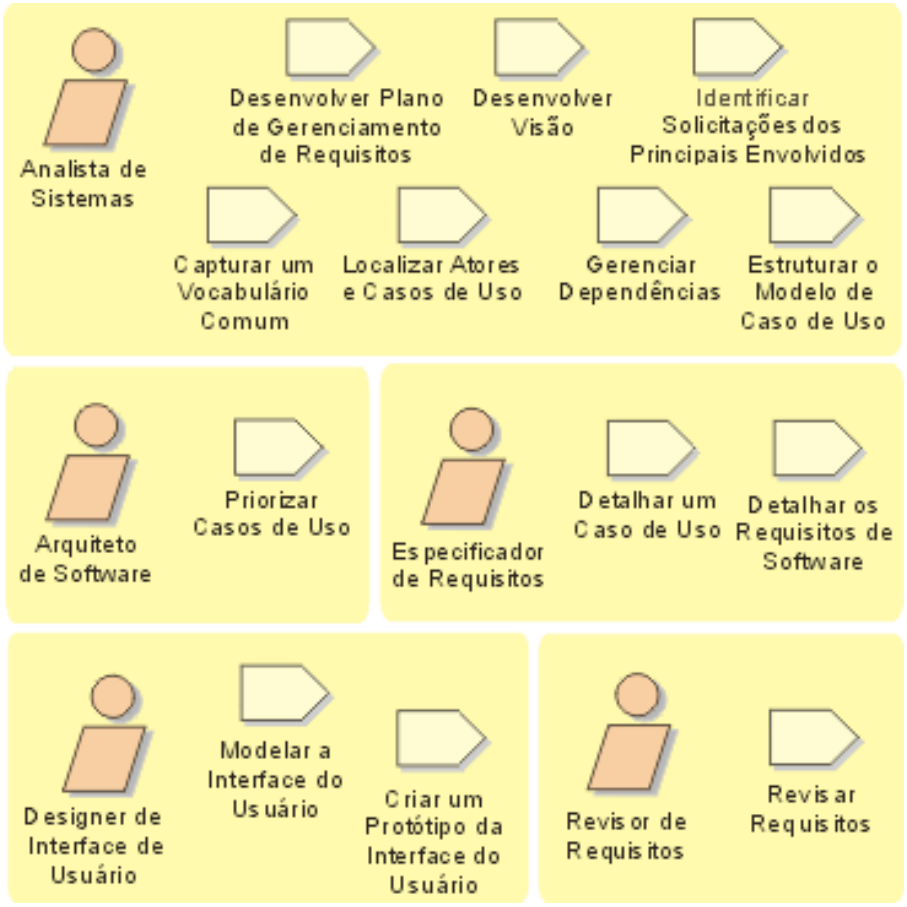
Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models, and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users, and installed in their workplace.
Configuration and change management	This supporting workflow manages changes to the system (see Chapter 25).
Project management	This supporting workflow manages the system development (see Chapters 22 and 23).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

merville

Workflows = Disciplinas

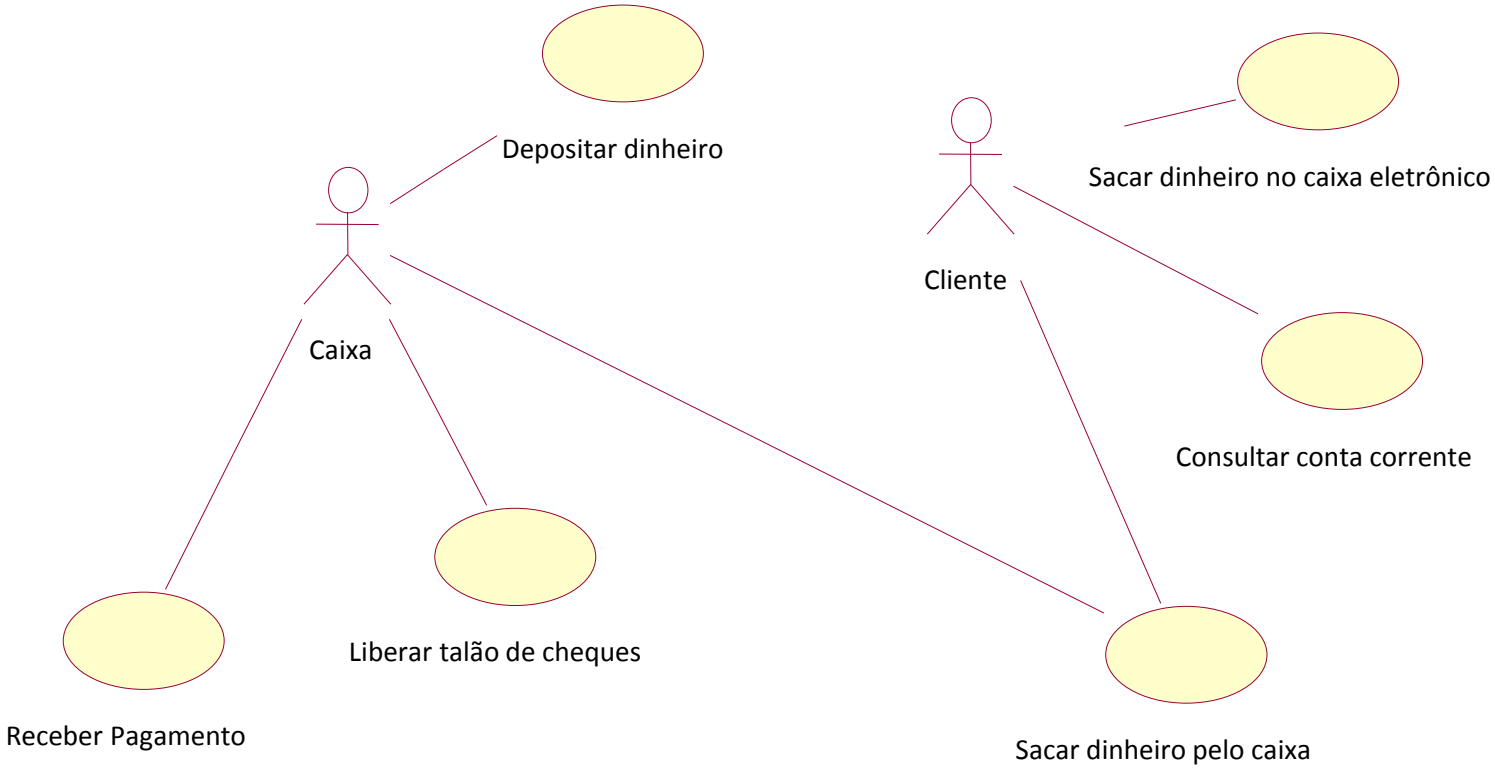
- Pessoas (papéis)
  - Quem?
- Atividades
  - Como?
- Artefatos
  - O quê?
- Fluxo de Trabalho
  - Coordenação dos elementos precedentes no tempo

Exemplo: Requisitos - Papéis e Atividades

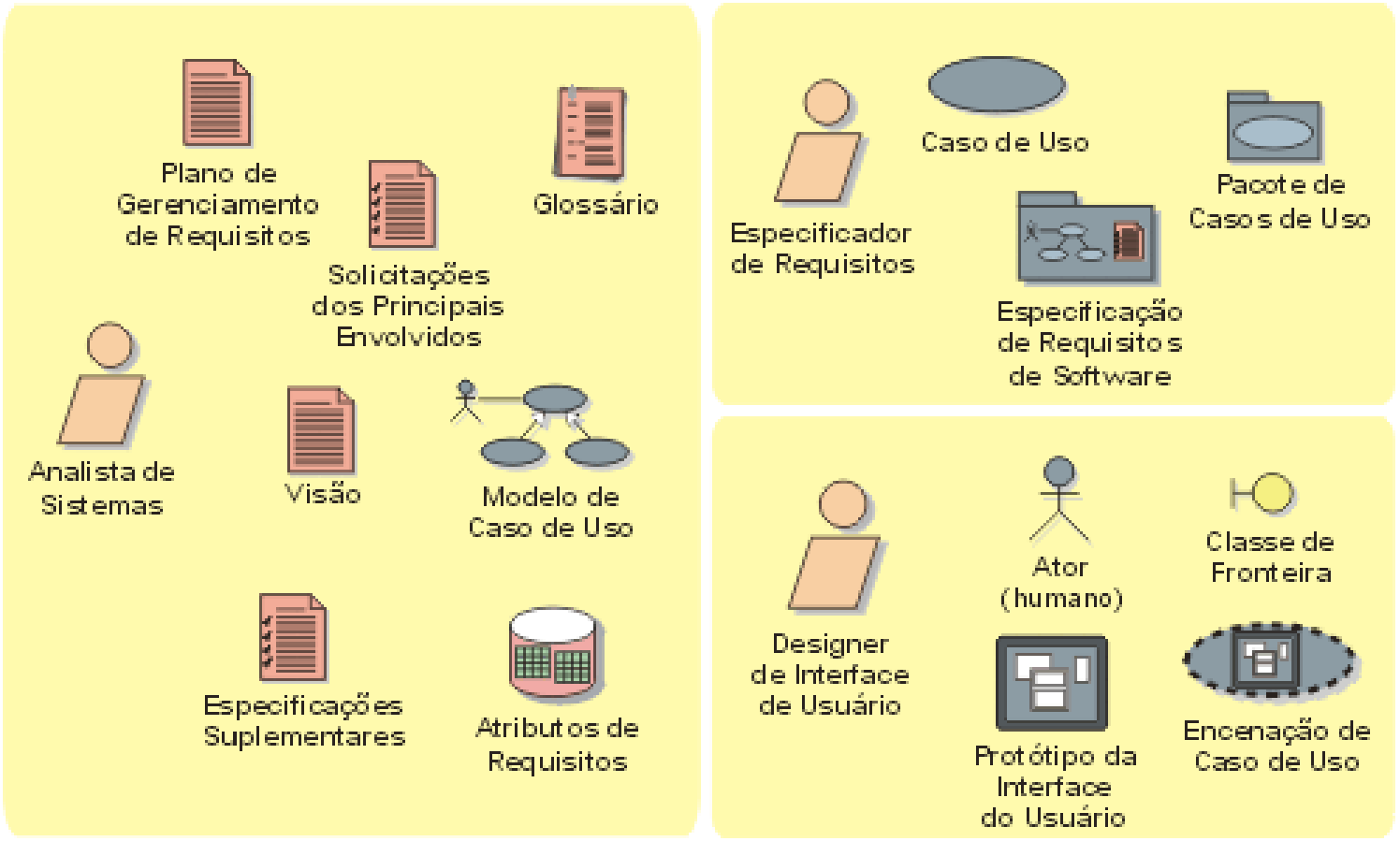


Requisitos: Modelo de Caso de Uso

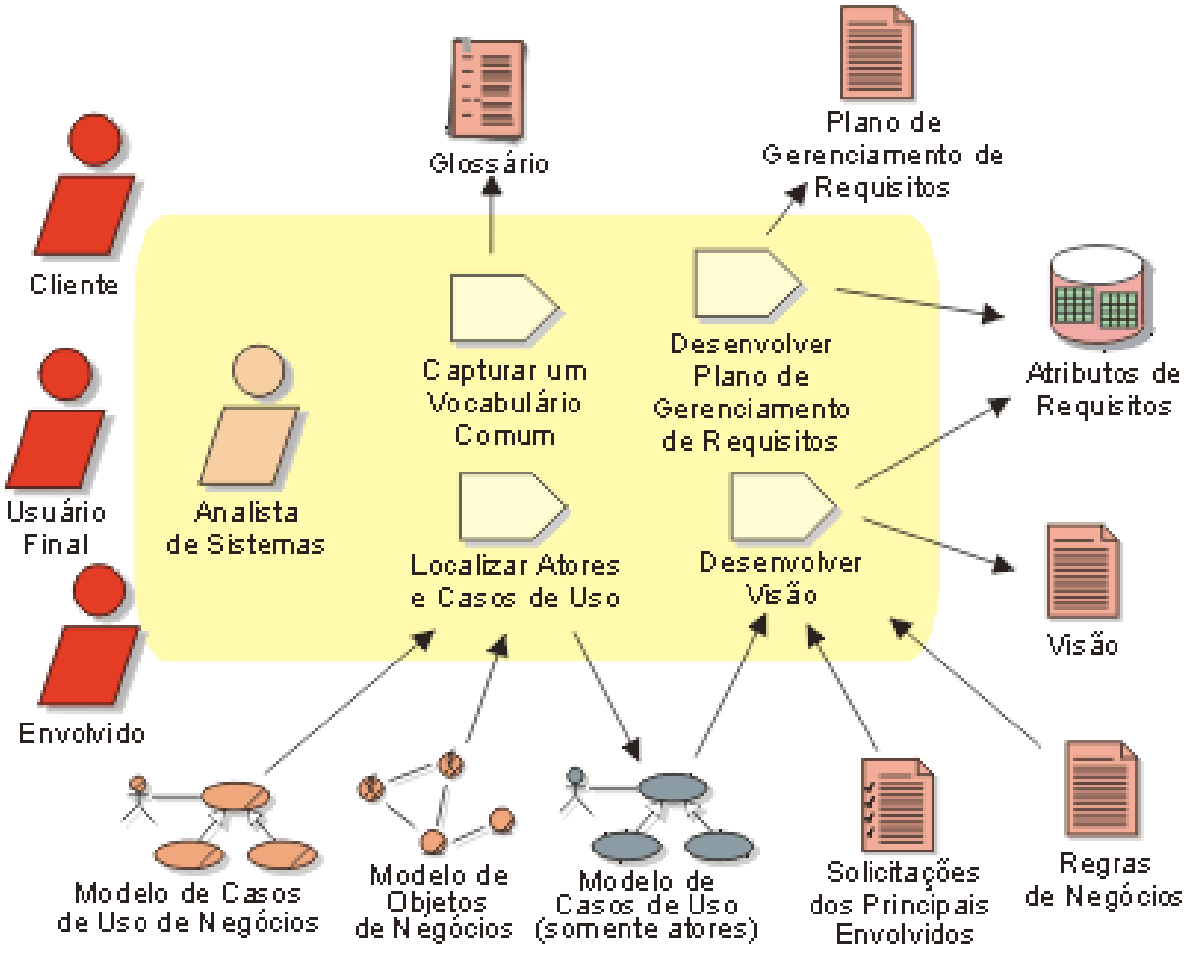
- Identificar Atores e Casos de Uso



Requisitos: Artefatos



Fluxo de Trabalho: Analisar o Problema



Boas Práticas do RUP

- **Desenvolvimento Iterativo e Incremental**
  - Fases
    - Diferentes momentos do projeto, com diferentes níveis de especificação, risco, planejamento e tipo de disciplina predominante
  - Iterações
    - Cada iteração gera versões de vários artefatos que, juntos, constituem a “baseline”
    - Baseline é o conjunto de artefatos revisados e aprovados
      - Fornecem uma base consolidada para a continuação
      - somente pode ser alterada através de um procedimento formal
  - Incremental
    - Todo tipo de artefato, inclusive código

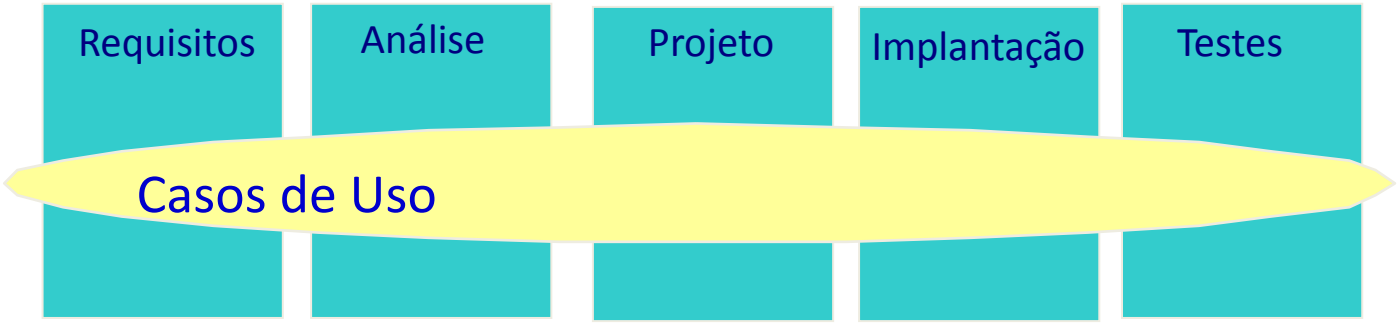
Boas Práticas do RUP

- **Gerenciamento de requisitos**
  - Documentar requisitos do cliente explicitamente e rastrear mudanças nesses requisitos
  - Analisar o impacto de mudanças no sistema antes de aceitá-las
- **Uso de arquiteturas baseadas em componentes**
  - Estruturar o sistema em componentes

Boas Práticas do RUP

- **Modelar o software visualmente**
  - Usar modelos de UML gráficos para representar visões estáticas e dinâmicas do software
- **Verificar a qualidade do software**
  - Assegurar que o software esteja de acordo com padrões organizacionais
- **Controlar mudanças no software**
  - Gerenciar mudanças no software usando um sistema de gerenciamento de mudança e procedimentos e ferramentas de gerenciamento de configuração

RUP: Orientado a Casos de Uso

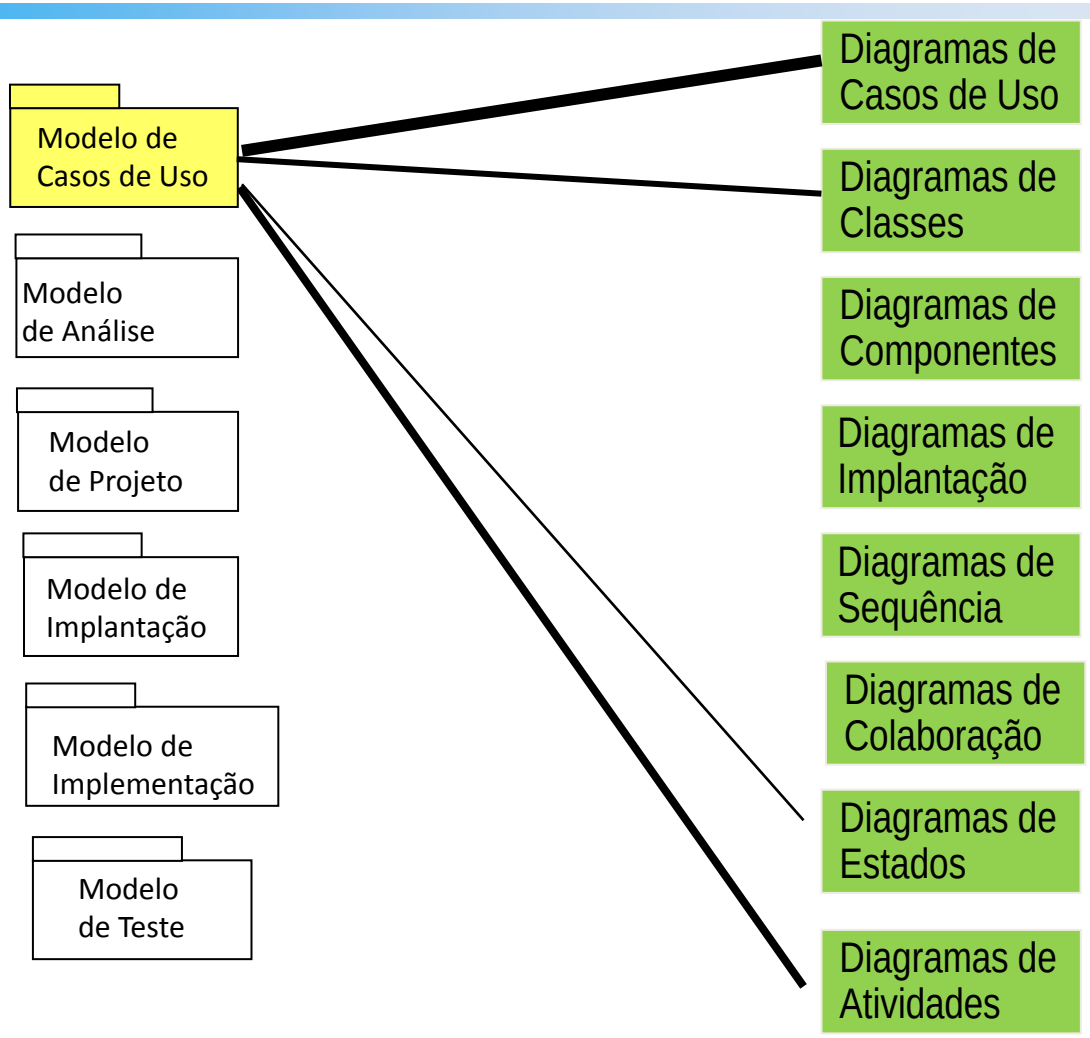


- Caso de uso: funcionalidade completa como externamente percebida pelo usuário
- Orienta uma série de atividades de desenvolvimento
  - Criação e validação da arquitetura do sistema
  - Definição dos casos de teste e procedimentos
  - Planejamento das iterações
  - Criação da documentação do usuário
  - Implantação do sistema

RUP: Orientado a Modelos

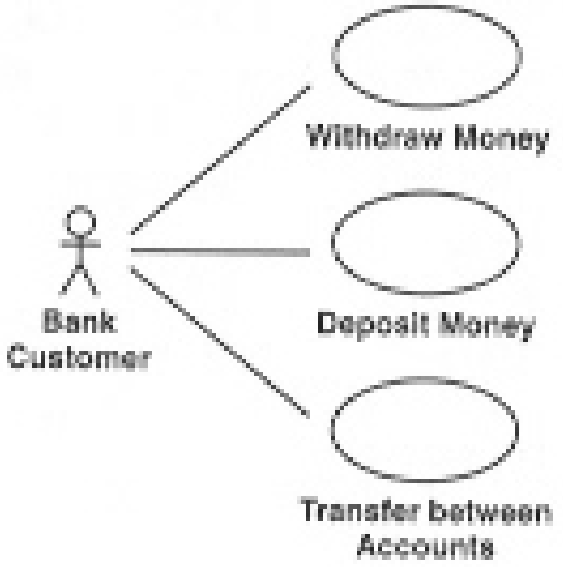
- Modelo
  - Abstração do sistema segundo um certo ponto de vista e nível de abstração
    - Ponto de vista
      - Modelo de projeto vs. Modelo de Casos de Uso
      - Destinados a diferentes pessoas, com diferentes missões
    - Semanticamente autocontido
      - "usuário" do modelo não necessita de informação complementar para interpretá-lo
- Manter relacionamentos entre modelos
- Modelo é descrito através de um conjunto de diagramas e documentação associada
  - RUP: notação é a UML
- Modelo ≠ Diagrama
  - Modelo é composto de um ou mais diagramas, possivelmente de tipos diferentes
  - Um diagrama é apenas um desenho

Diagrama vs. Modelo



Breve tour pelo RUP

Requisitos: o começo de tudo!!



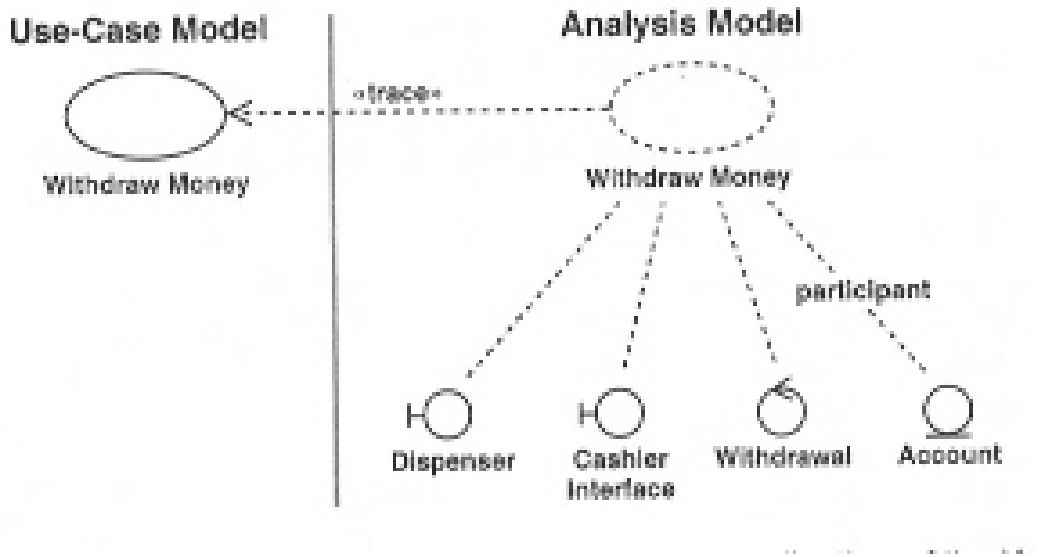
Modelo de Casos de Uso

Breve tour pelo RUP

Iniciando a Solução...

"Temos que identificar em nossos requisitos, quais são os elementos essenciais para satisfazê-los..."

Modelo de Análise

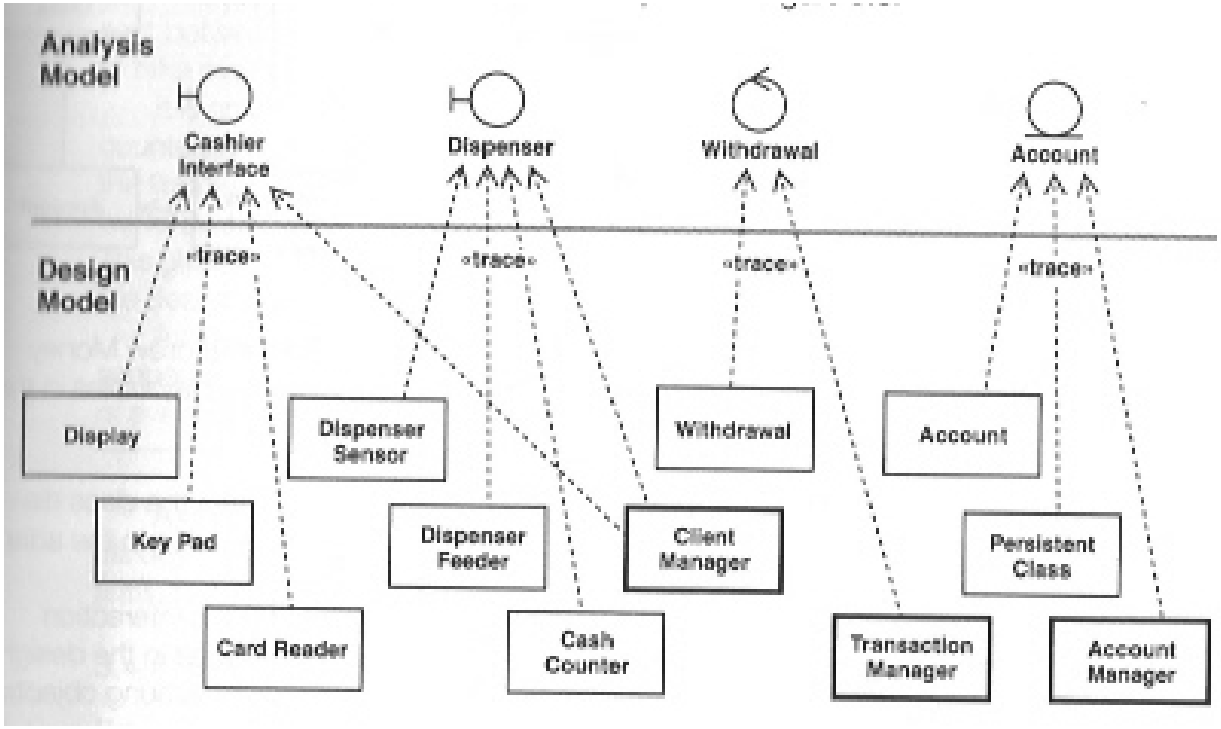


Breve tour pelo RUP

Sedimentando a Solução...

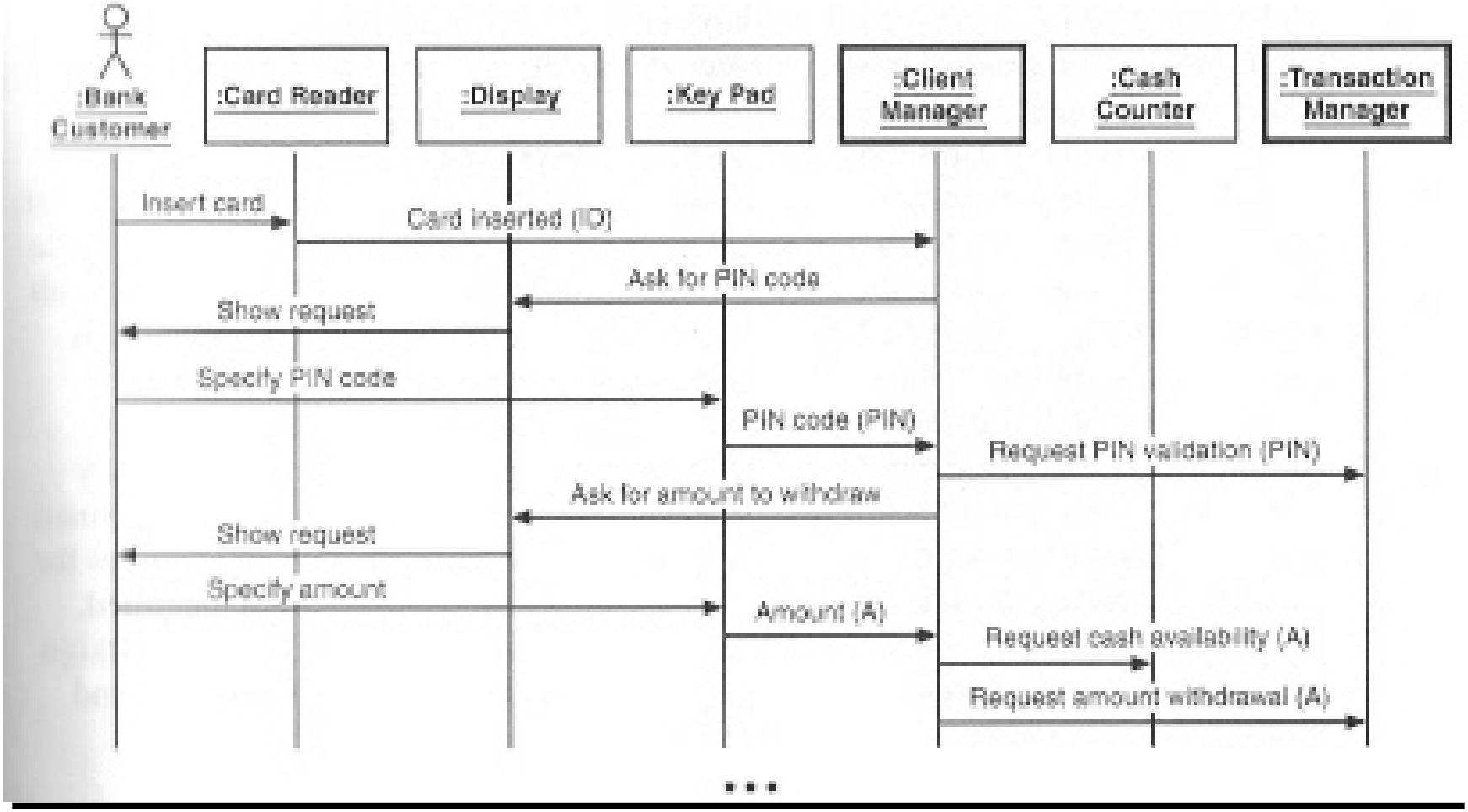
"A partir dos elementos essenciais, precisamos definir estratégias para satisfazê-los incluindo suas restrições..."

Modelo de Projeto



Breve tour pelo RUP

Modelo de Projeto



Breve tour pelo RUP

- Garantindo qualidade

```
public class Account {
  private int balance;
  /*@ invariant balance>=0 @*/
  ...
  void debit(int amount) {
   /*@ requires amount <= balance @*/
   /*@ ensures balance = \old(balance) - amount @*/
  }
...
}
```

Breve tour pelo RUP

- “Com a solução definida, o próximo passo é operacionalizá-la”

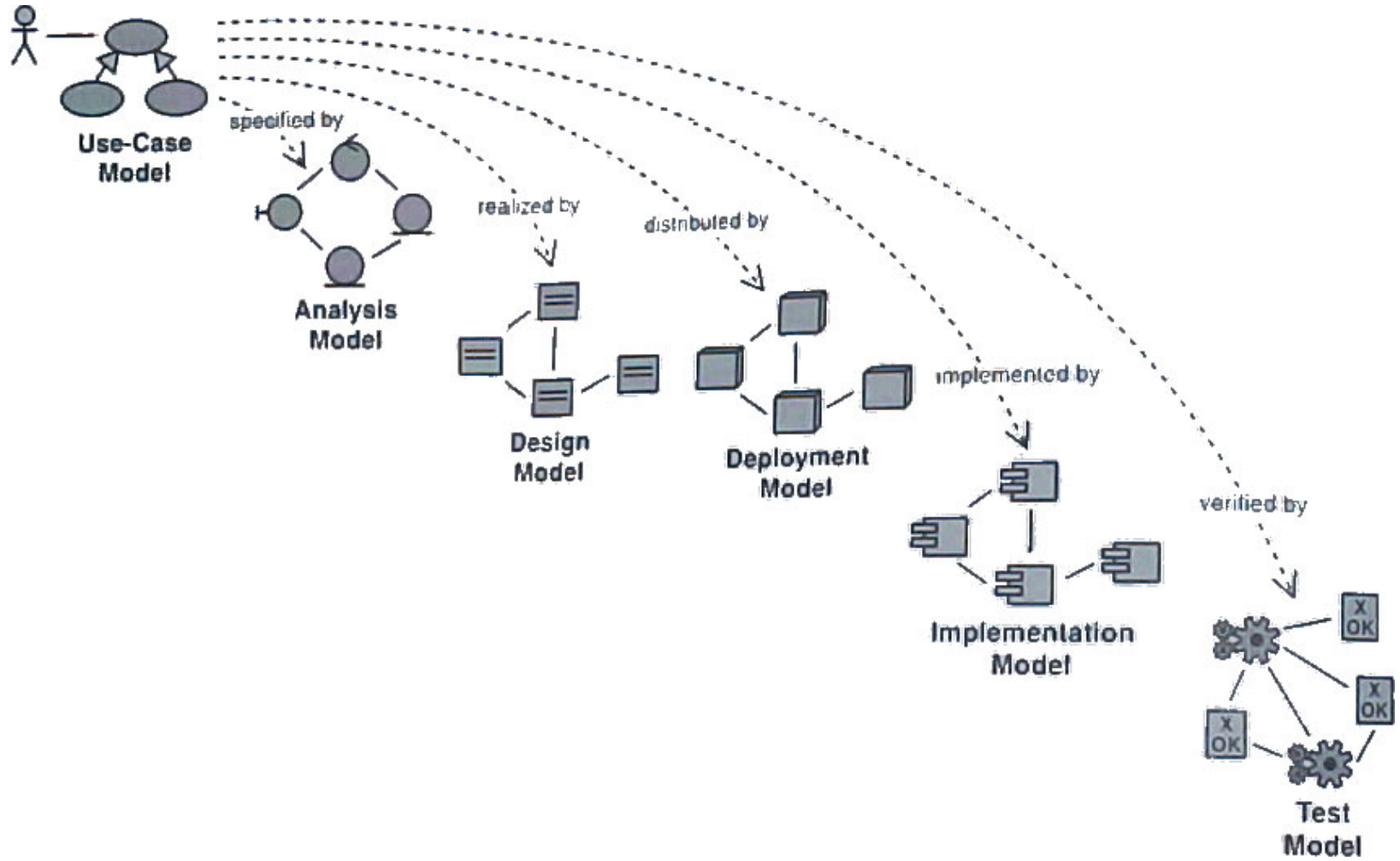
```
public class Account {
  private int balance;
  ...
  void debit(int amount) {
    if(amount<=balance)
       balance = balance – amount;
    else throw
        new AccountException(“...”);
  }
...
}
```

Breve tour pelo RUP

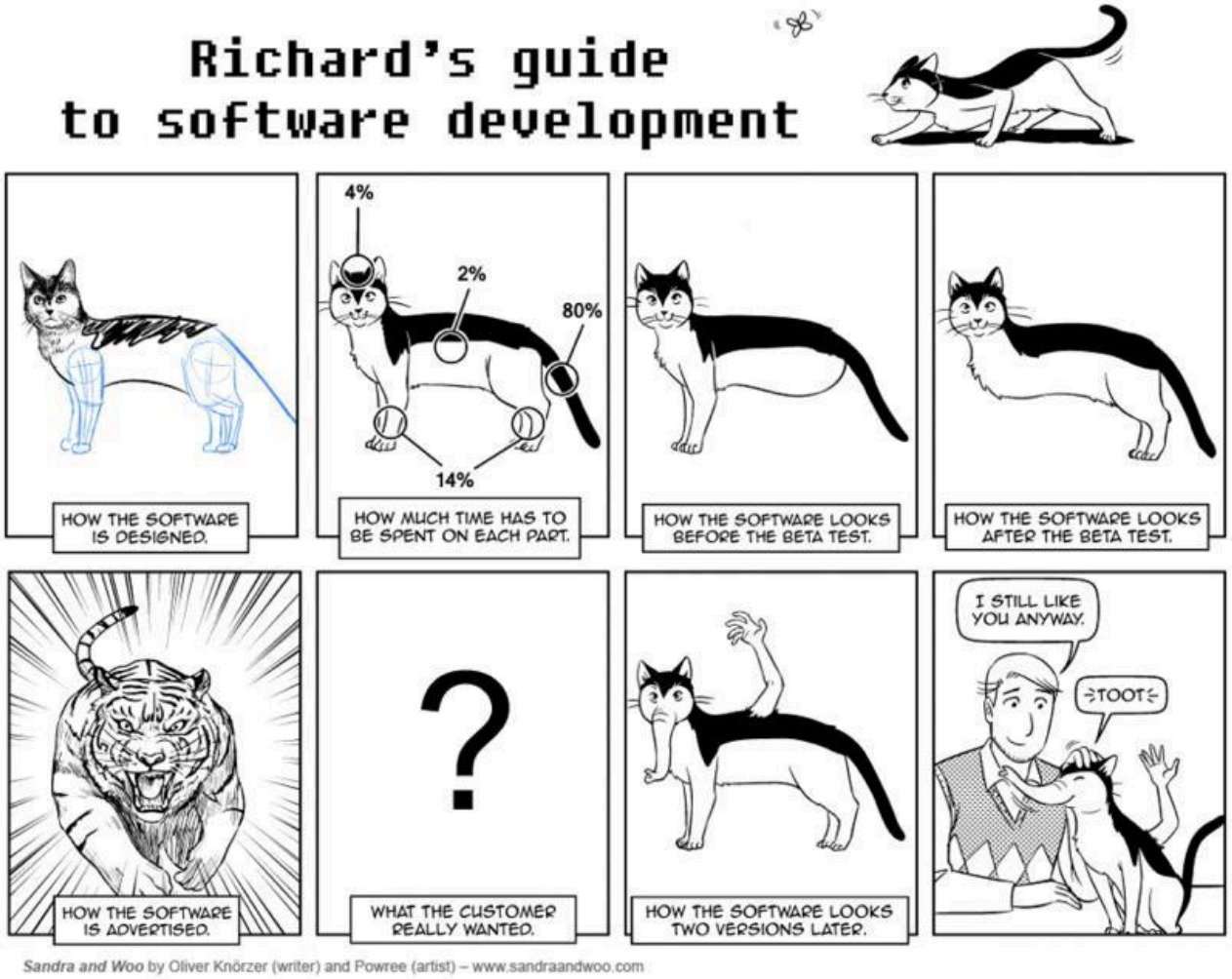
- “Funciona?? Com a implementação feita, podemos então executar os testes !!!”

```
public class AccountTest extends
TestCase {

  void testDebit() {
    Account acc = new Account(10);
    acc.debit(10);
    assertEquals(0, acc.getBalance());
  }

}
```

RUP: Relacionamentos entre Modelos



Descontraindo



Referências

- **Leitura Obrigatória**
  - **Sommerville, I. Engenharia de software, 9a edição. Pearson, 2011.**
    - **Capítulo 2**
  - **Pressman, Roger. Engenharia de Software: Uma Abordagem Profissional, 7a edição. McGraw-Hill, 2011.**
    - **Capítulo 2**

Perguntas?

- Este material tem contribuições de
  - Ingrid Nunes
  - Karin Becker
  - Lucinéia Thom
  - Marcelo Pimenta

Prosoft

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf
INSTITUTO DE INFORMÁTICA
UFRGS