Processo de Desenvolvimento de Software – Modelos

Prof. Ingrid Nunes

INF01127 - Engenharia de Software N

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

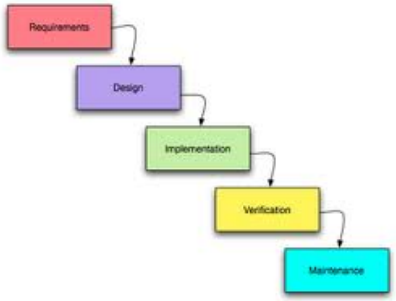
Prosoft

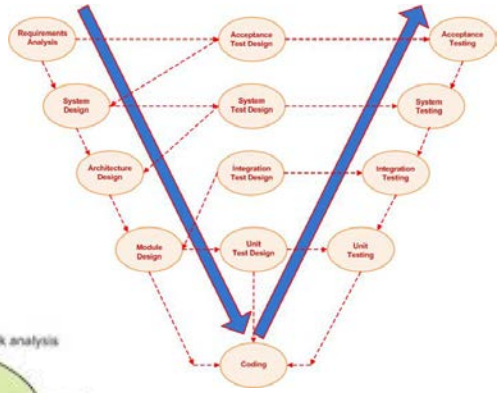
.inf
INSTITUTO DE INFORMÁTICA
UFRGS

Processo de Desenvolvimento de SW

Atividades Genéricas

Modelos de Processo





Processo de Desenvolvimento de Software

MODELOS DE PROCESSO DE SOFTWARE

Modelo de Processo de Software

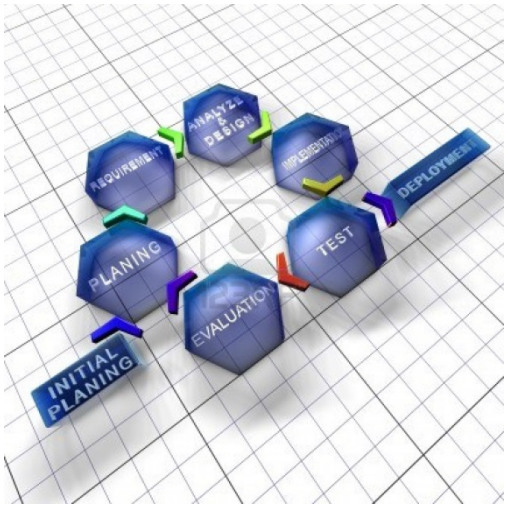
- Representação
  - **Simplificada** de um processo de software
  - **Abstrata** das características marcantes de um ou mais processos específicos

"Família de Processos"

Modelo de Processo de Software

Sommerville

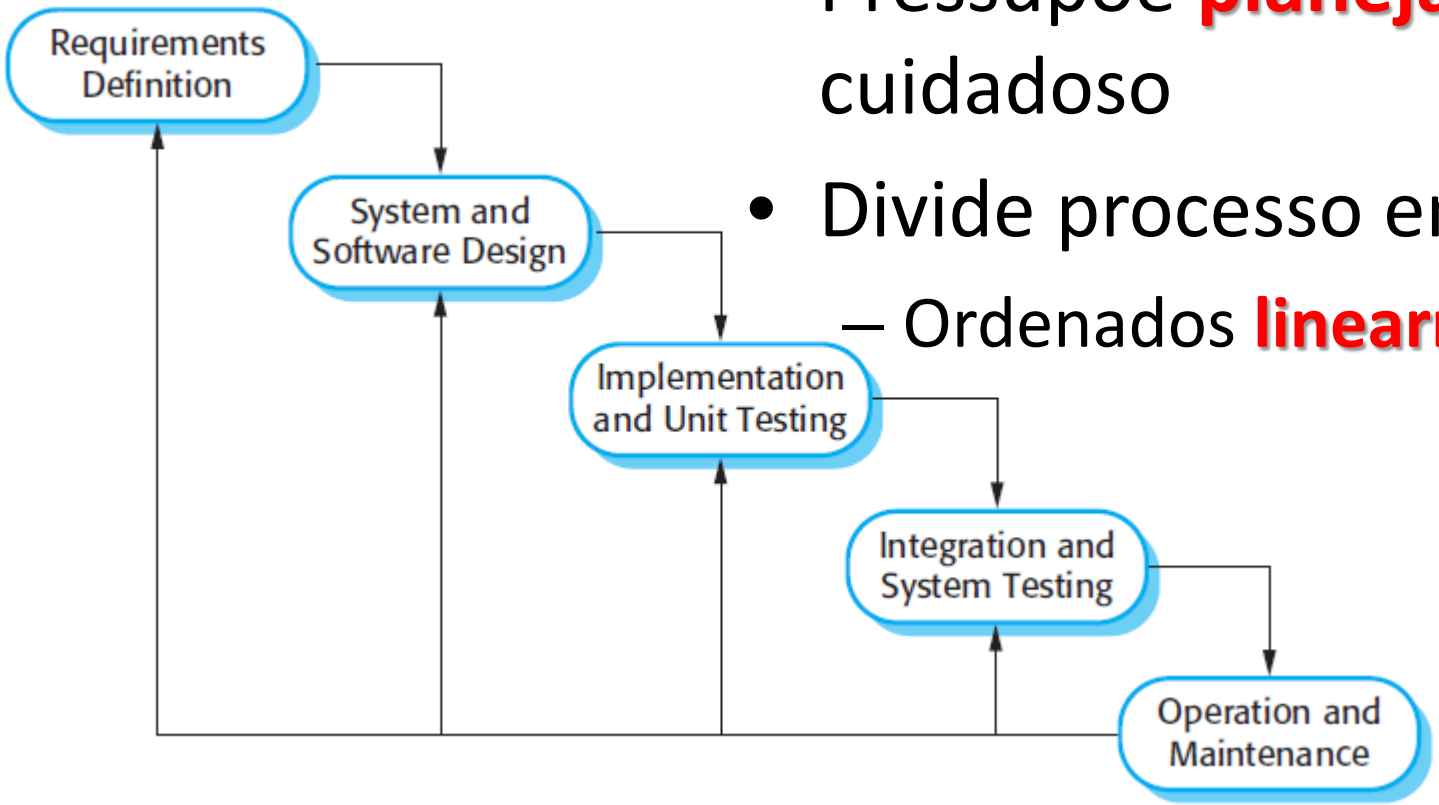
- Modelo em Cascata
- Modelo Incremental
- Engenharia de Software Orientada a Reuso



Pressman

- Modelos Prescritivos
  - Modelo em Cascata
    - Modelo em V
  - Modelos Incremental
  - Modelo Evolucionário
    - Prototipação
    - Modelo Espiral
  - Modelo Concorrente
- Modelos Especializados
  - Baseado em Componentes
  - Métodos Formais
  - Orientado a Aspectos

Modelo em Cascata (Waterfall)



- Histórico: NATO 1968
- Pressupõe **planejamento** cuidadoso
- Divide processo em **fases**
  - Ordenados **linearmente**

Modelo em Cascata (Waterfall)

- Resultados de uma fase tornam-se **entradas** da **próxima**
  - Qualquer ordenamento diferente resulta num produto inferior
- Resultados de cada fase devem ser **analisados** e **aprovados** para que se passe à fase seguinte
  - Certificar-se que resultados intermediários são consistentes com a entrada e com requisitos do sistema
  - Muito laços para fases anteriores

- **Requisitos do usuário são "congelados" antes do início do projeto**

Modelo em Cascata (Waterfall)

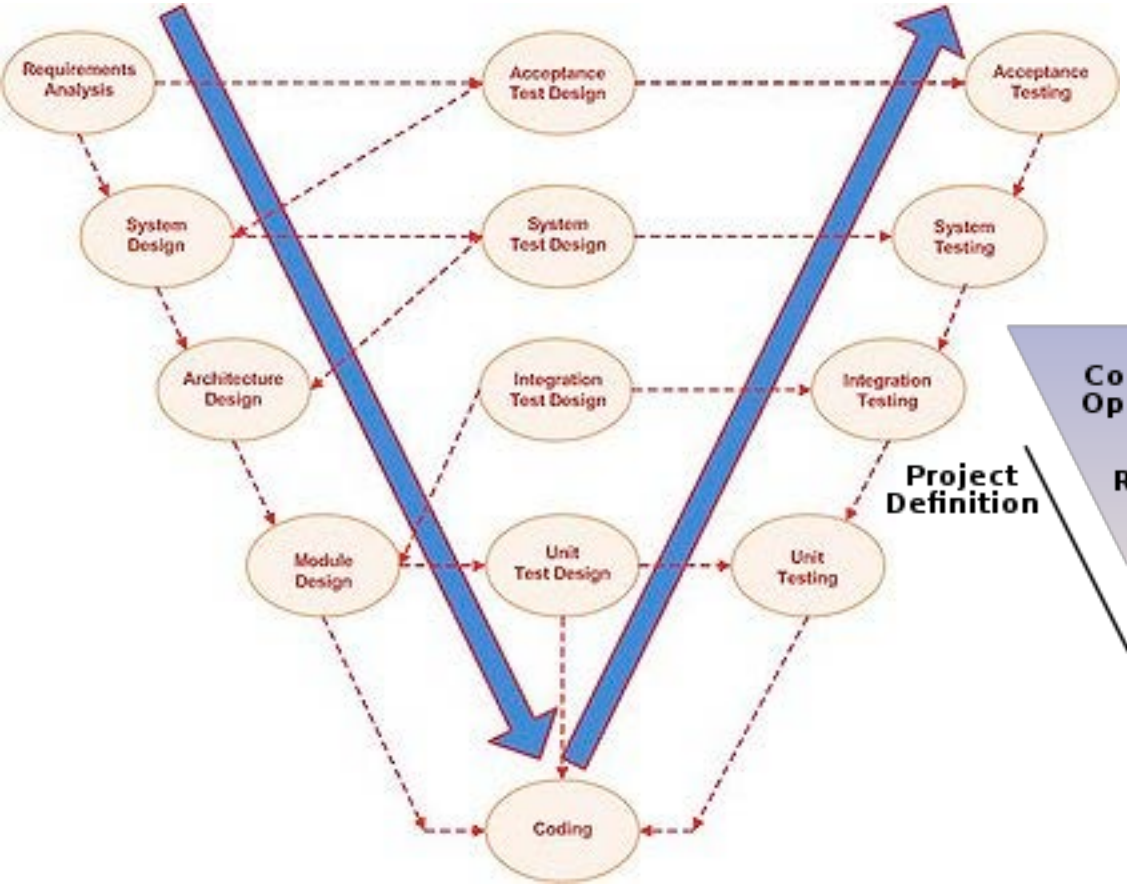
- Particionamento **inflexível** do processo em fases distintas
  - Uma fase deve estar completa para se passar à fase seguinte
- **Custo** do **erro** é muito **grande**, quando descoberto em fases adiantadas
  - Desvios de compreensão são detectados tardiamente
- Permite **pouco feedback** do usuário
  - Requisitos “capturados”
- **Dificuldade** de assimilar e **reagir** a **mudanças** depois de iniciado o processo

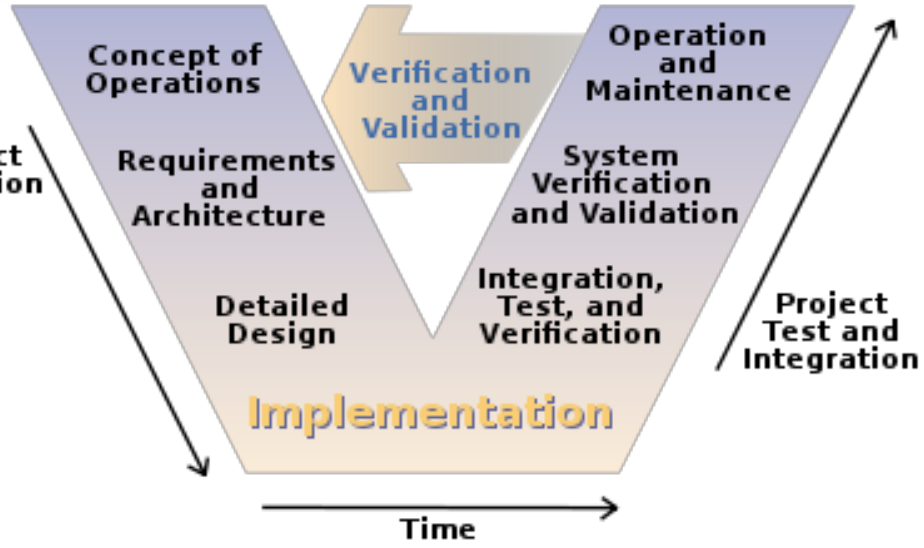
Modelo em Cascata (Waterfall)

- **Grande mérito**
  - Distinguiu o desenvolvimento de software de programação
  - Diferenciou a natureza das diferentes atividades envolvidas
- **Origem de pesquisas** sobre as diversas atividades no desenvolvimento de software
- Apropriado somente quando
  - **Requisitos** estão bem compreendidos e requisitos são **estáveis**
  - Para grande parte de aplicações, **requisitos estáveis é uma utopia**
- Bastante usado em projetos de engenharia de sistemas de grande porte, onde um sistema é desenvolvido em várias localidades

Modelo em Cascata (Waterfall)

- **Variação: V-Model**

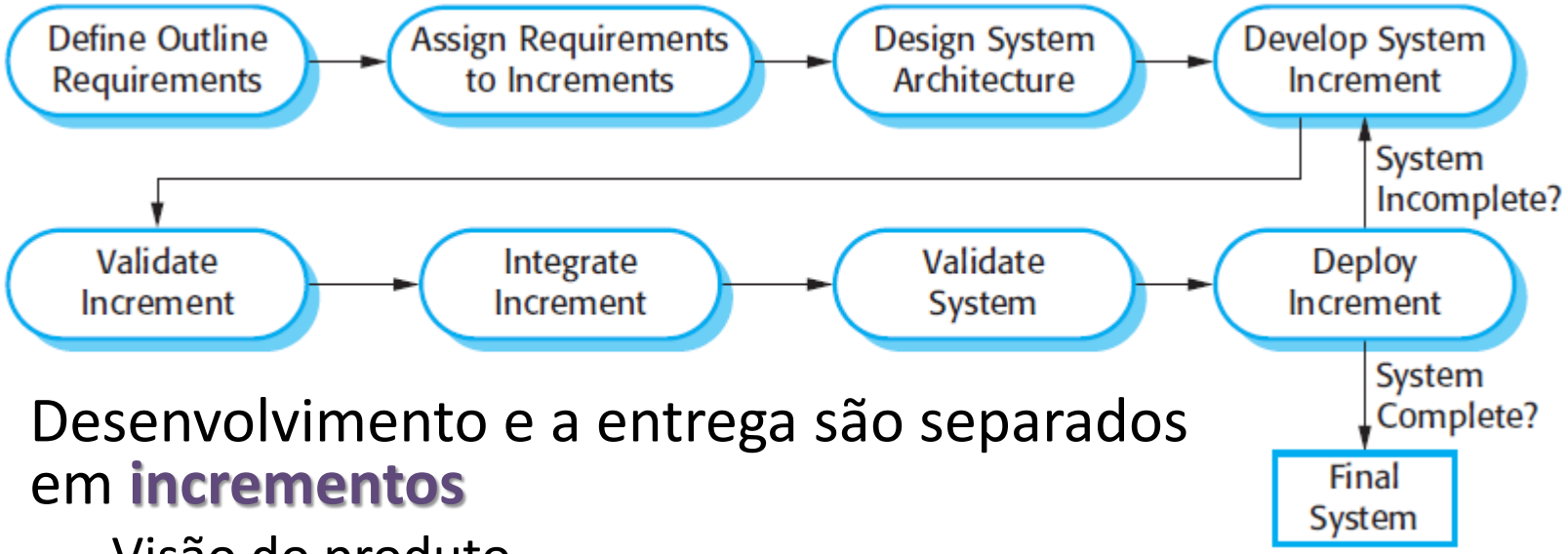




Modelo Iterativo Incremental

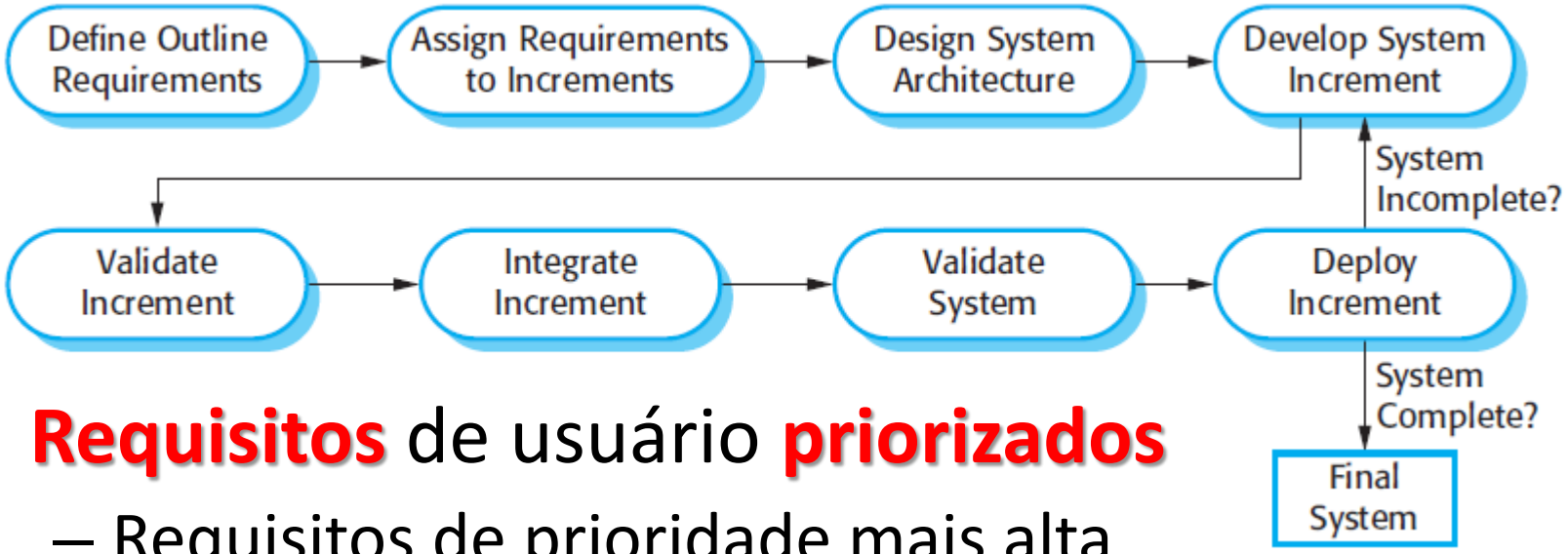
- Organização das atividades em incrementos, desenvolvida ao longo de ciclos
  - **Iteração**
- Cada ciclo visa realizar um conjunto completo de atividades do desenvolvimento
  - Duração fixa ou variável
- Ao fim de cada ciclo
  - (re)planeja-se o próximo ciclo
- Tendências
  - Processo Unificado e suas variantes
    - Heavy weight
  - Métodos Ágeis (XP, SCRUM, Modelagem Ágil, etc.)
    - Light weight

Modelo Iterativo Incremental



- Desenvolvimento e a entrega são separados em **incrementos**
  - Visão do produto
  - Releases
  - Incrementos
- Cada incremento fornece parte da funcionalidade solicitada
  - Construir sobre incremento anterior
  - Incremento: Código e/ou modelos de software

Modelo Iterativo Incremental

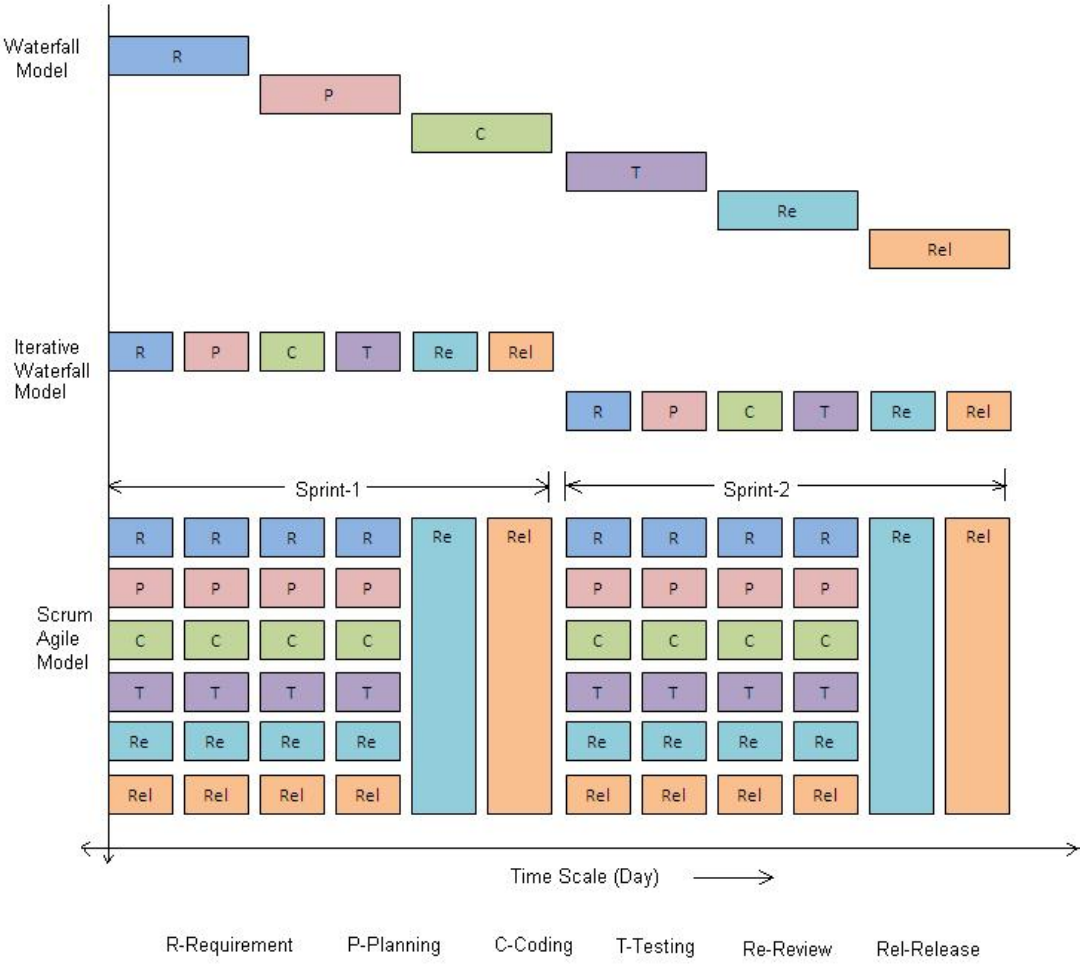


- **Requisitos** de usuário **priorizados**
  - Requisitos de prioridade mais alta incluídos nos incrementos iniciais
- Cada ciclo pode usar uma **abordagem interna**
  - Em cascata (requisito bem entendido)
  - Evolutiva (requisito ainda não estável)

Modelo Iterativo Incremental

- **(+) Primeiros serviços** (mais importantes) são entregues **antes** do **término** do projeto
- **(+)** Primeiros incrementos permitem que os **usuários** tomem **conhecimento** do **sistema** concretamente e interfiram no processo em caso de desalinhamentos
- **(+) Diminuição** do **risco** para o projeto como um todo
- **(+)** Como os serviços iniciais são mais **testados**, os usuários irão encontrar menos erros nos serviços críticos do sistema
- **(-) Processo** não é **visível**
- **(-) Degeneração** da estrutura do sistema

Modelo Iterativo Incremental

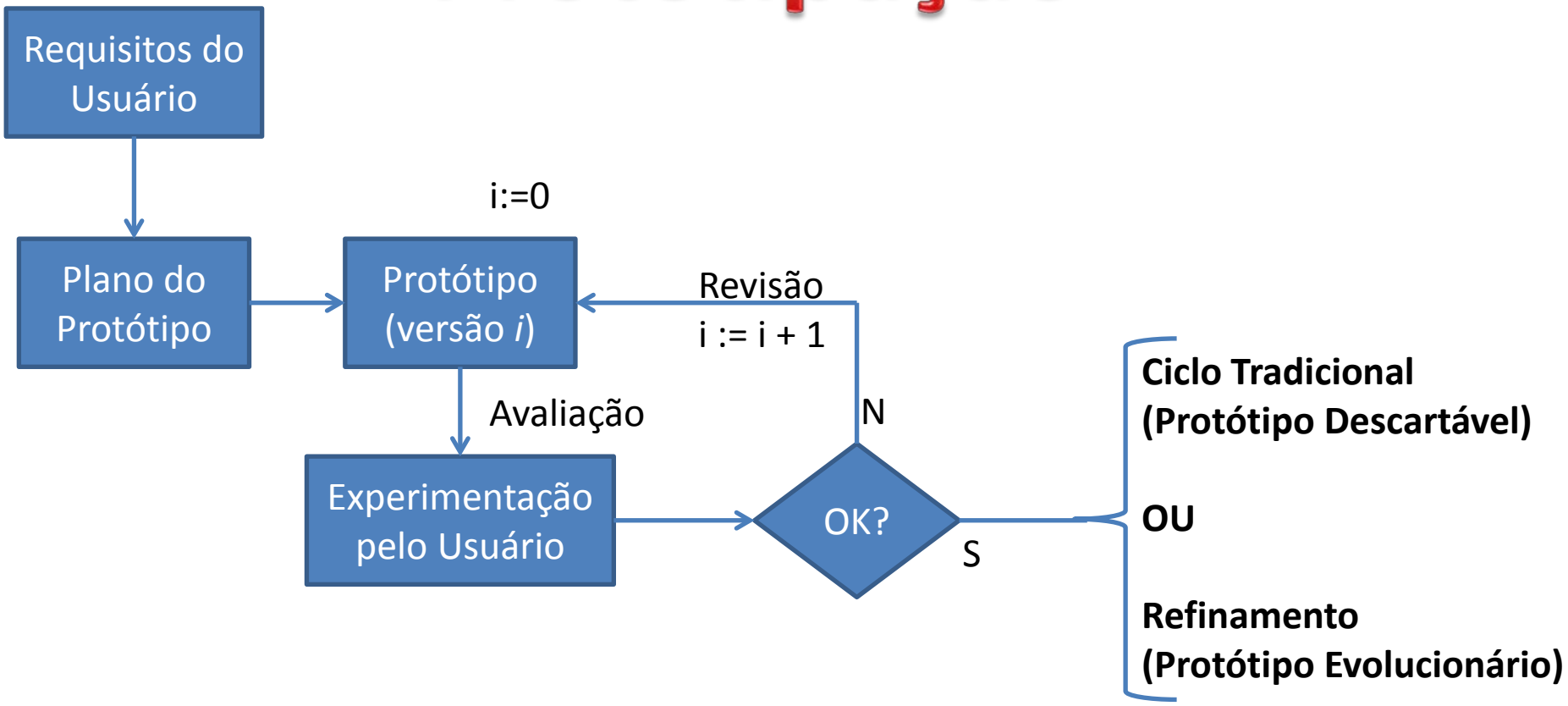


Modelo Evolucionário

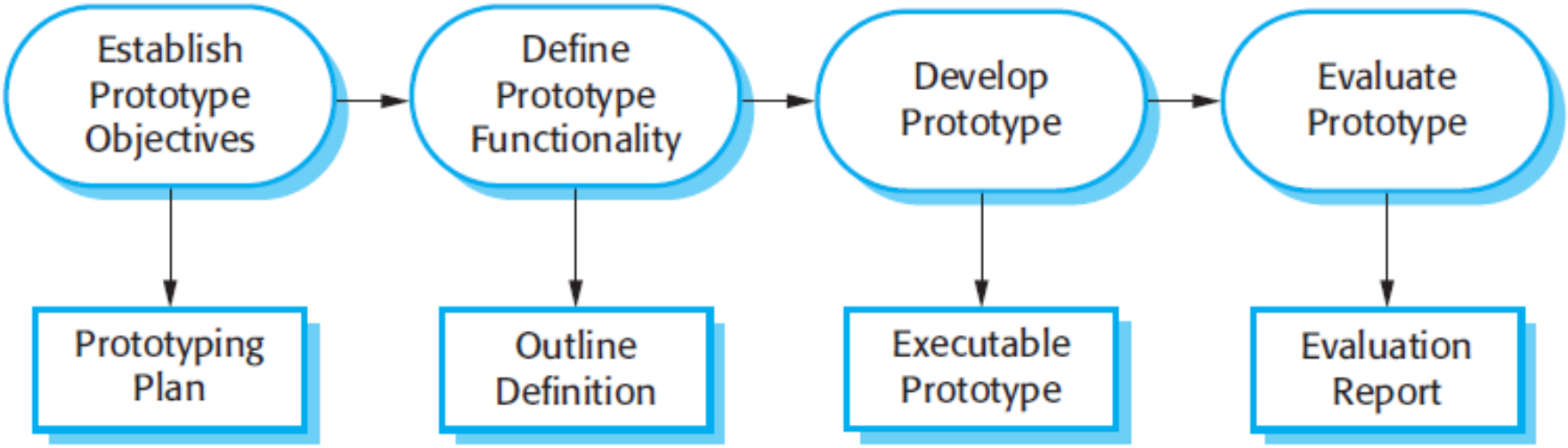
- Prototipação
  - Protótipo: versão inicial de sistema
    - Compreender e validar requisitos
    - Testar idéias
    - Mostrar viabilidade
    - Testar opções de projeto
    - Demonstrar conceitos
    - Projeto de interfaces
  - Rápido e barato
  - Não precisa envolver implementação
  - **Protótipo ≠ Sistema Final**

Modelo Evolucionário

Prototipação



Modelo Evolucionário

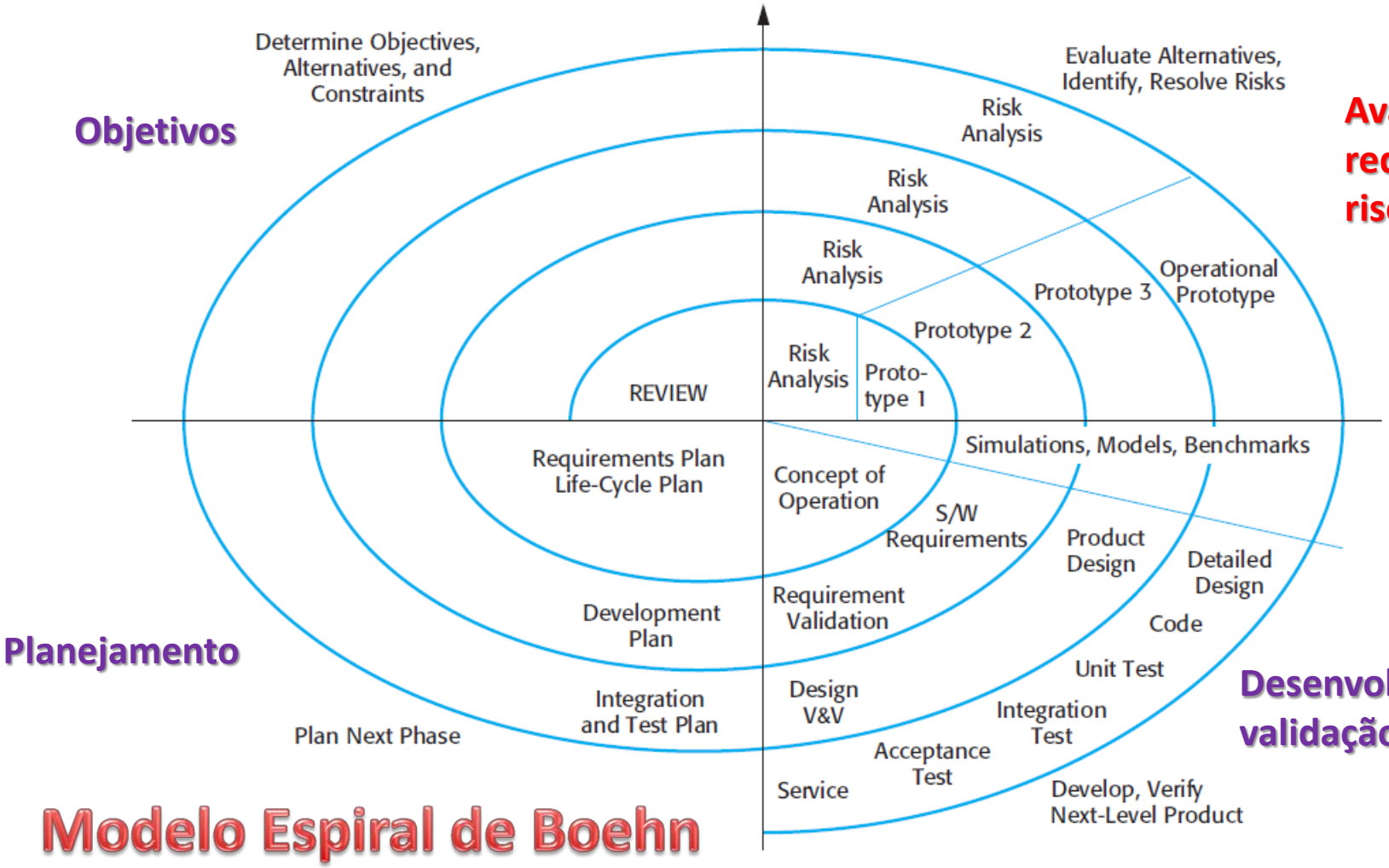


- Protótipo **Vertical** *vs.* **Horizontal**
- Protótipo **Evolucionário** *vs.* **Descartável**

Modelo Evolucionário

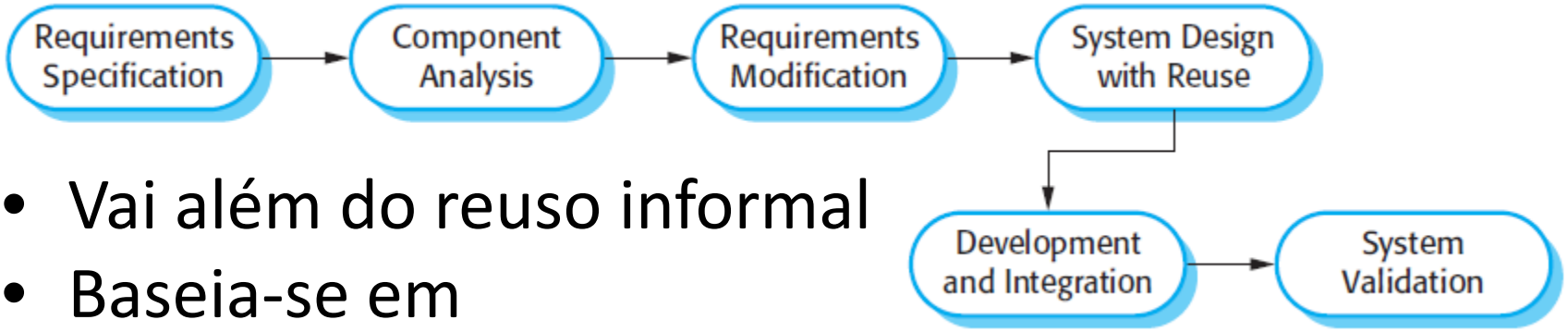
- **Boa ferramenta** para
  - Compreensão de requisitos
  - Discussão de projeto de interface
  - Provas de conceitos, viabilidade técnica
  - Diminuir riscos
- **Difícil** convencer o cliente que é **descartável**
  - Efeito colcha de retalhos
  - Descartável → evolucionário (incremental)
- Problemas de **visibilidade** do **processo**
  - Gerentes verificam o andamento do projeto através dos artefatos gerados
  - Velocidade de entrega de versões executáveis vs. preocupação com aspectos mais abrangentes de software
    - Documentação, critérios de qualidade, testes, etc.

Modelo Evolucionário



Modelo Espiral de Boehn

ES Orientada a Reuso



- Vai além do reuso informal
- Baseia-se em
  - Grande número de **componentes reusáveis**
  - Framework de **integração** destes componentes
- Possui estágio adicionais aos outros modelos
  - Analisar componentes
  - Modificar requisitos
  - Projetar sistema e reuso
  - Desenvolvimento do sistema e integração

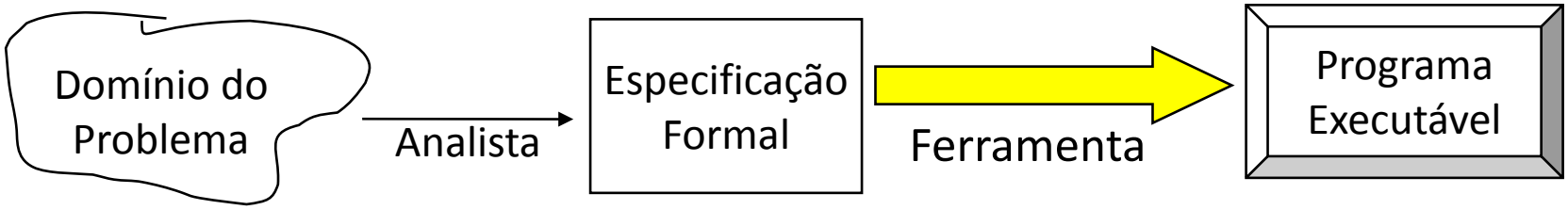
ES Orientada a Reuso

- Exemplos
  - Web Services
  - Coleção de componentes .NET ou J2EE
  - COTs (commercial off-the-shelf)
- (+) Reduz software desenvolvido, custo e riscos
- (+) Entrega mais rápida
- (-) Pode comprometer requisitos
  - Levar ao que o usuário não quer/precisa
- (-) Perda de controle dos componentes controlados pela organização

MODELOS ESPECIALIZADOS

Modelos Formais

- Requisitos são definidos em uma especificação detalhada
  - Uso de notações formais
- Processo de desenvolvimento transformacional



- Transformações são AUTOMÁTICAS suportadas por ferramentas (geradores)
- Analistas criam modelos
- Dispensa o papel de programador
- Modificações sempre feitas na especificação (modelos)

Modelos Formais

- Utiliza **notação matemática** para descrever de maneira precisa as propriedades que um sistema deve ter
- Descrevem "**o que**" o sistema deve fazer sem explicitar "como" será feito
- Uma das notações utilizadas é a **Z**
  - Possibilita decompor uma especificação em pequenos esquemas

Modelos Formais

- Z
  - Esquemas usados para descrever aspectos estáticos e dinâmicos de um sistema
    - **Estáticos**
      - Estados que pode assumir
      - Relacionamentos invariantes quando o estado se altera
    - **Dinâmicos**
      - Operações possíveis
      - Relacionamentos entre as entradas e saídas
      - Mudanças de estado

Modelos Formais

$[NAME, DATE]$.

BirthdayBook

$known : \mathbb{P}\ NAME$

$birthday : NAME \pfun DATE$

$known = \mathrm{dom}\ birthday$

AddBirthday

$\Delta BirthdayBook$

$name? : NAME$

$date? : DATE$

$name? \notin known$

$birthday' = birthday \cup \{name? \mapsto date?\}$

Tipos

State Space (Variáveis, Relações)

State change (pré e pós condições)

$known' = known \cup \{name?\}.$ 

$$
\begin{aligned}
known' &= \mathrm{dom}\ birthday' && \text{[invariant after]} \\
&= \mathrm{dom}(birthday \cup \{name? \mapsto date?\}) && \text{[spec. of } AddBirthday\text{]} \\
&= \mathrm{dom}\ birthday \cup \mathrm{dom}\ \{name? \mapsto date?\} && \text{[fact about 'dom']} \\
&= \mathrm{dom}\ birthday \cup \{name?\} && \text{[fact about 'dom']} \\
&= known \cup \{name?\}. && \text{[invariant before]}
\end{aligned}
$$

Modelos Formais

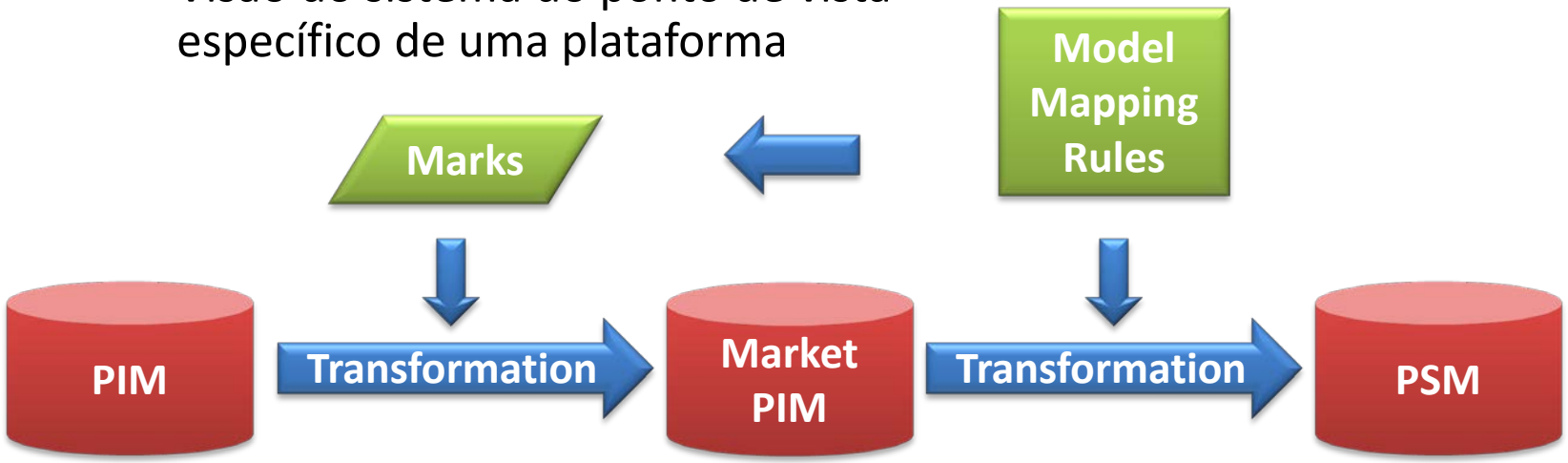
- **(+)** Especificação **compacta**
- **(+) Correção** da especificação pode ser provada
- **(+) Geração automática** de transformações
- **(-)** Necessidade de **habilidades** menos difundidas entre os profissionais
  - Conhecer notações/linguagens formais
  - Formas de verificação de corretude
  - Consistência de uma especificação
- **(-) Dificuldade** de especificar formalmente algumas partes de um sistema
- **(-)** Ainda **carece** de **ferramentas** de apoio ao desenvolvimento
- Aplicabilidade
  - Sistemas críticos
    - Especialmente aqueles em que aspectos de segurança e confiabilidade devem ser verificados ANTES de entrar em operação

Engenharia Dirigida a Modelos

- MDA: Model Driven Architecture
  - Proposta da OMG
  - Construção de sistemas através da confecção de **modelos** que assumam a **posição central** no processo de desenvolvimento
  - **Eleva** o **nível** de **abstração** em todas as etapas do projeto
    - Permitindo aumentar a automação na implementação de sistemas

Engenharia Dirigida a Modelos

- MDA: Modelos
  - **CIM (Computation Independent Model)**
    - Visão do sistema do ponto de vista independente de computação
  - **PIM (Platform Independent Model**
    - Visão do sistema do ponto de vista independente de plataforma
  - **PSM (Platform Specific Model)**
    - Visão do sistema do ponto de vista específico de uma plataforma



Engenharia Dirigida a Modelos

- Transformações
  - São essenciais em MDA
    - PIM → PSM e PSM → código
  - Devem ser bem definidas
    - De preferência, apoiadas por ferramentas
- Objetivos das transformações
  - Otimização
  - Refatoração
  - Geração de código
  - Engenharia reversa
  - Migração

Engenharia Dirigida a Modelos

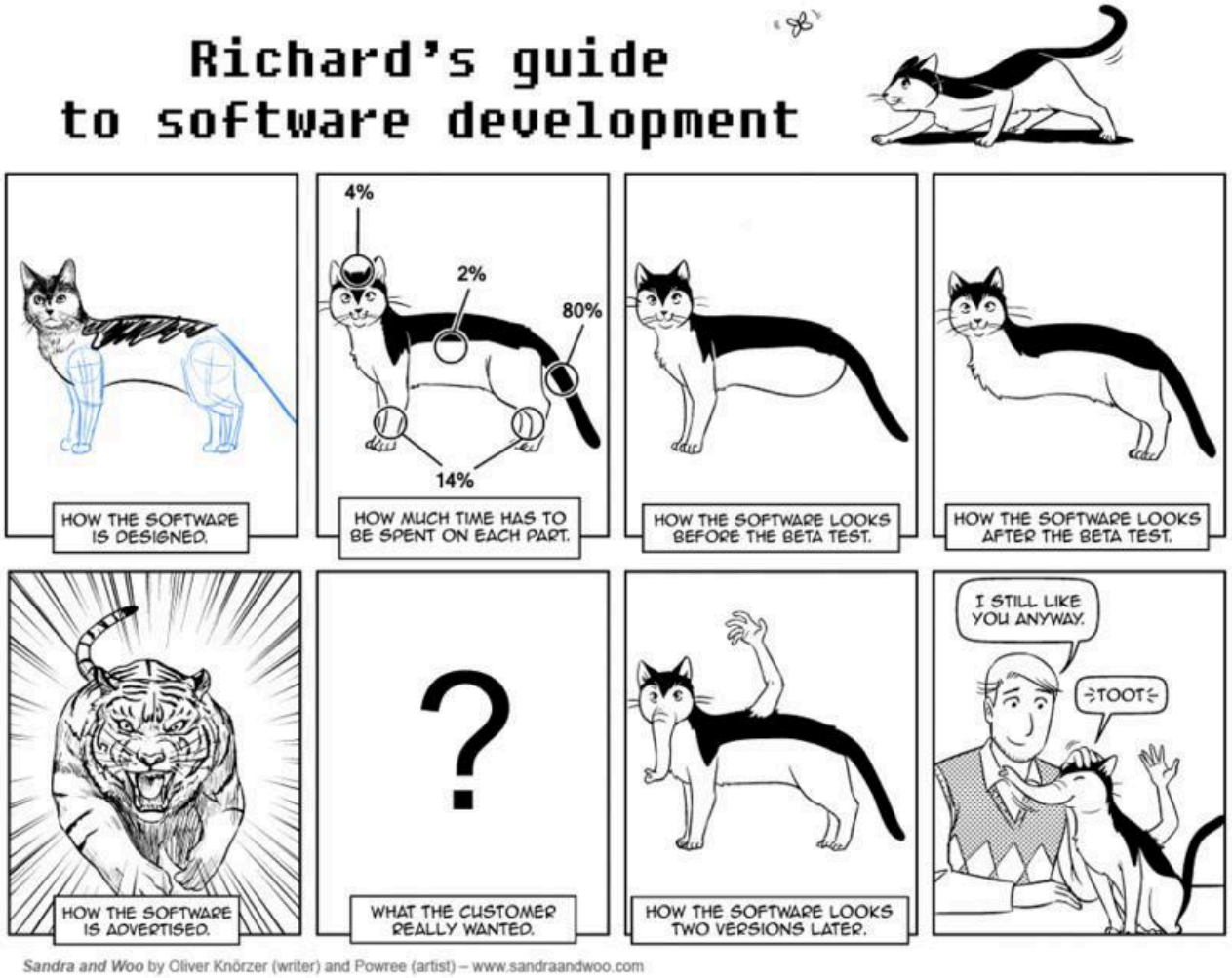
Para cada atributo público nomeAtr: Tipo da classe **nomeClasse** do PIM, criar os seguintes atributos e operações na classe **nomeClasse** do PSM:

- um atributo privado nomeAtr:Tipo
- uma operação pública getAtr():Tipo
- uma operação pública setAtr(atr:Tipo)

Modelos de Processo de Software

- Aspectos comuns a todos modelos
  - Todos começam com (ou visam) uma **compreensão** melhor dos **requisitos** do sistema a ser desenvolvido
  - Todos visam **aumento** da **qualidade** dos resultados parciais e finais
  - Todos visam **melhoria** da **gerência** do processo de desenvolvimento (alguns, o controle do processo; outros, o acompanhamento do processo)
- Discussão
  - Qual o modelo de ciclo de vida adotado no seu ambiente de trabalho?
  - A seu ver, quais as razões para a adoção deste modelo de ciclo?

Descontraindo



Referências

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