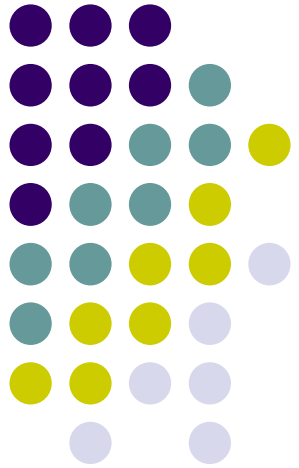
Engenharia de Software I – 2012.2

Modelos de Processo de Software



Ricardo Argenton Ramos

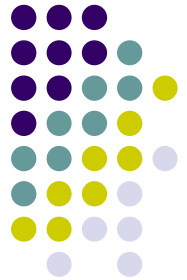
CECOMP

ricargentonramos@gmail.com

A Engenharia de Software

Uma Tecnologia em Camadas

Gerenciamento da Qualidade Total e filosofias similares produzem uma mudança cultural que permite o desenvolvimento crescente de abordagens mais maduras para a Engenharia de Software

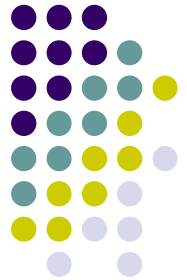


ENGENHARIA DE SOFTWARE

pode ser vista como uma abordagem dedesenvolvimento de software elaborada com disciplina e métodos bem definidos.

.....“a construção por múltiplas pessoas de um software com múltiplas versões” [Parnas 1987]

Modelos de Processo de Software

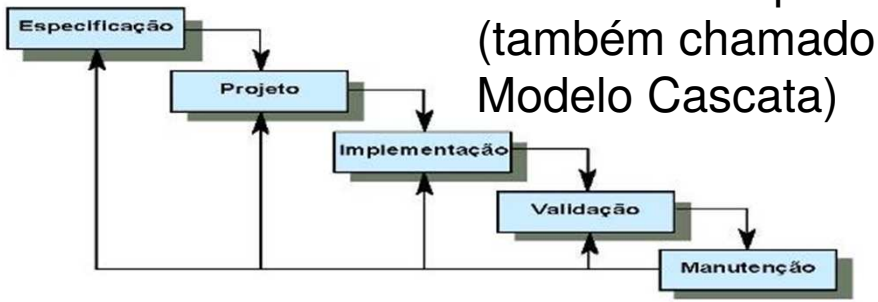


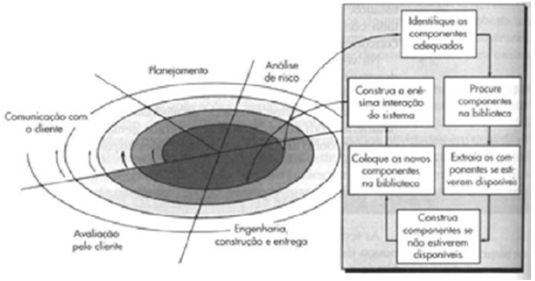
- Existem vários *modelos de processo de software* (ou *paradigmas de engenharia de software*)
- Cada um representa uma tentativa de colocar ordem em uma atividade inerentemente caótica

FACAPE

Modelos de Processo de Software

O Modelo Seqüencial Linear (também chamado Ciclo de Vida Clássico ou Modelo Cascata)

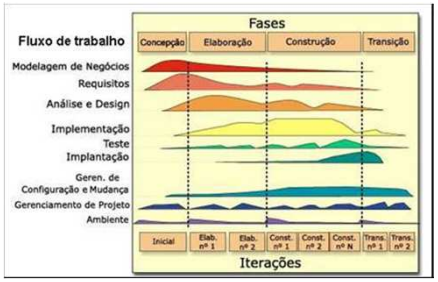




O Modelo Espiral

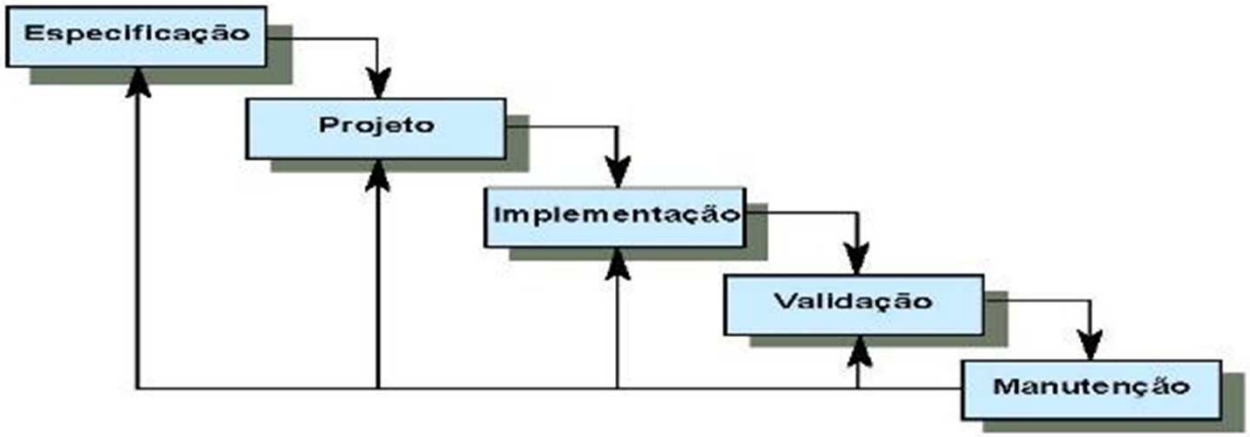
O Modelo Baseado em Componentes

O Paradigma de Prototipação



Processo Unificado

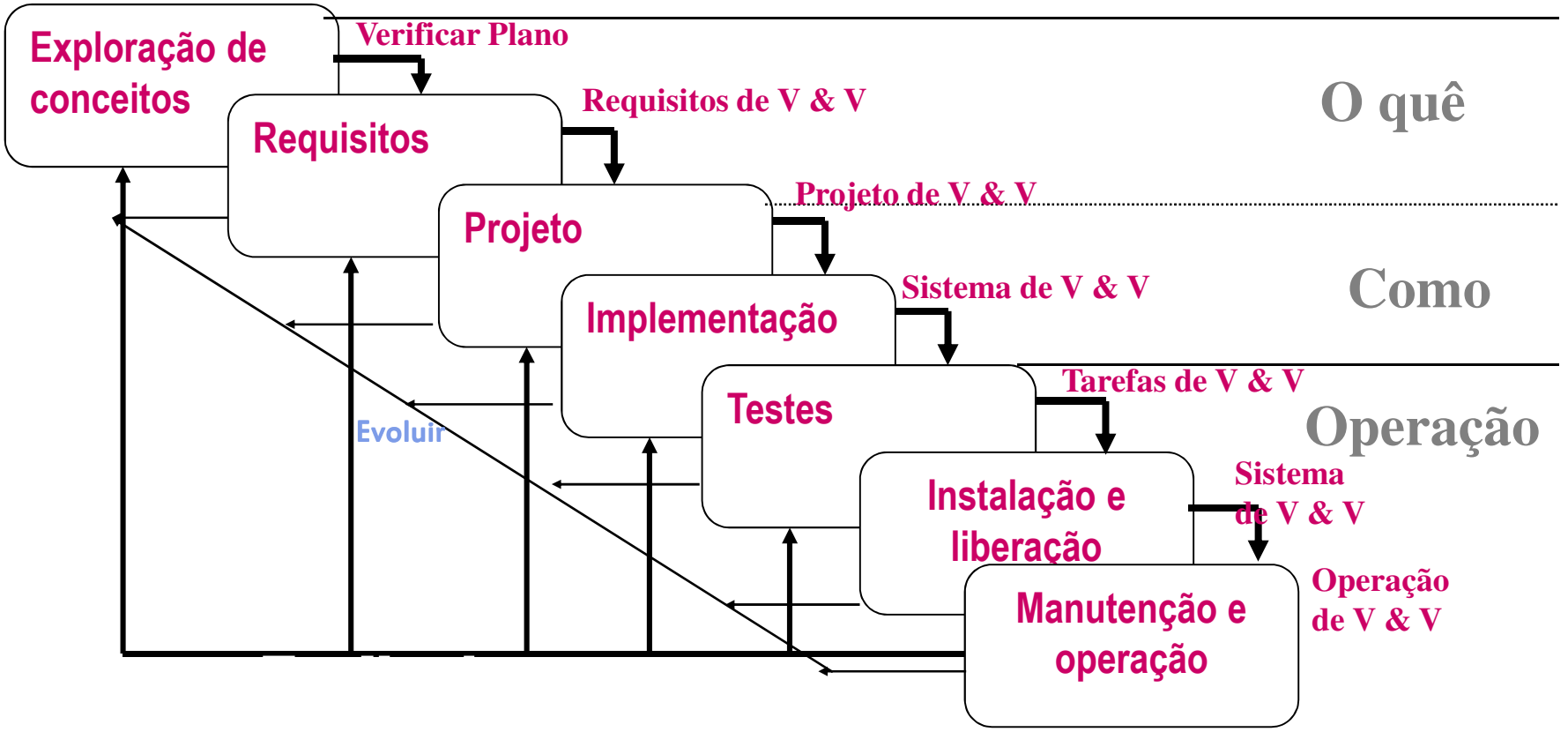
O Modelo Seqüencial Linear (também chamado Ciclo de Vida Clássico ou Modelo Cascata)



O Modelo Cascata

- modelo mais antigo e o mais amplamente usado da engenharia de software
- modelado em função do ciclo da engenharia convencional
- requer uma abordagem sistemática, seqüencial ao desenvolvimento de software
- o resultado de uma fase se constitui na entrada da outra

O Modelo em Cascata

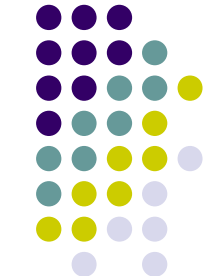


O Modelo em Cascata

Exploração de Conceitos / Informação e Modelagem

- Envolve a elicitação de requisitos do sistema, com uma pequena quantidade de projeto e análise de alto nível;
- Preocupa-se com aquilo que conhecemos como engenharia progressiva de produto de software;
- Iniciar com um modelo conceitual de alto nível para um sistema e prosseguir com o projeto, implementação e teste do modelo físico do sistema.

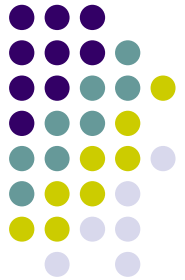
O Modelo em Cascata



Análise de Requisitos de Software

- o processo de elicitação dos requisitos é intensificado e concentrado especificamente no software
- deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos
- os requisitos (para o sistema e para o software) são documentados e revistos com o cliente

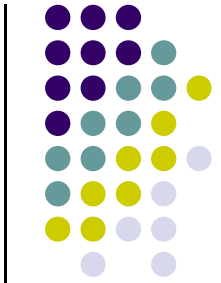
O Modelo em Cascata



Projeto

- tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação inicie

O Modelo em Cascata



Implementação

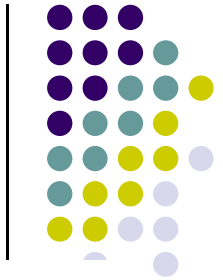
- tradução das representações do projeto para uma linguagem "artificial" resultando em instruções executáveis pelo computador e implementado num ambiente de trabalho.

O Modelo em Cascata

Testes

- Concentra-se:
  - nos aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas
  - nos aspectos funcionais externos, para descobrir erros e garantir que a entrada definida produza resultados que concordem com os esperados.

O Modelo em Cascata

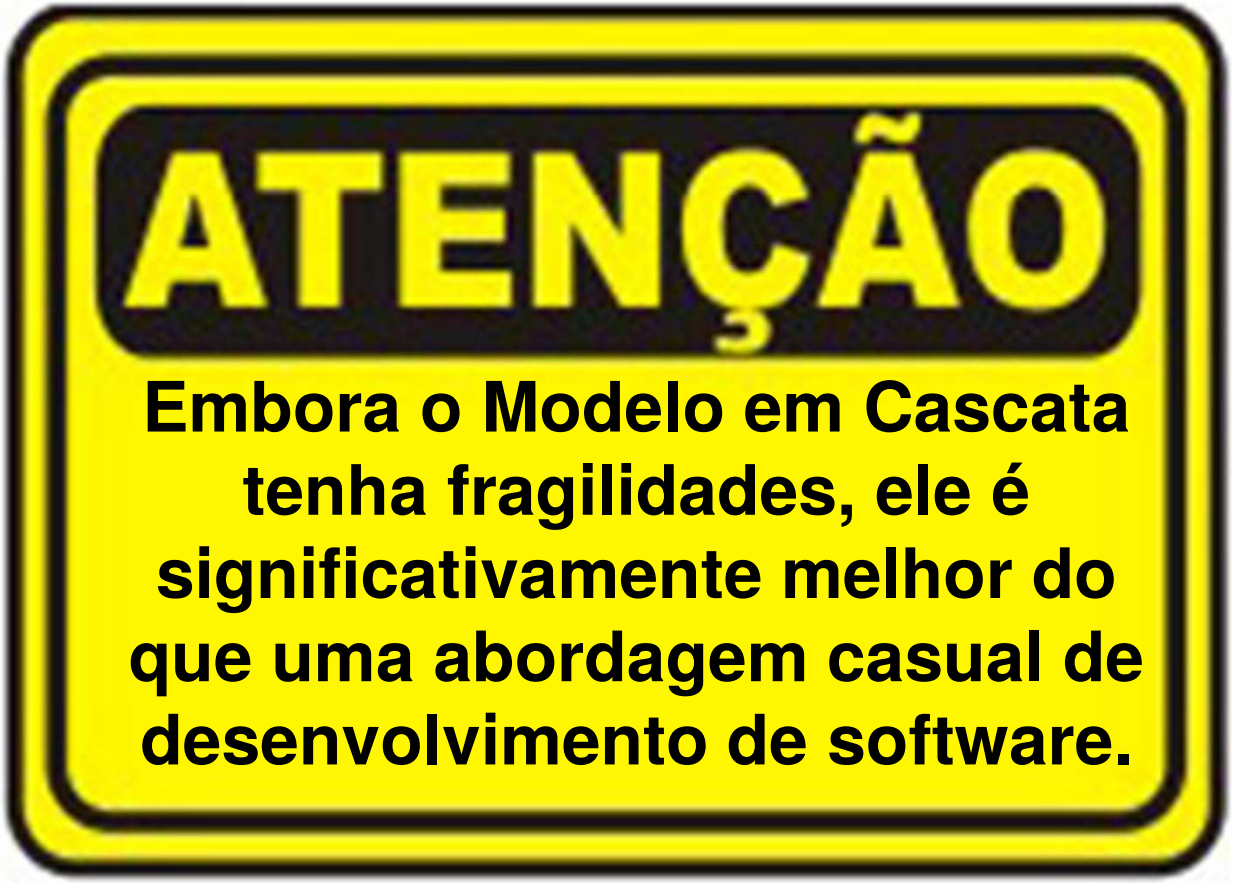


Manutenção

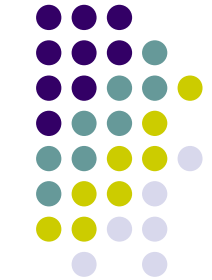
- provavelmente o software deverá sofrer mudanças depois que for entregue ao cliente
- causas das mudanças: e*rros*, *adaptação do software para acomodar mudanças em seu ambiente externo* e *exigência do cliente para acréscimos funcionais e de desempenho*

Problemas com o Modelo em Cascata

- Projetos reais raramente seguem o fluxo seqüencial que o modelo propõe;
- Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural;
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento (na instalação);
- Difícil identificação de sistemas legados (não acomoda a engenharia reversa).


ATENÇÃO
Embora o Modelo em Cascata tenha fragilidades, ele é significativamente melhor do que uma abordagem casual de desenvolvimento de software.

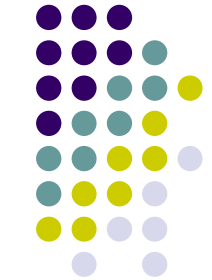
O Modelo em Cascata



- O Modelo de processo em Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:
  - Imposição de **disciplina**, **planejamento** e **gerenciamento**
  - a implementação do produto deve ser **postergada** até que os objetivos tenham sido completamente entendidos;
  - Permite gerência do ***baseline***, que identifica um conjunto ***fixo*** de documentos produzidos ao longo do processo de desenvolvimento;

O Paradigma de Prototipação

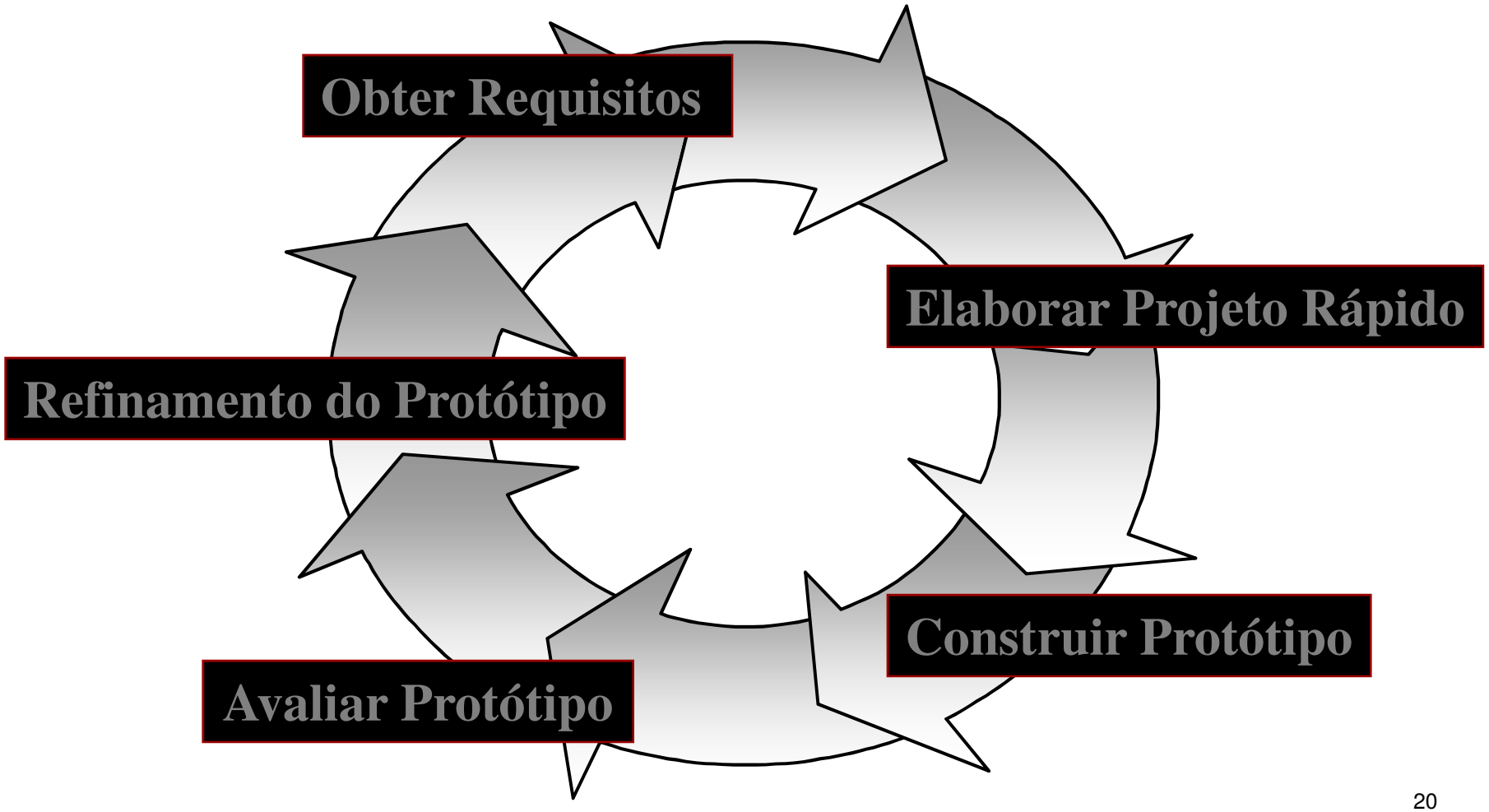
O Modelo de Prototipação



- o objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.
- possibilita que o desenvolvedor crie um modelo (protótipo)do software que deve ser construído
- apropriado para quando o cliente não definiu detalhadamente os requisitos.

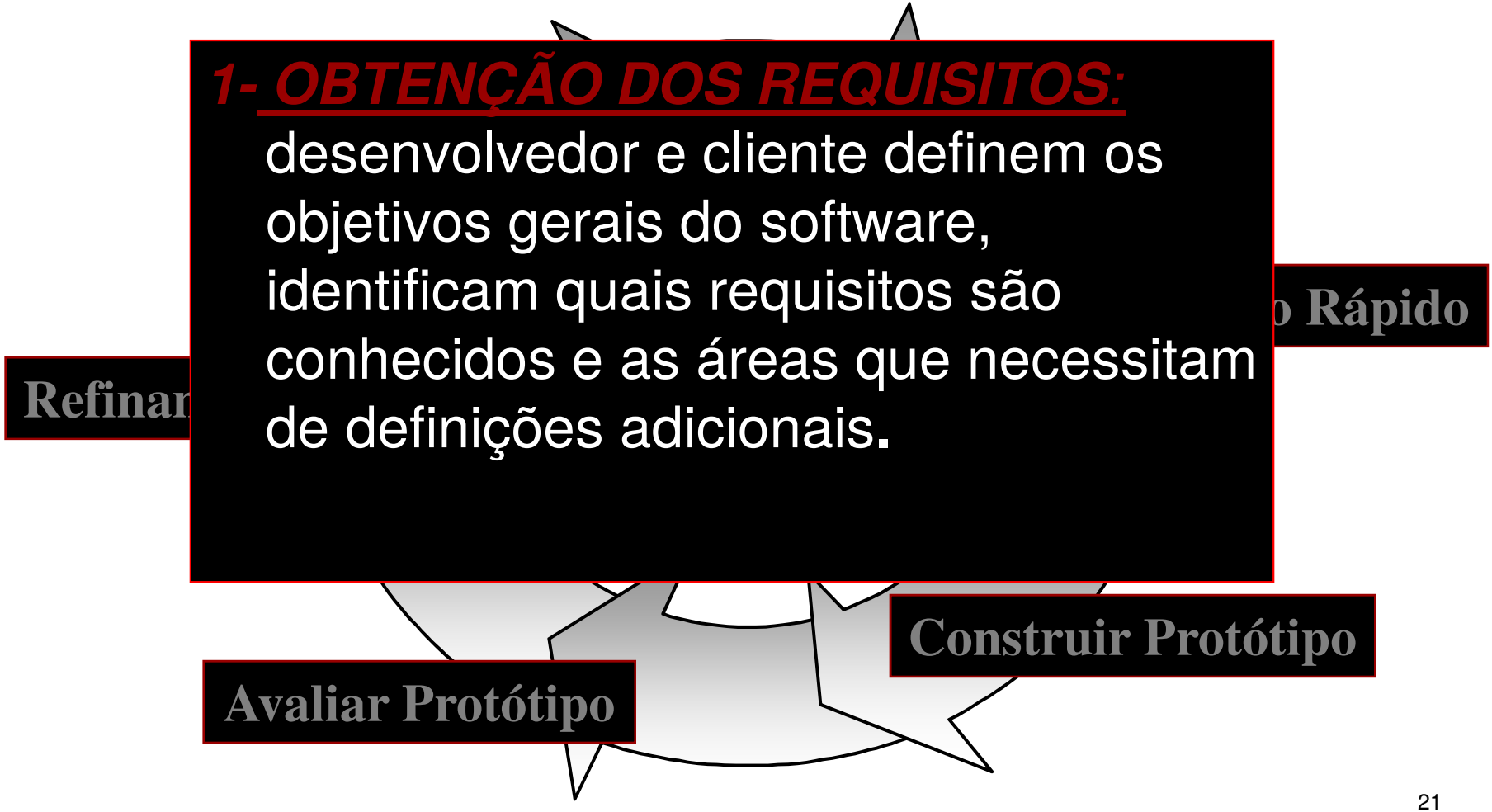
O Paradigma de Prototipação

para obtenção dos requisitos



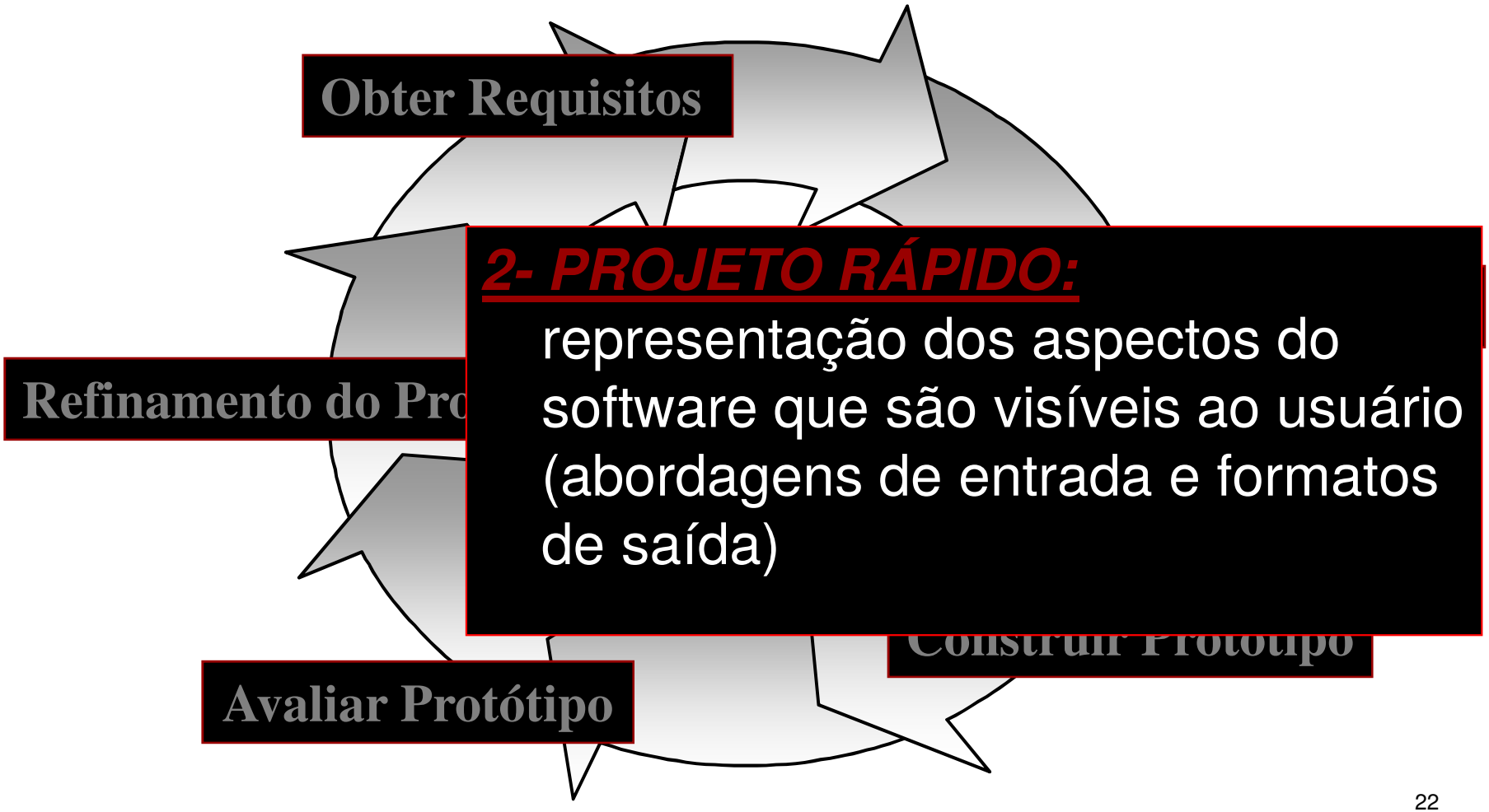
O Paradigma de Prototipação

para obtenção dos requisitos



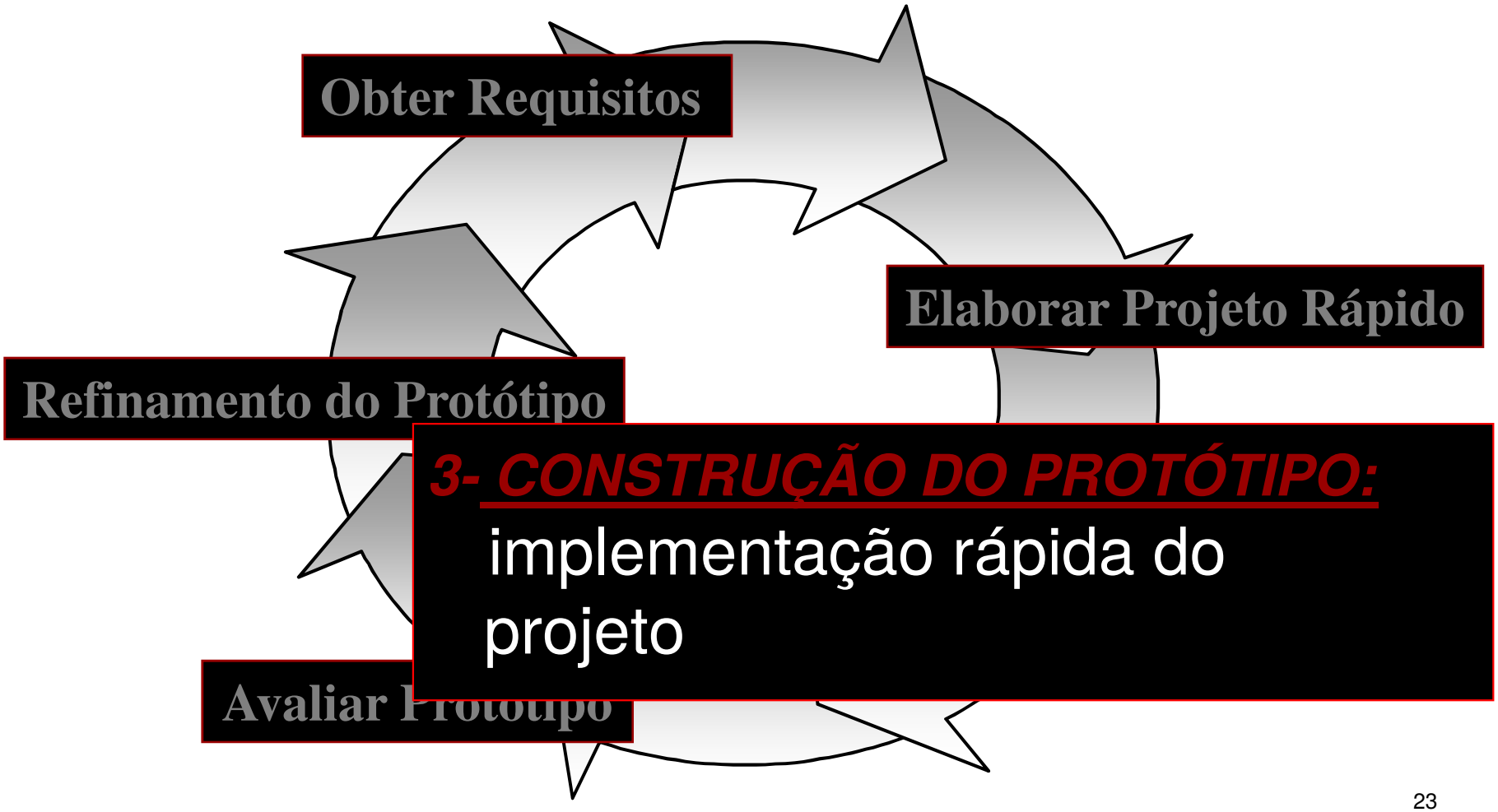
O Paradigma de Prototipação

para obtenção dos requisitos



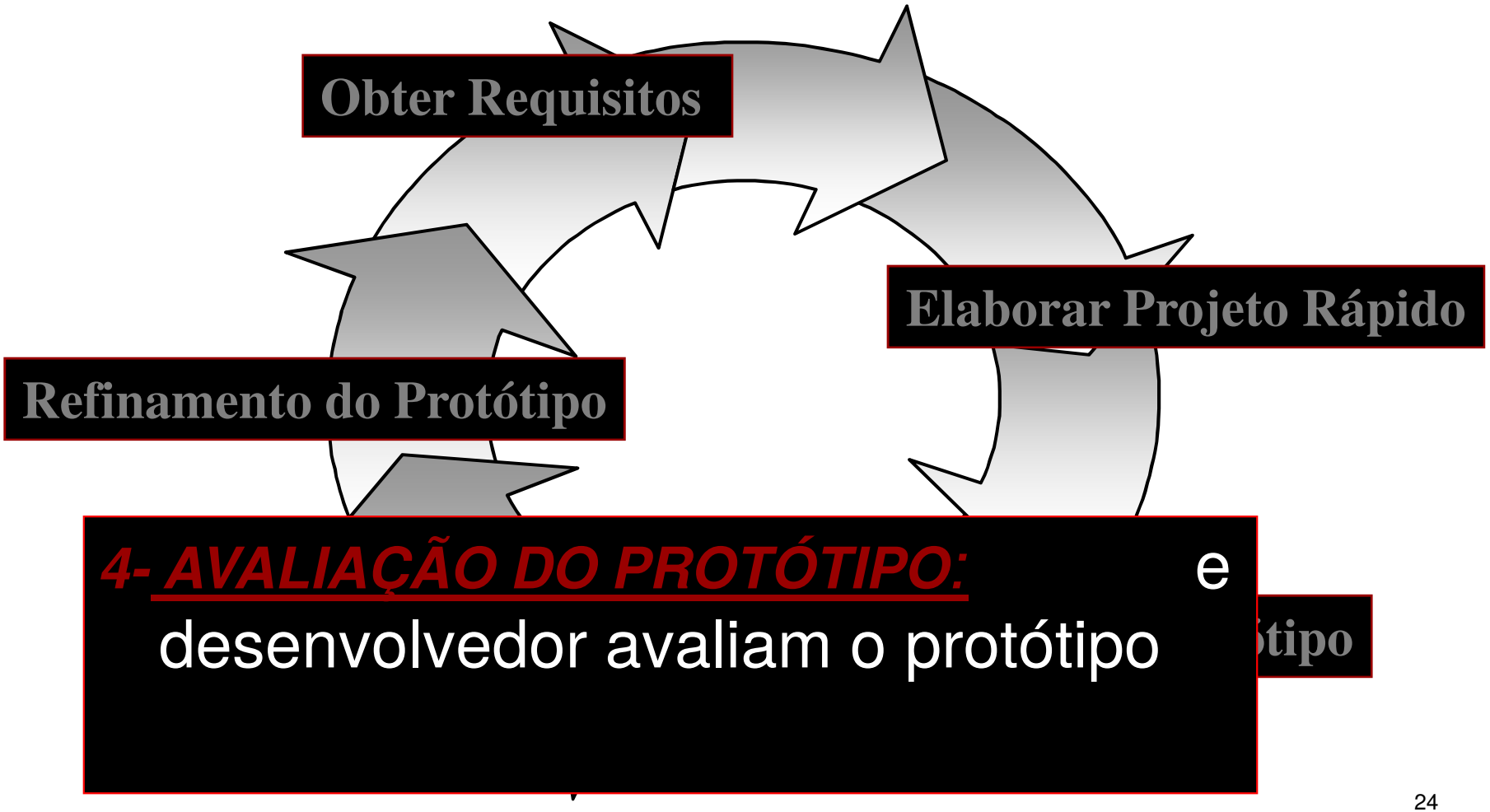
O Paradigma de Prototipação

para obtenção dos requisitos



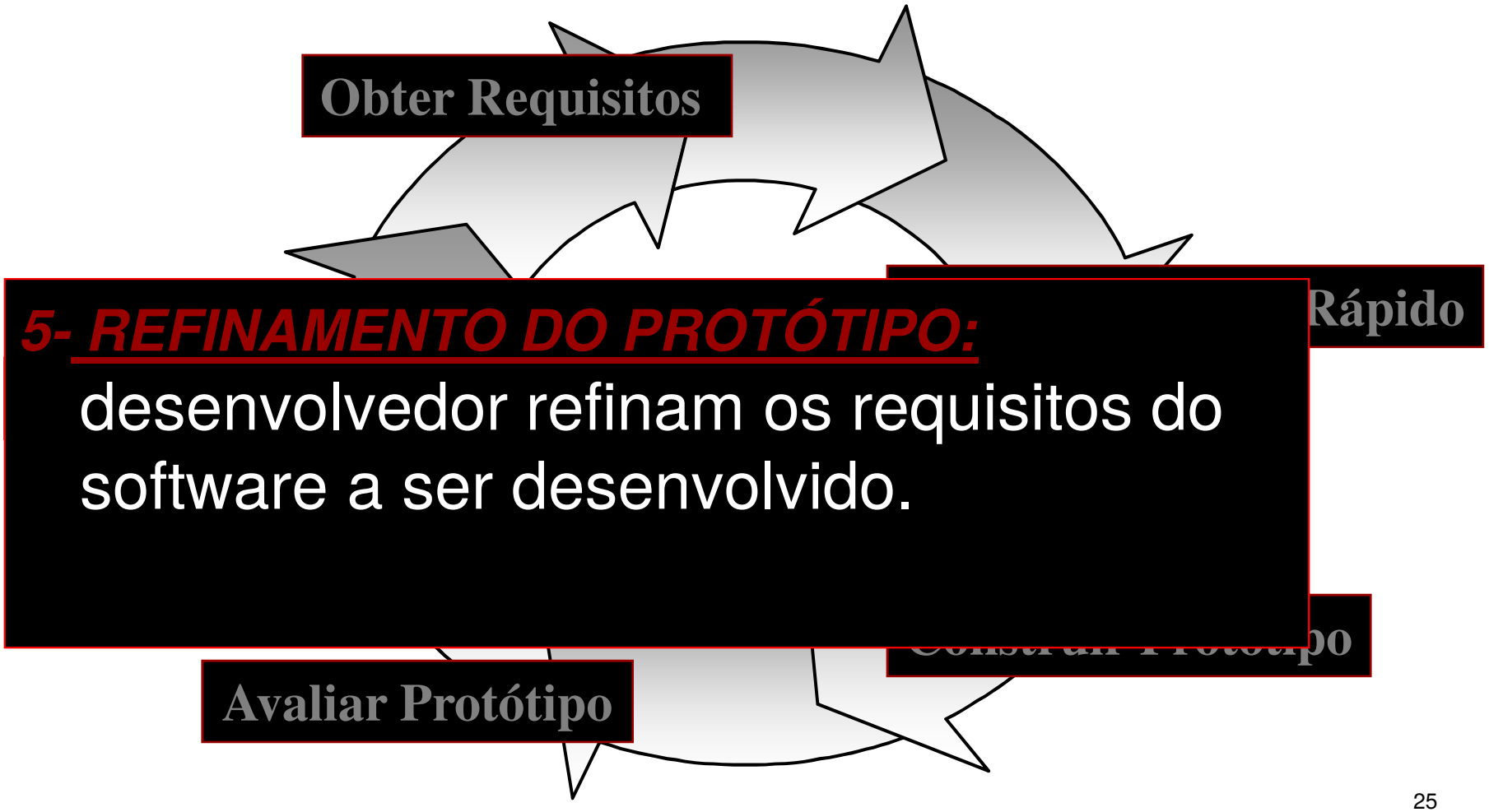
O Paradigma de Prototipação

para obtenção dos requisitos



O Paradigma de Prototipação

para obtenção dos requisitos

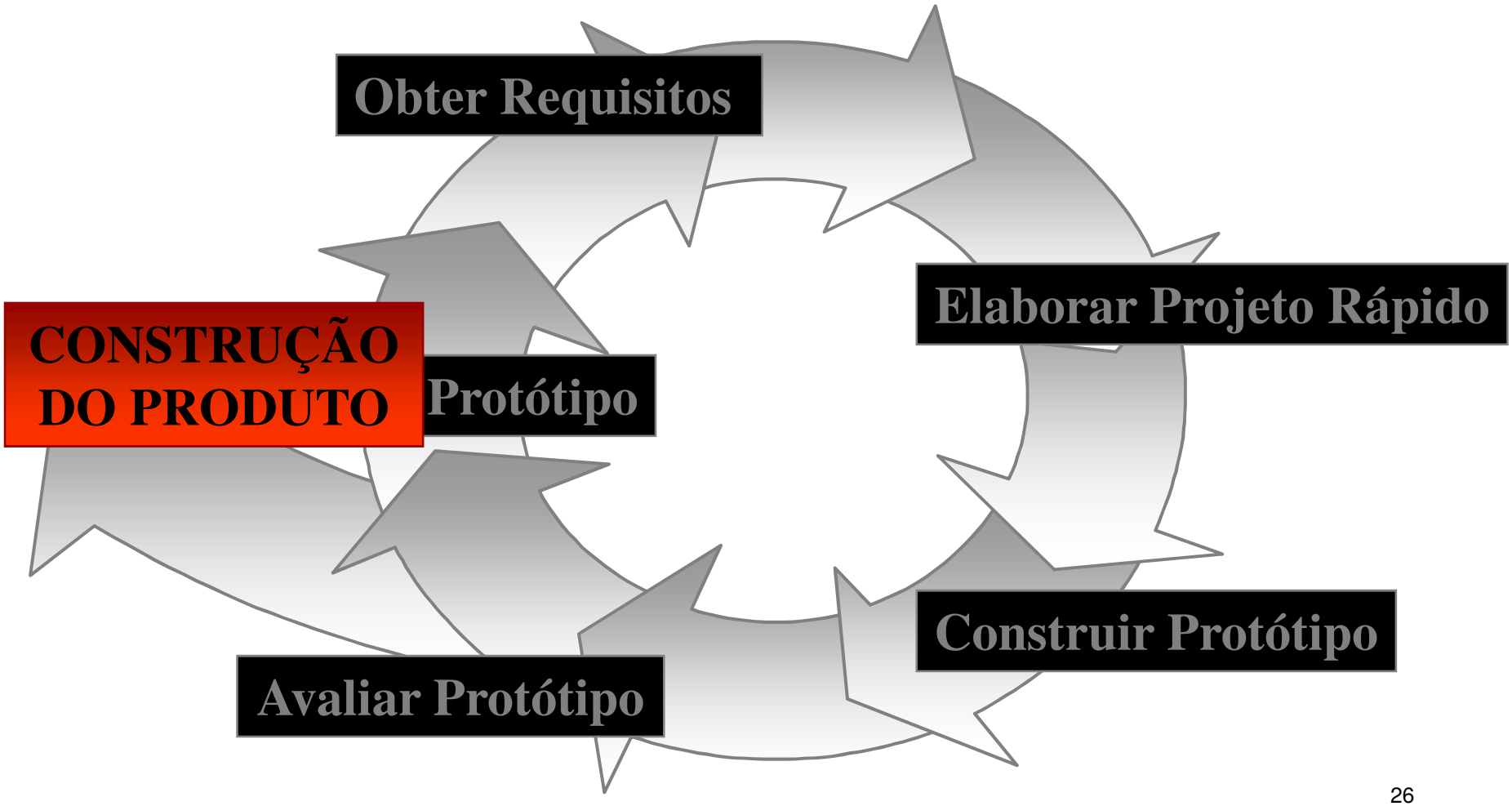


5- REFINAMENTO DO PROTÓTIPO:

desenvolvedor refinam os requisitos do software a ser desenvolvido.

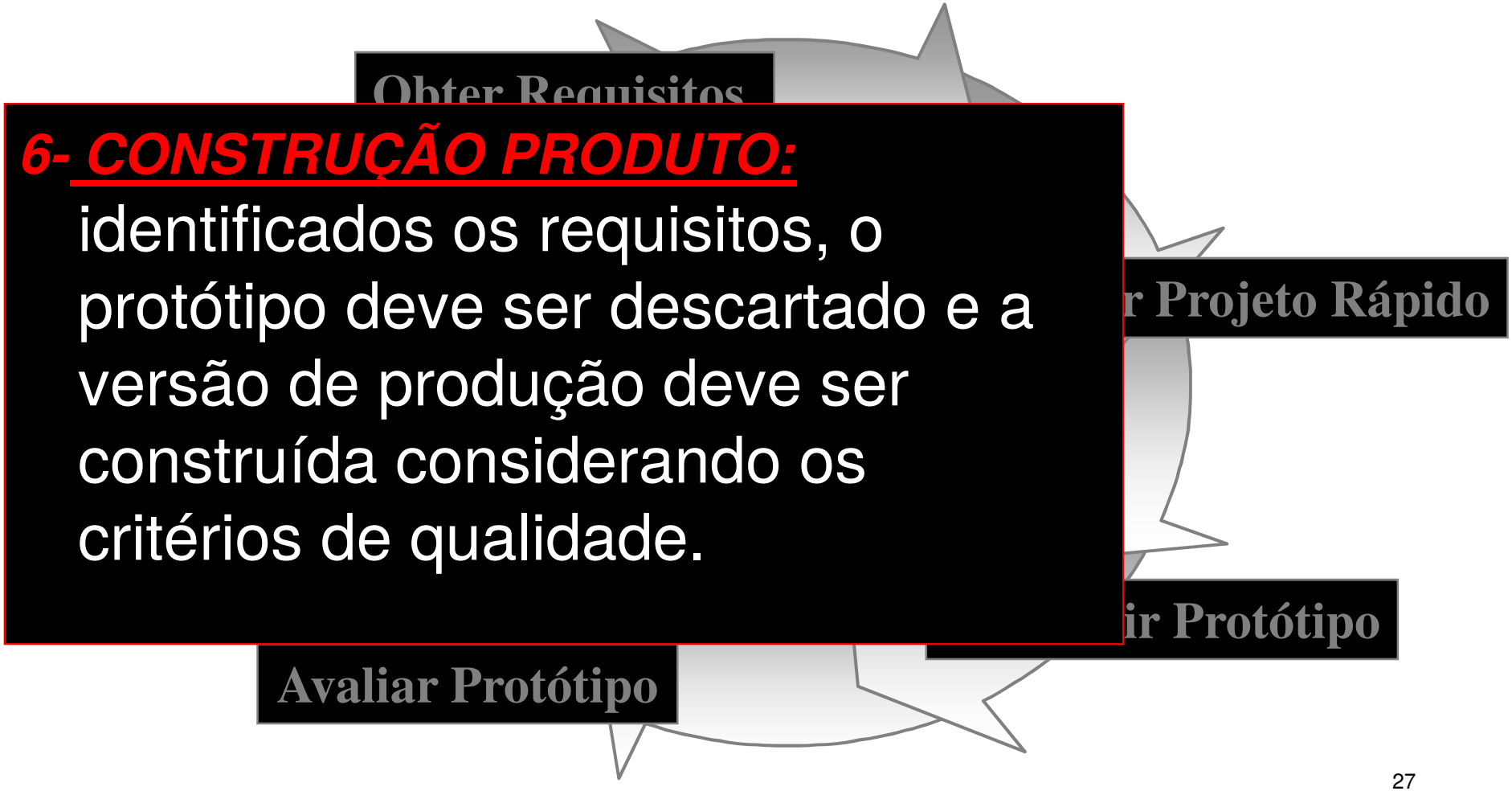
O Paradigma de Prototipação

para obtenção dos requisitos



O Paradigma de Prototipação

para obtenção dos requisitos



6- CONSTRUÇÃO PRODUTO:

identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.

Problemas com a Prototipação

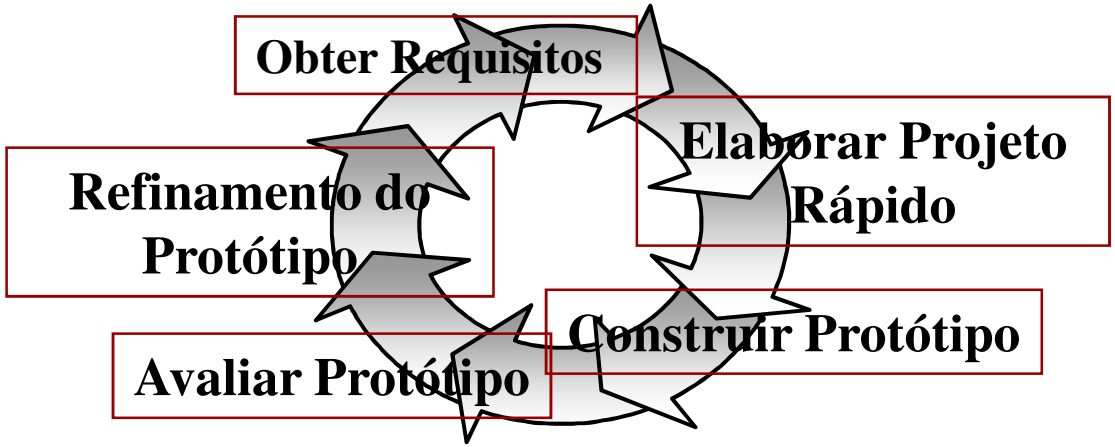
- cliente não sabe que o software que ele vê não considerou, durante o desenvolvimento, a qualidade global e a manutenibilidade a longo prazo
- desenvolvedor freqüentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo

Comentários sobre o Paradigma de Prototipação

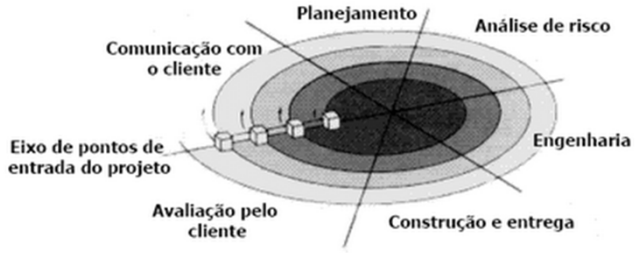
- ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- a chave é definir as regras do jogo logo no começo.
- o cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo para definir os requisitos

Exercício

- Vamos refazer o exercício da primeira aula, agora utilizando a Prototipação para construir o Projeto.



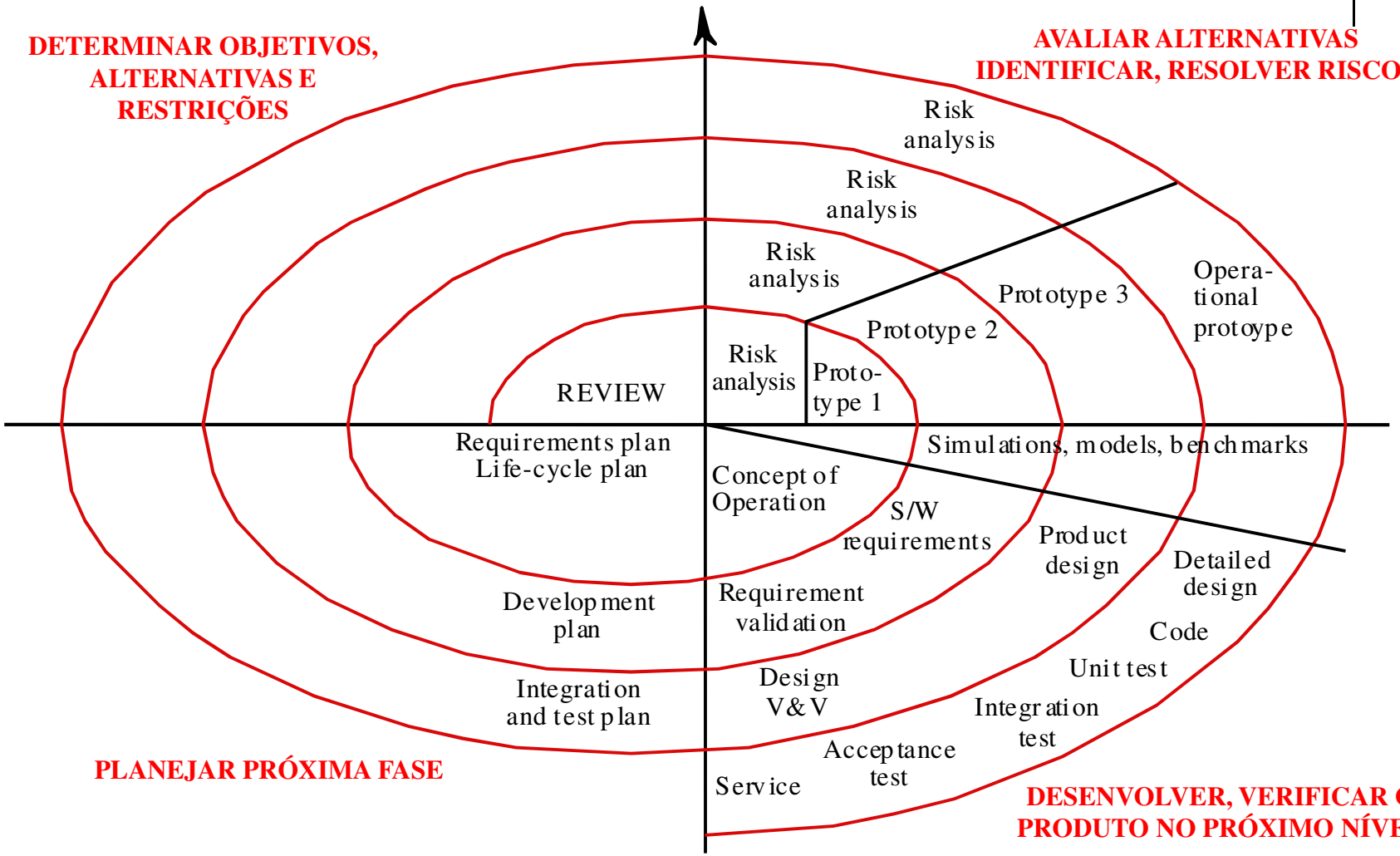
O Modelo Espiral



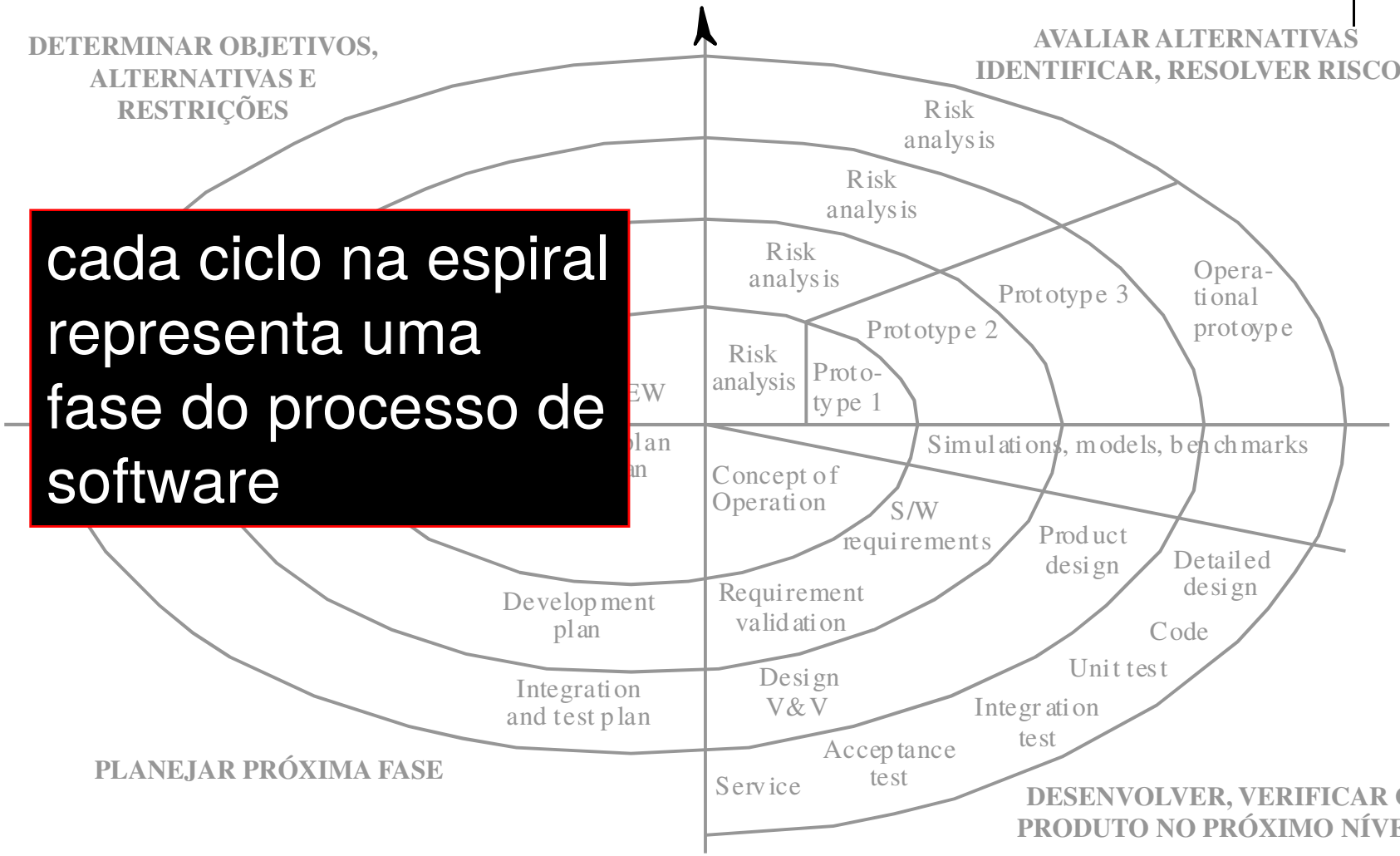
O Modelo Espiral (Boehm, 1986)

- O modelo espiral acopla a natureza iterativa da prototipação com os aspectos controlados e sistemáticos do modelo cascata.
- O modelo espiral é dividido em uma série de *atividades de trabalho* ou *regiões de tarefa*.
- Existem tipicamente de 3 a 6 regiões de tarefa.
- Combina as características positivas da gerência ***baseline*** (documentos associados ao processo);

O Modelo Espiral (com 4 regiões)

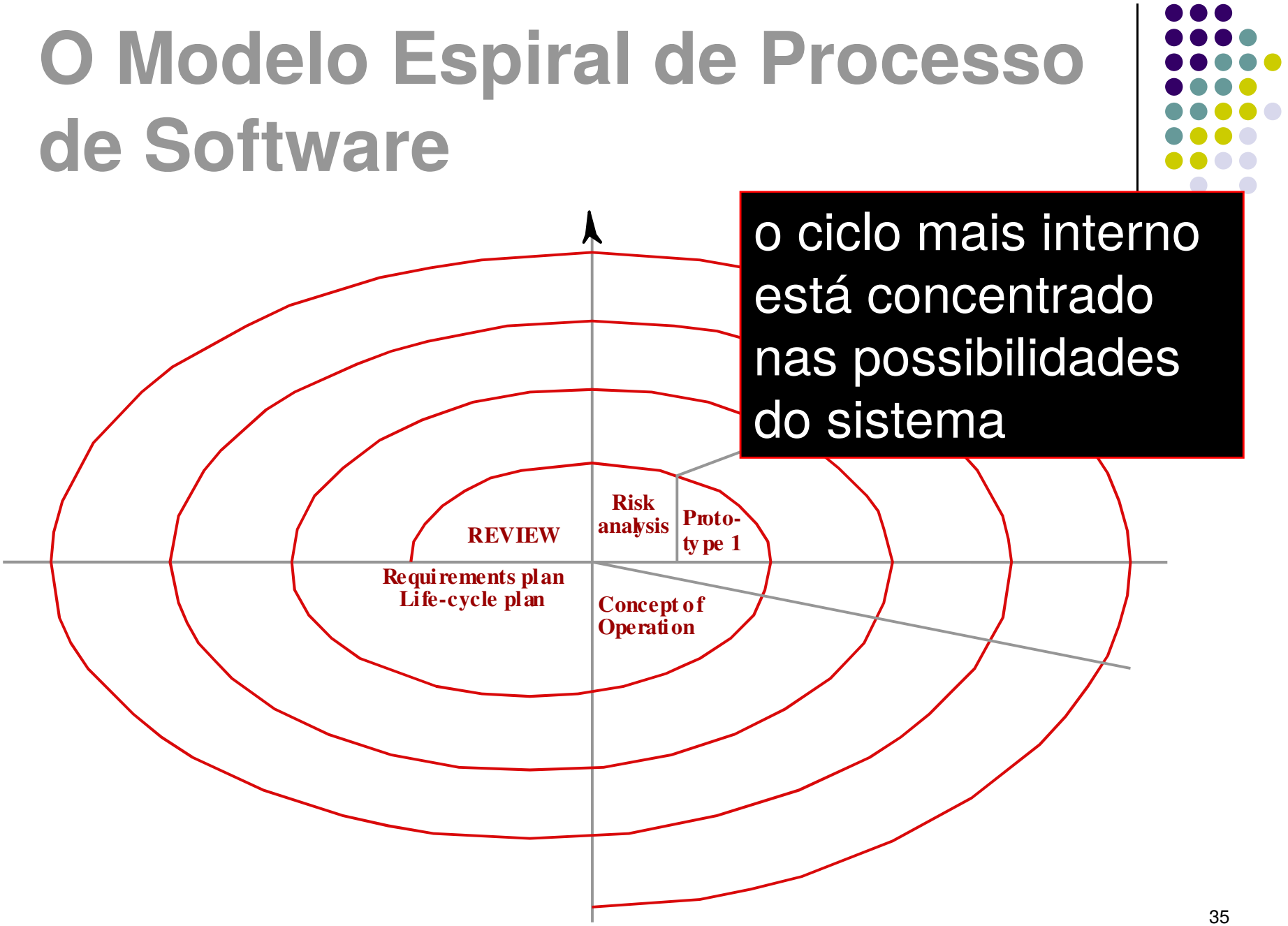


O Modelo Espiral (com 4 regiões)

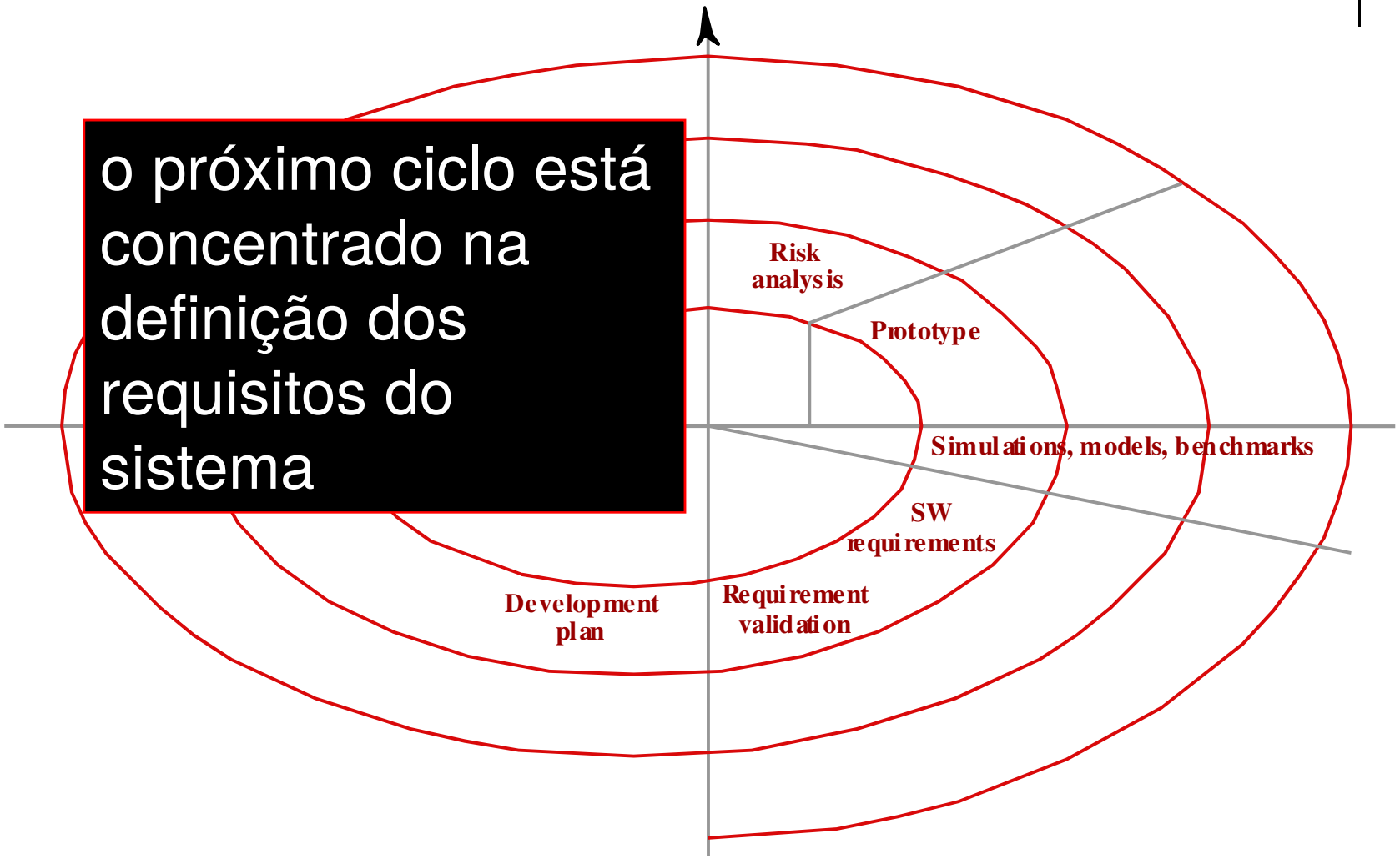


cada ciclo na espiral representa uma fase do processo de software

O Modelo Espiral de Processo de Software

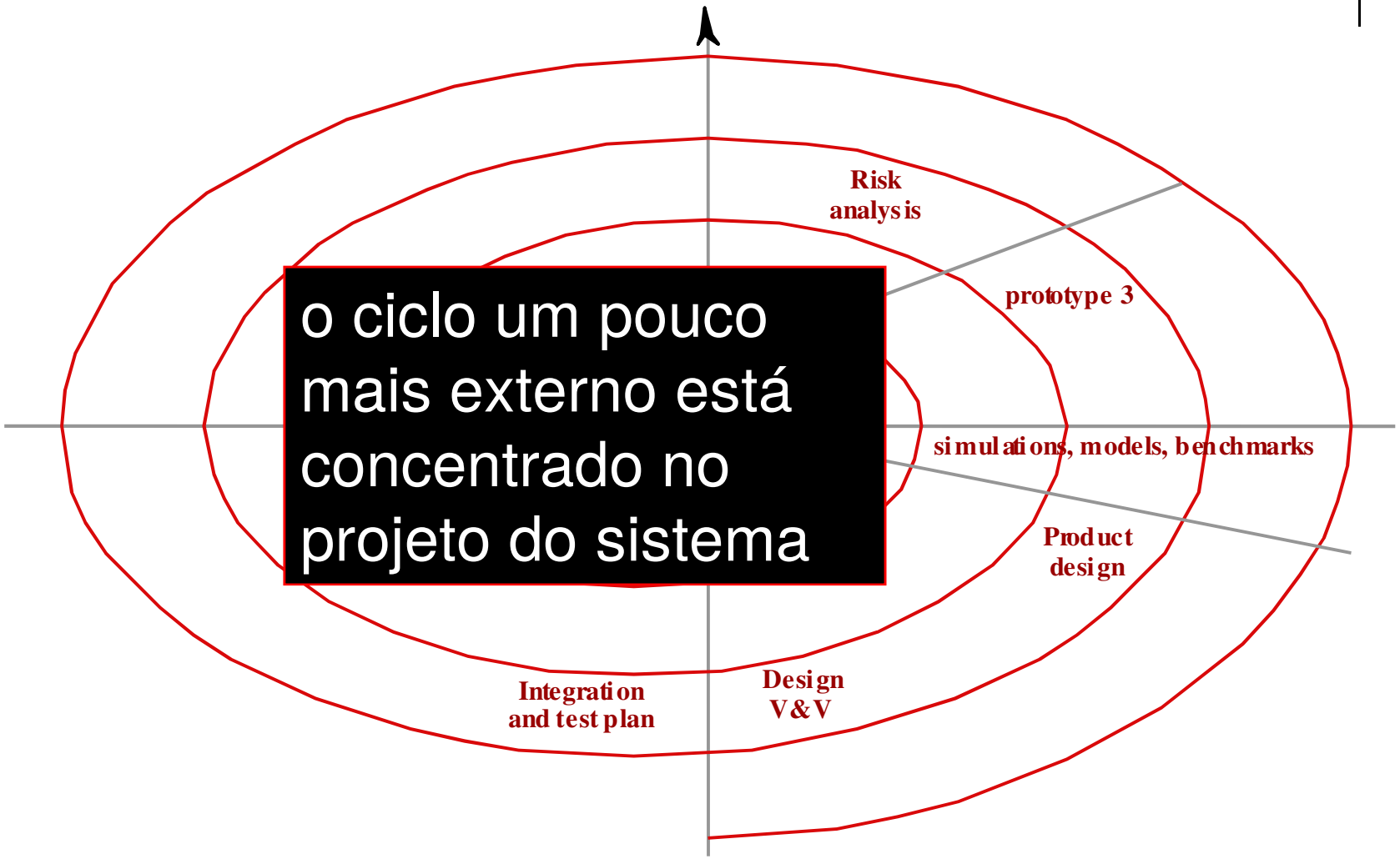


O Modelo Espiral de Processo de Software



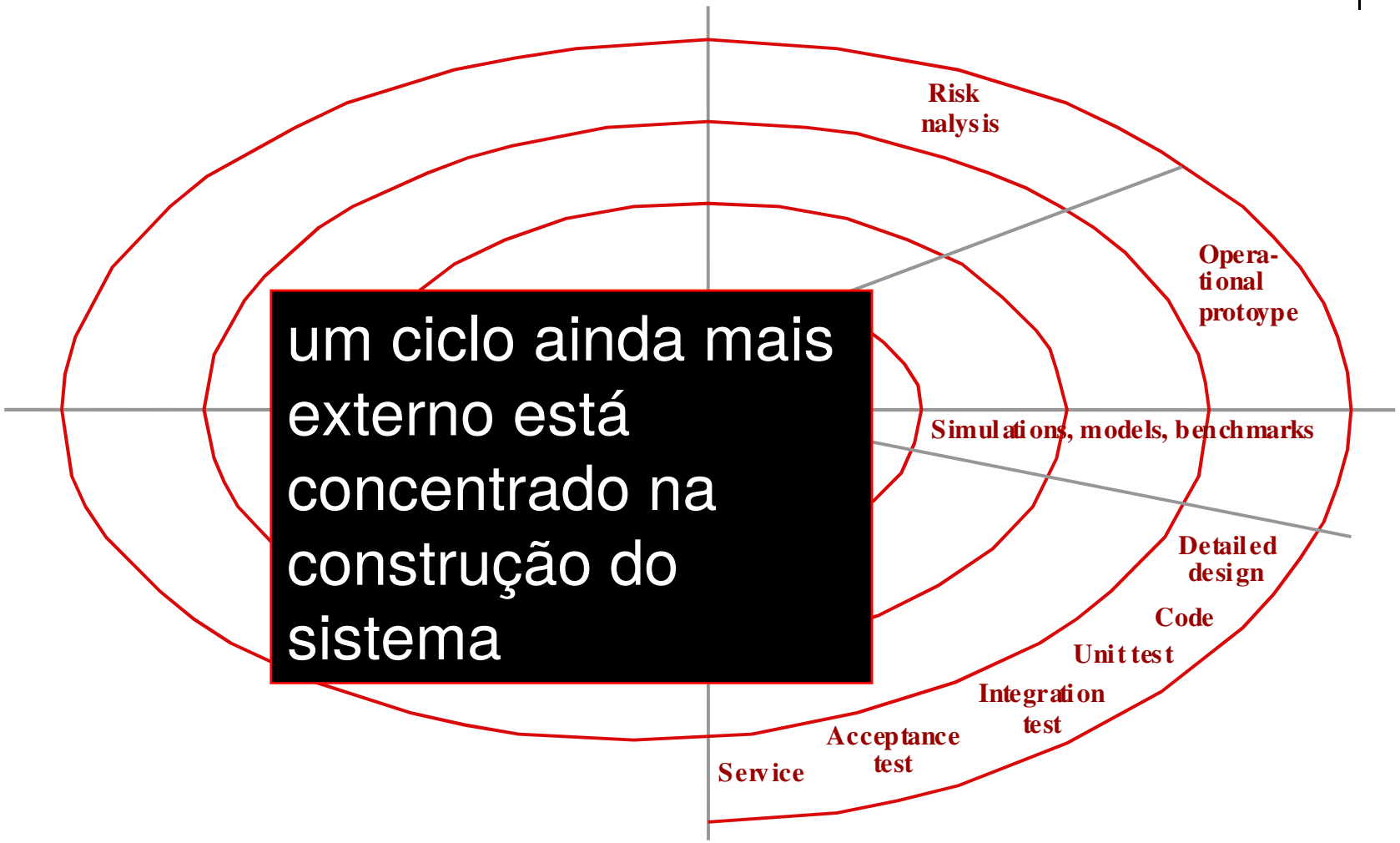
o próximo ciclo está concentrado na definição dos requisitos do sistema

O Modelo Espiral de Processo de Software



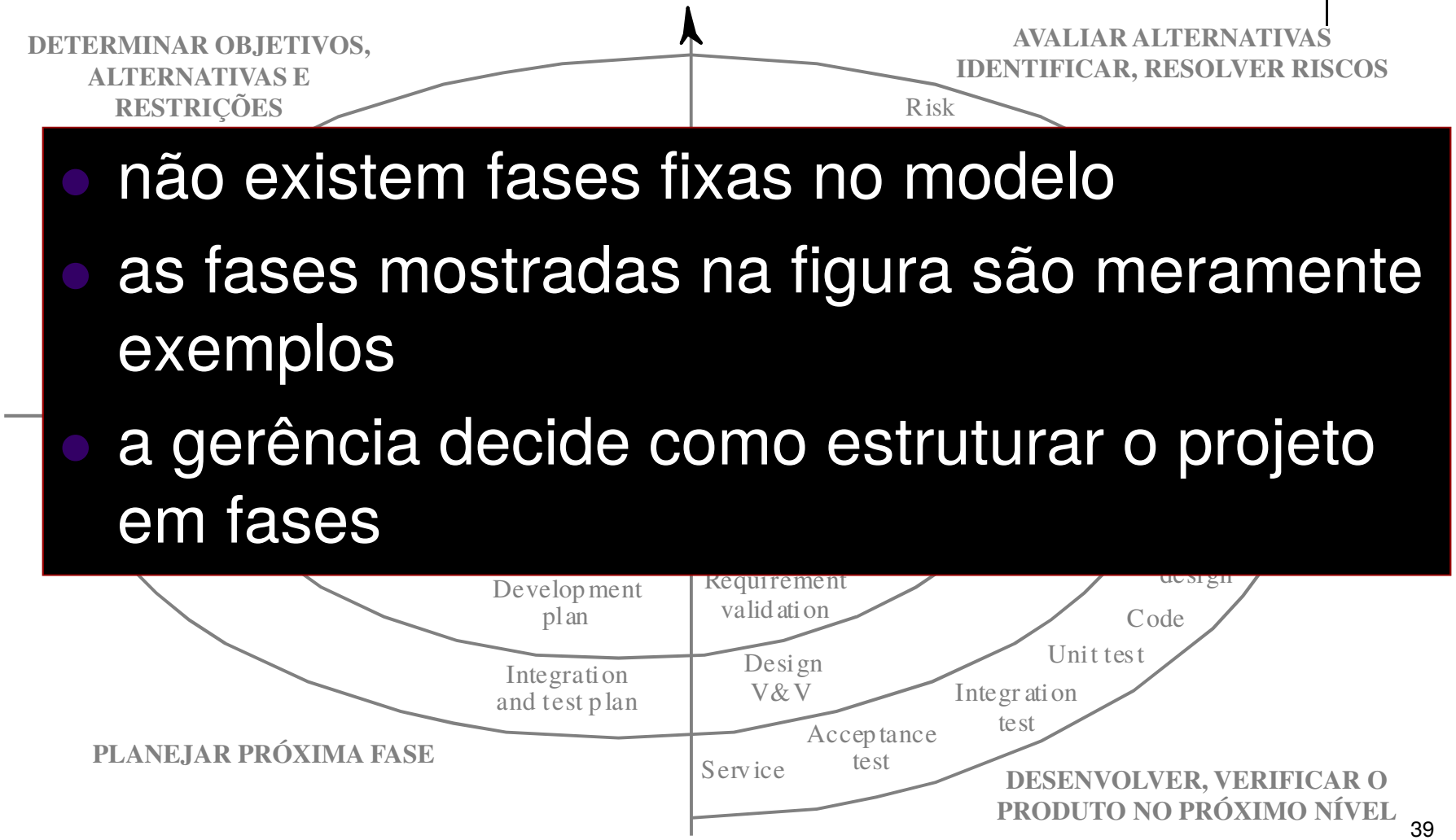
o ciclo um pouco mais externo está concentrado no projeto do sistema

O Modelo Espiral de Processo de Software



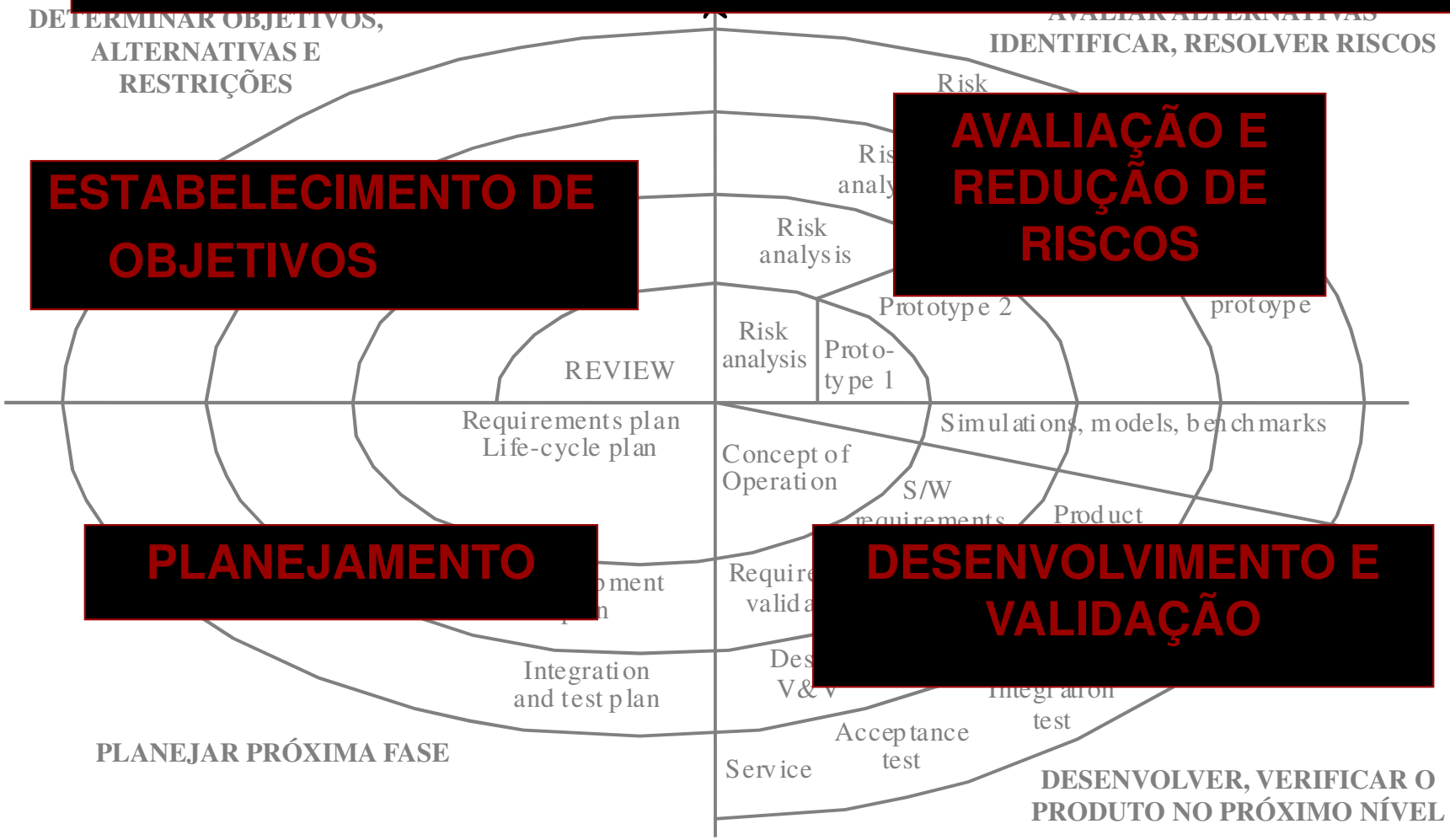
um ciclo ainda mais externo está concentrado na construção do sistema

O Modelo Espiral (com 4 regiões)

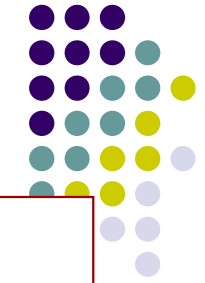


- não existem fases fixas no modelo
- as fases mostradas na figura são meramente exemplos
- a gerência decide como estruturar o projeto em fases

- Cada “loop” do espiral é dividido em 4 setores



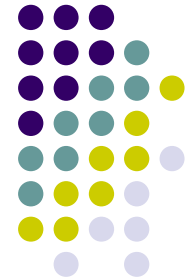
O Modelo Espiral de Processo de Software

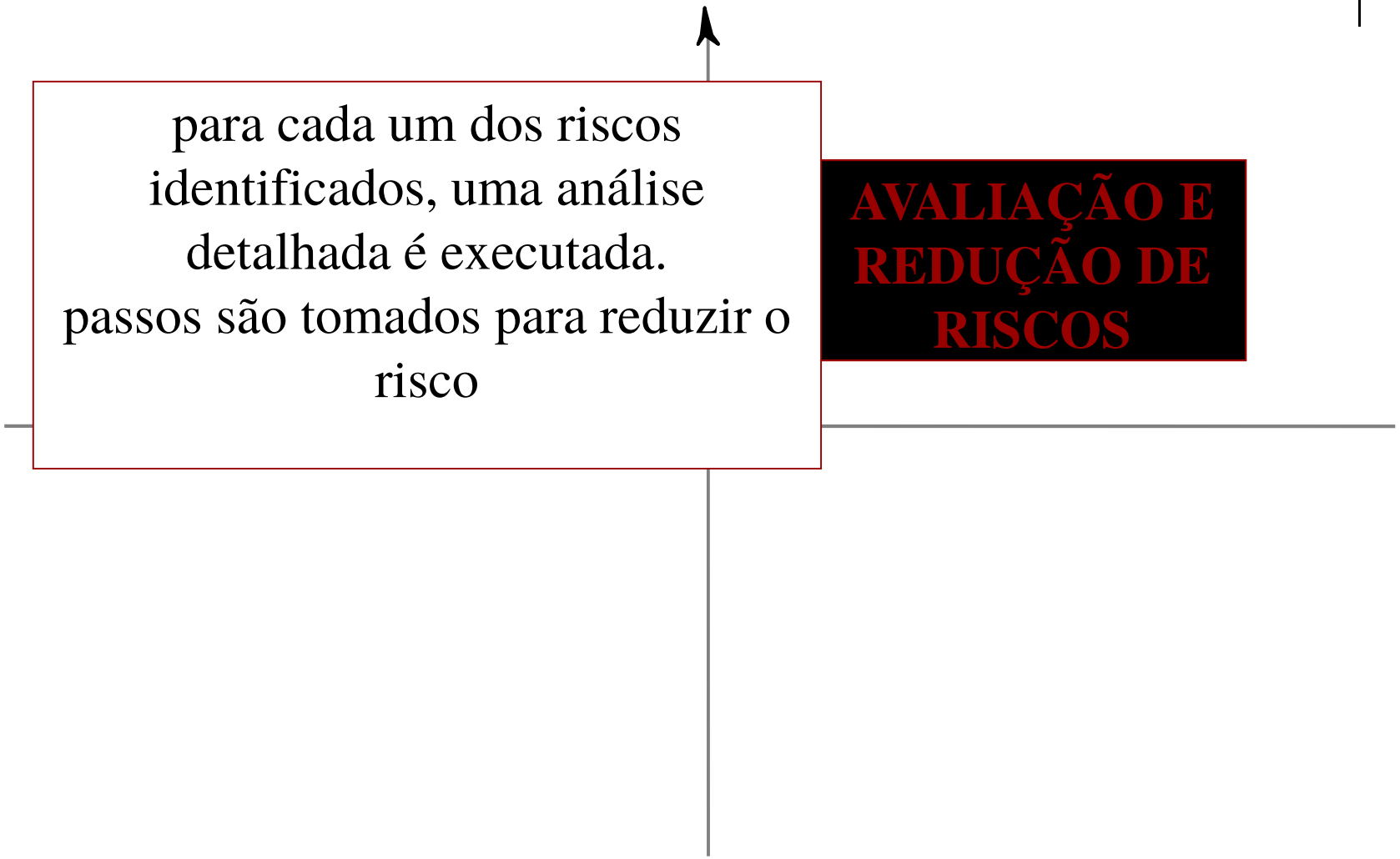


ESTABELECIMENTO DE OBJETIVOS

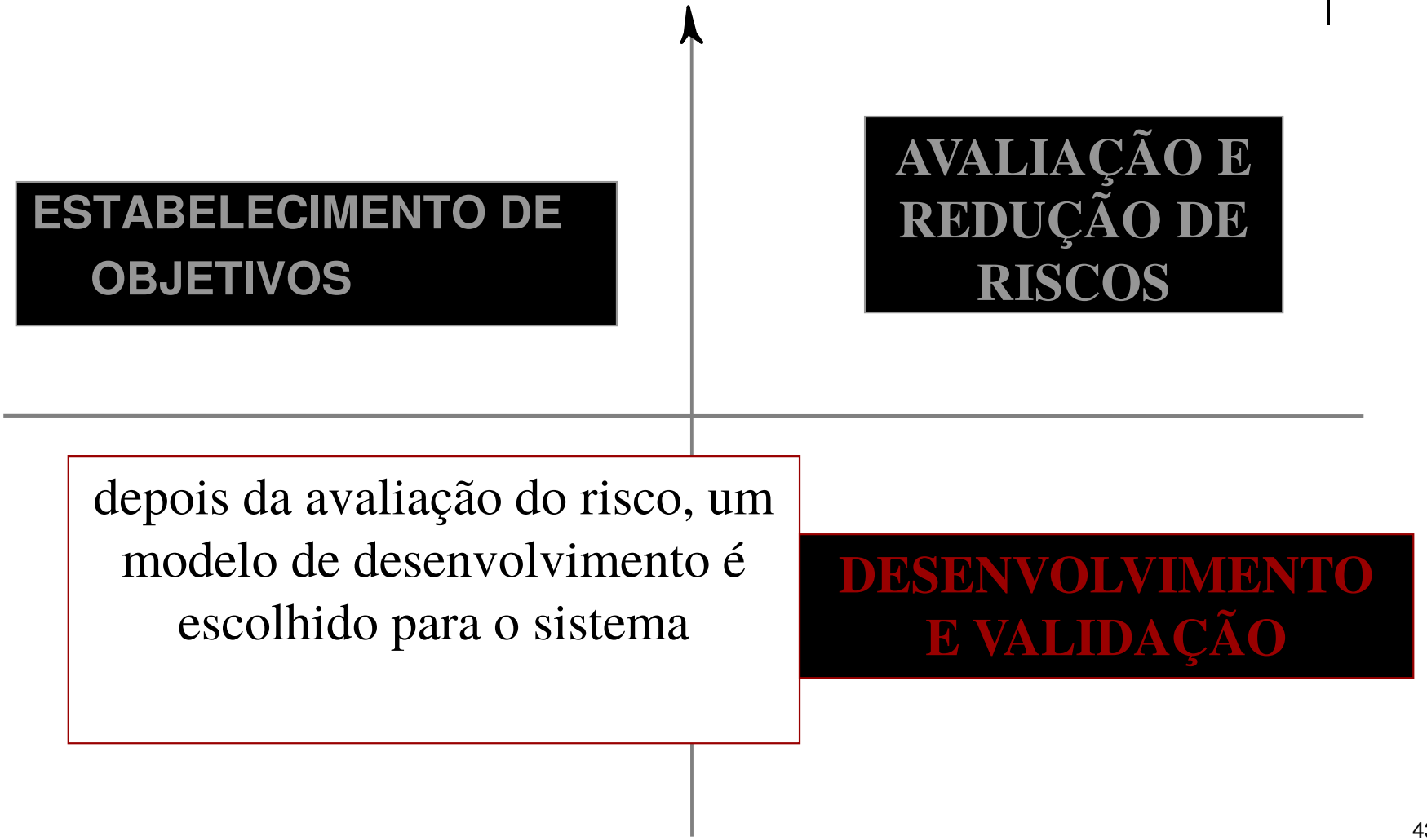
são definidos objetivos específicos para a fase do projeto
são identificadas restrições sobre o processo e o produto
é projetado um plano de gerenciamento detalhado
são identificados riscos do projeto
dependendo dos riscos, estratégias alternativas podem ser planejadas

O Modelo Espiral de Processo de Software

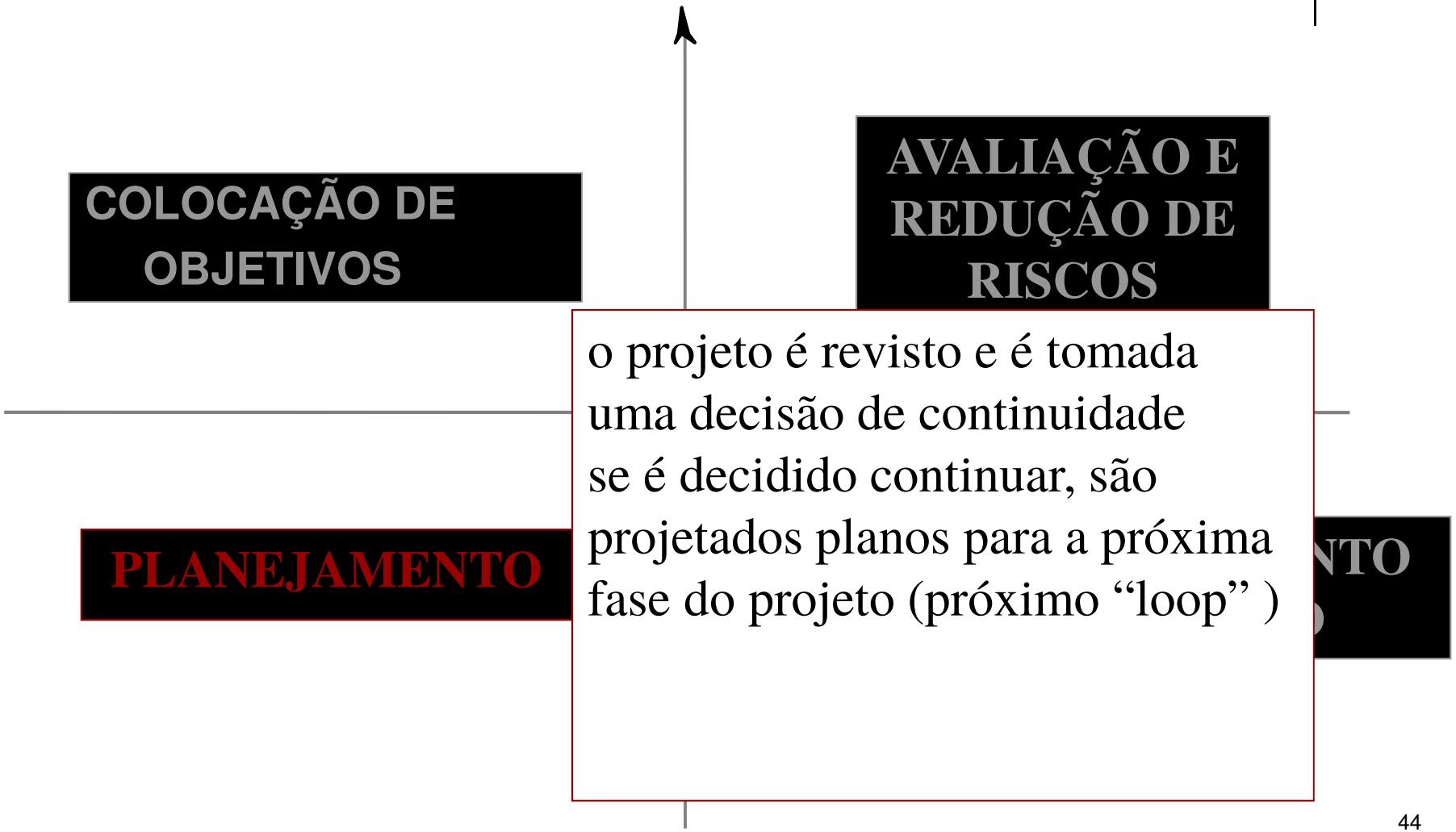




O Modelo Espiral de Processo de Software



O Modelo Espiral de Processo de Software



O Modelo Espiral

- engloba as melhores características do ciclo de vida Clássico e da Prototipação, adicionando um novo elemento: a *Análise de Risco*
- segue a abordagem de passos sistemáticos do Ciclo de Vida Clássico incorporando-os numa estrutura iterativa que reflete mais realisticamente o mundo real
- usa a Prototipação em todas as etapas da evolução do produto, como mecanismo de redução de riscos

Comentários sobre o Ciclo de Vida em Espiral

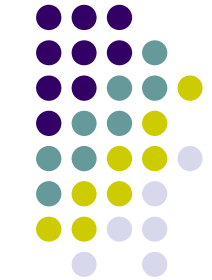
- usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva

- pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável

Comentários sobre o Ciclo de Vida em Espiral

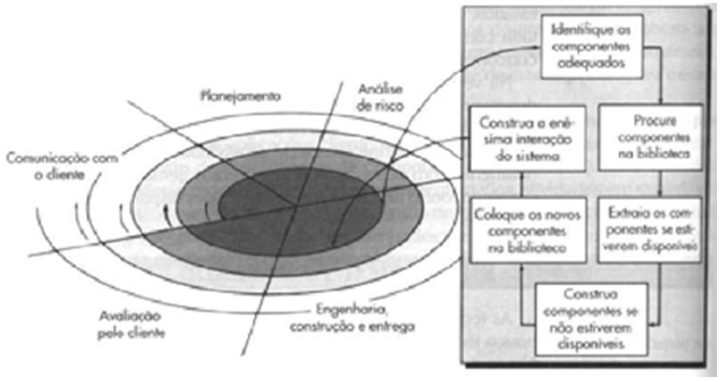
- exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso
- o modelo é relativamente novo e não tem sido amplamente usado. Demorará muitos anos até que a eficácia desse modelo possa ser determinada com certeza absoluta

O Modelo Espiral

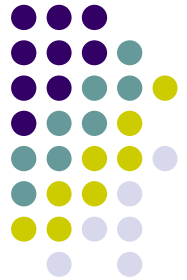


- adiciona um novo elemento: a *Análise de Risco*
- usa a Prototipação, em qualquer etapa da evolução do produto, como mecanismo de redução de riscos
- exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso

O MODELO BASEADO EM COMPONENTES

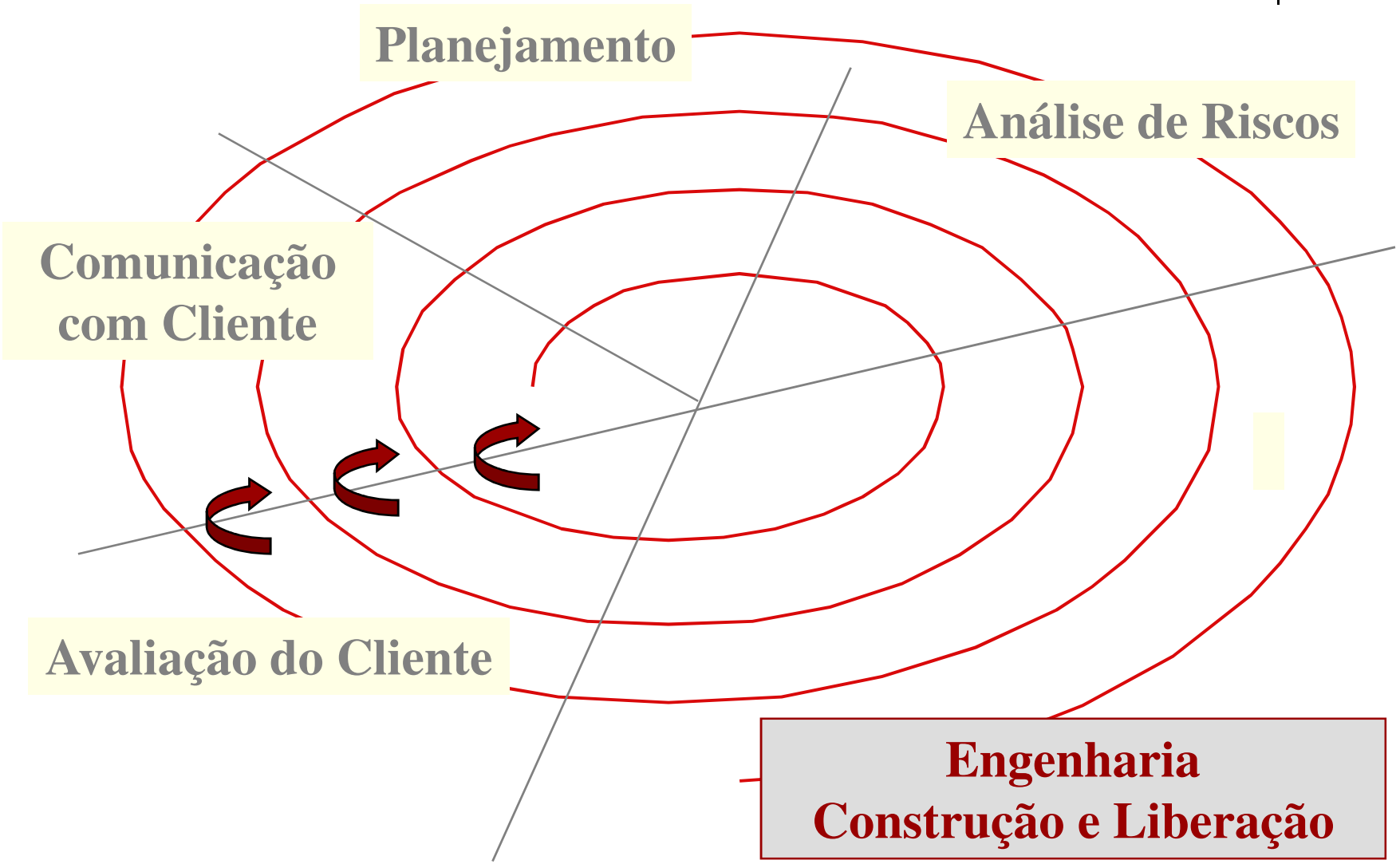


O Modelo Baseado em Componentes

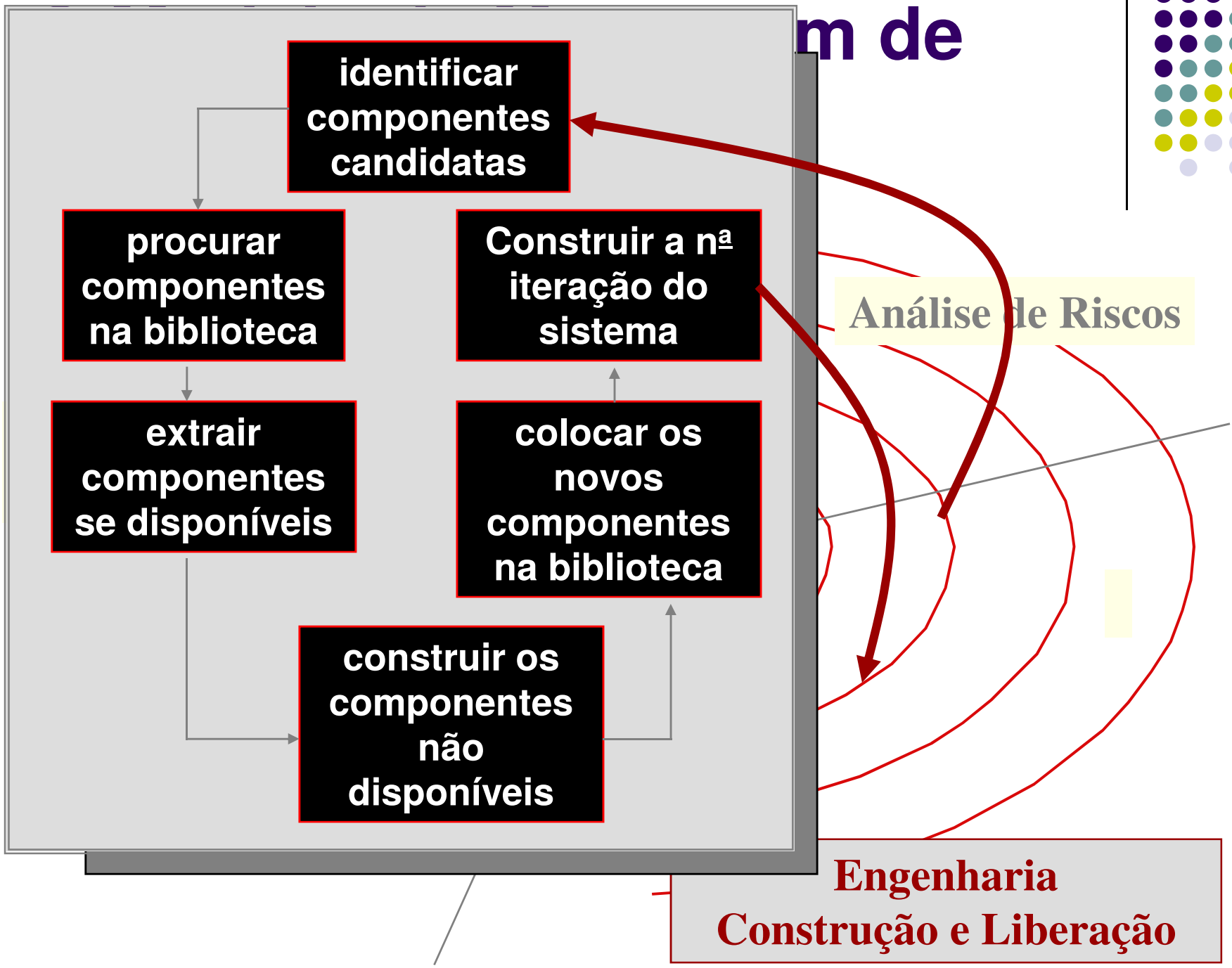


- Utiliza tecnologias orientadas a objeto
- Quando projetadas e implementadas apropriadamente as classes orientadas a objeto são reutilizáveis em diferentes aplicações e arquiteturas de sistema
- O modelo de montagem de componentes incorpora muitas das características do modelo espiral.

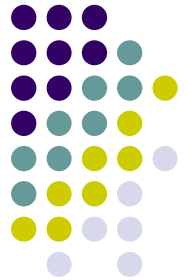
O Modelo de Montagem de Componentes



m de


identificar componentes candidatas
procurar componentes na biblioteca
Construir a nª iteração do sistema
extrair componentes se disponíveis
colocar os novos componentes na biblioteca
construir os componentes não disponíveis
Análise de Riscos
Engenharia
Construção e Liberação

O Modelo Baseado em Componentes



- O modelo baseado em componentes conduz ao reuso do software
- a reusabilidade fornece uma série de benefícios:
  - redução de até 70% no tempo de desenvolvimento
  - redução de até 84% no custo do projeto
  - índice de produtividade de até 26.2 (normal da indústria é de 16.9)
- esses resultados dependem da robustez da biblioteca de componentes