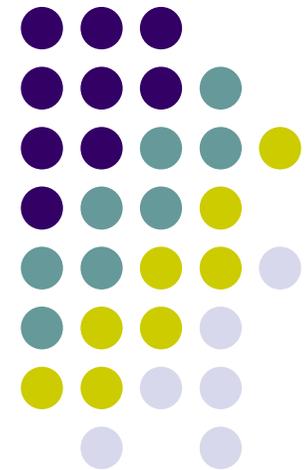


Modelos de Processo de Software

Ricardo Argenton Ramos 
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

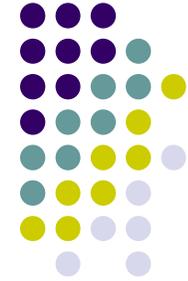
pode ser vista como uma abordagem de desenvolvimento de software elaborada com disciplina e métodos bem definidos.

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

Modelos de Processo de Software

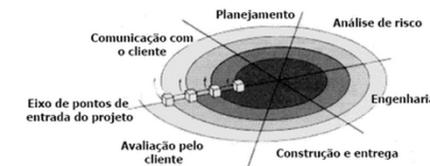
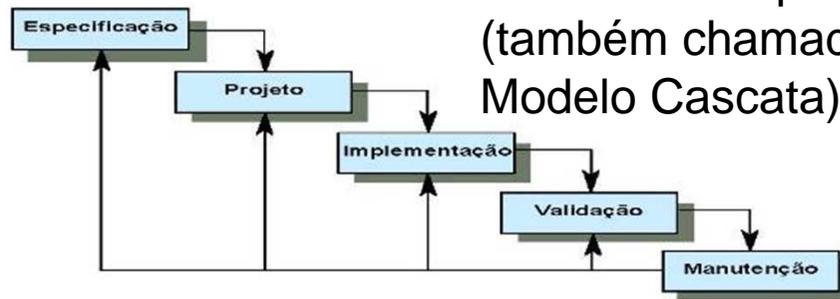


- Existem vários *modelos de processo de software* (ou *paradigmas de engenharia de software*)
- Cada um representa uma tentativa de colocar ordem em uma atividade inerentemente caótica

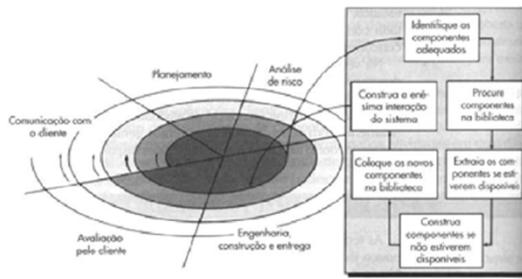


Modelos de Processo de Software

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

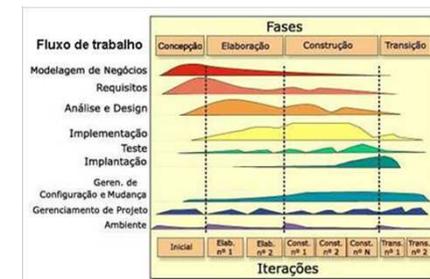


O Modelo Espiral



O Modelo Baseado em Componentes

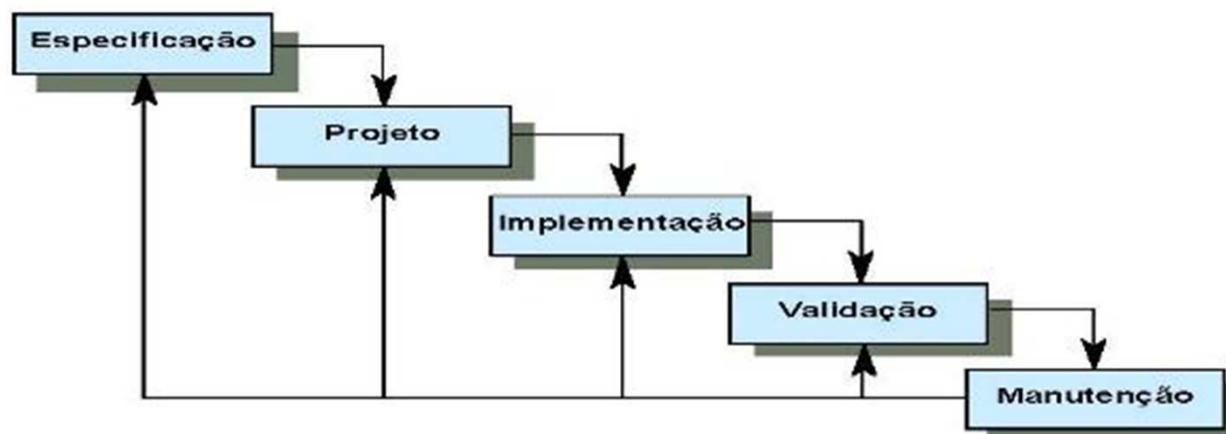
O Paradigma de Prototipação



Processo Unificado



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

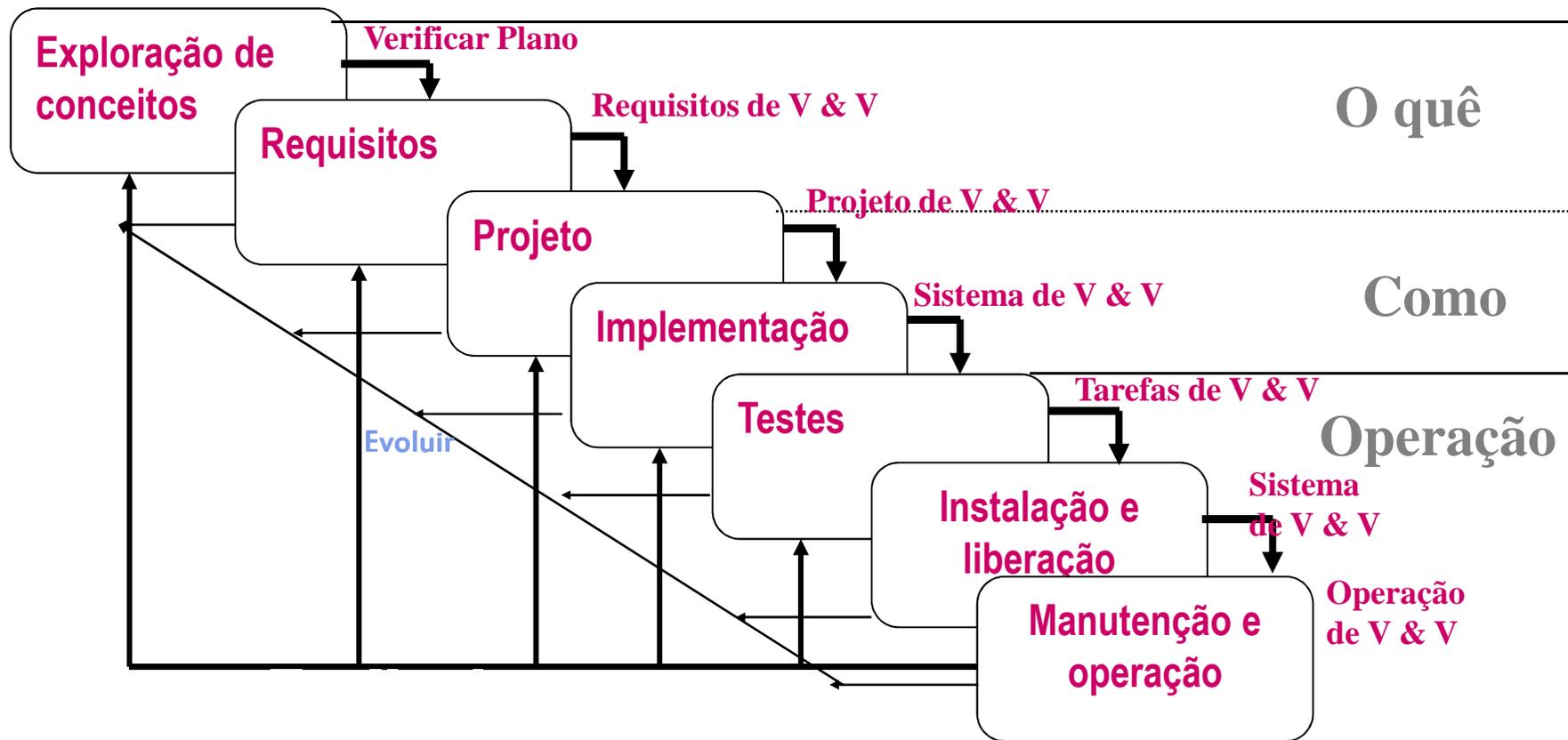




O Modelo Cascata

- 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

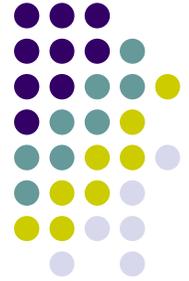
- tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação inicie

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: *erros, 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



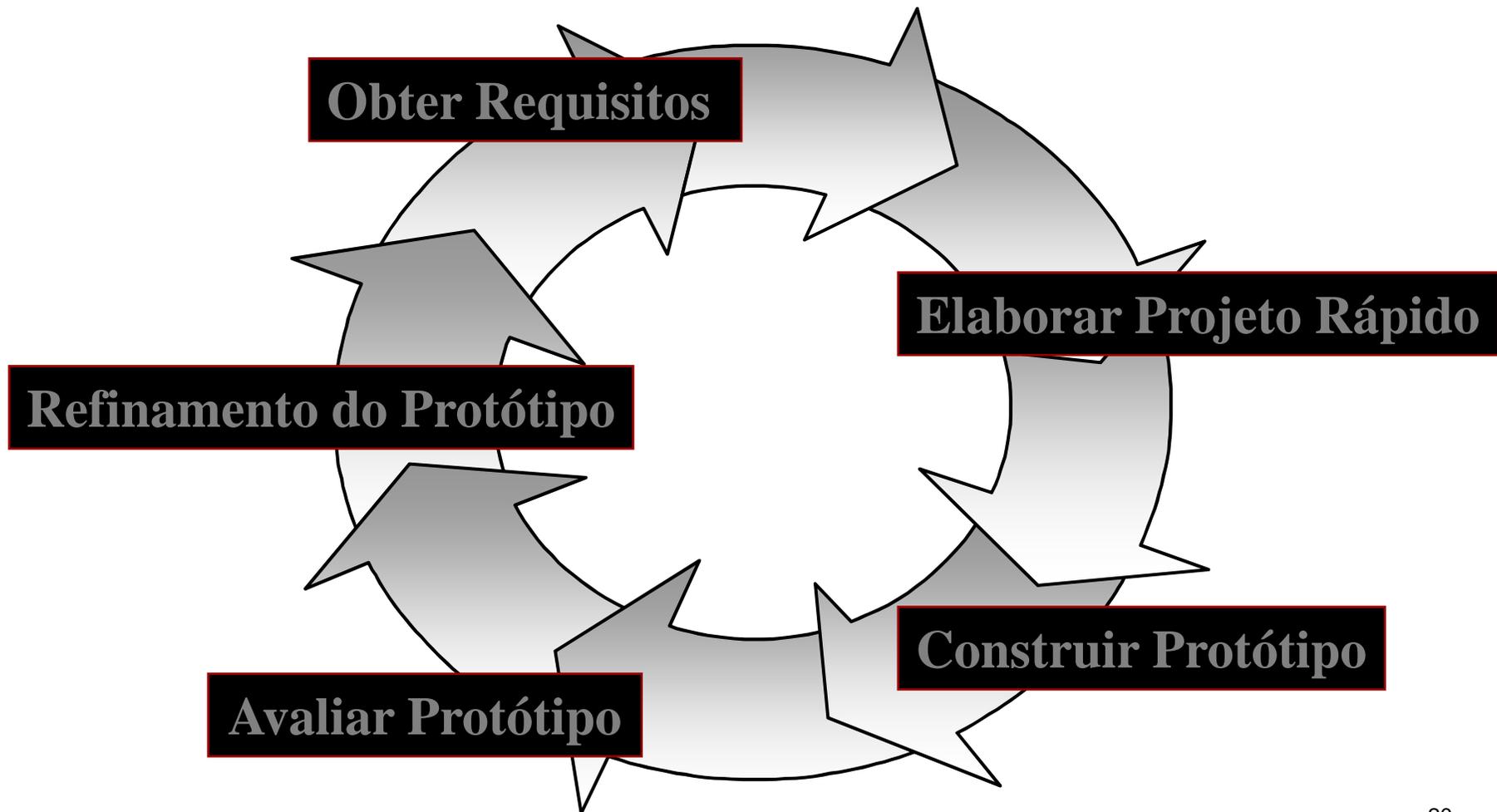


O Modelo de Prototipação

- 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



1- OBTENÇÃO DOS REQUISITOS:

desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais.

Refinar

o Rápido

Construir Protótipo

Avaliar Protótipo

O Paradigma de Prototipação

para obtenção dos requisitos



Obter Requisitos

Refinamento do Pro

2- PROJETO RÁPIDO:

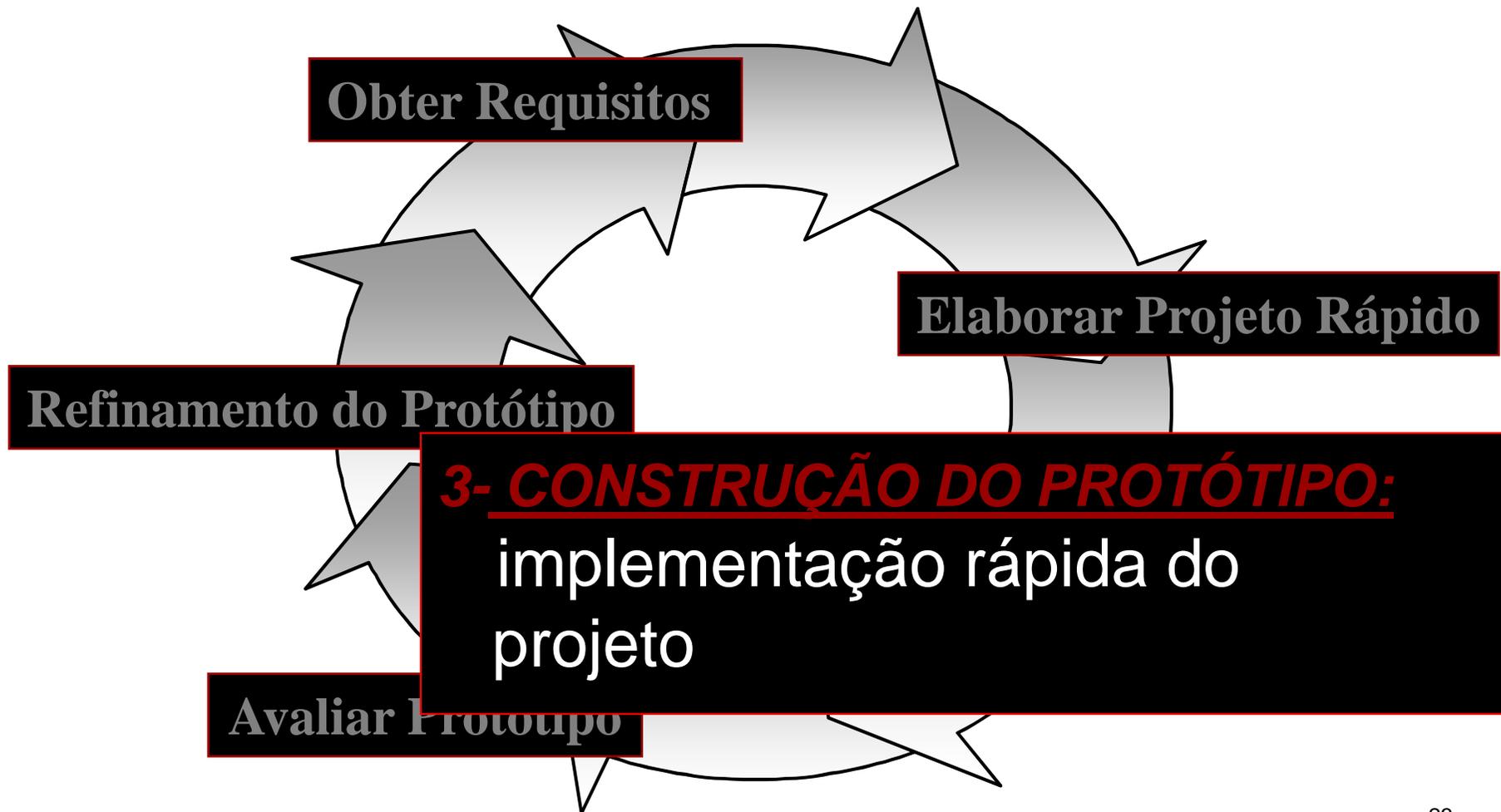
representação dos aspectos do software que são visíveis ao usuário (abordagens de entrada e formatos de saída)

Construir Protótipo

Avaliar Protótipo

O Paradigma de Prototipação

para obtenção dos requisitos



O Paradigma de Prototipação

para obtenção dos requisitos



O Paradigma de Prototipação

para obtenção dos requisitos



Obter Requisitos

5- REFINAMENTO DO PROTÓTIPO:

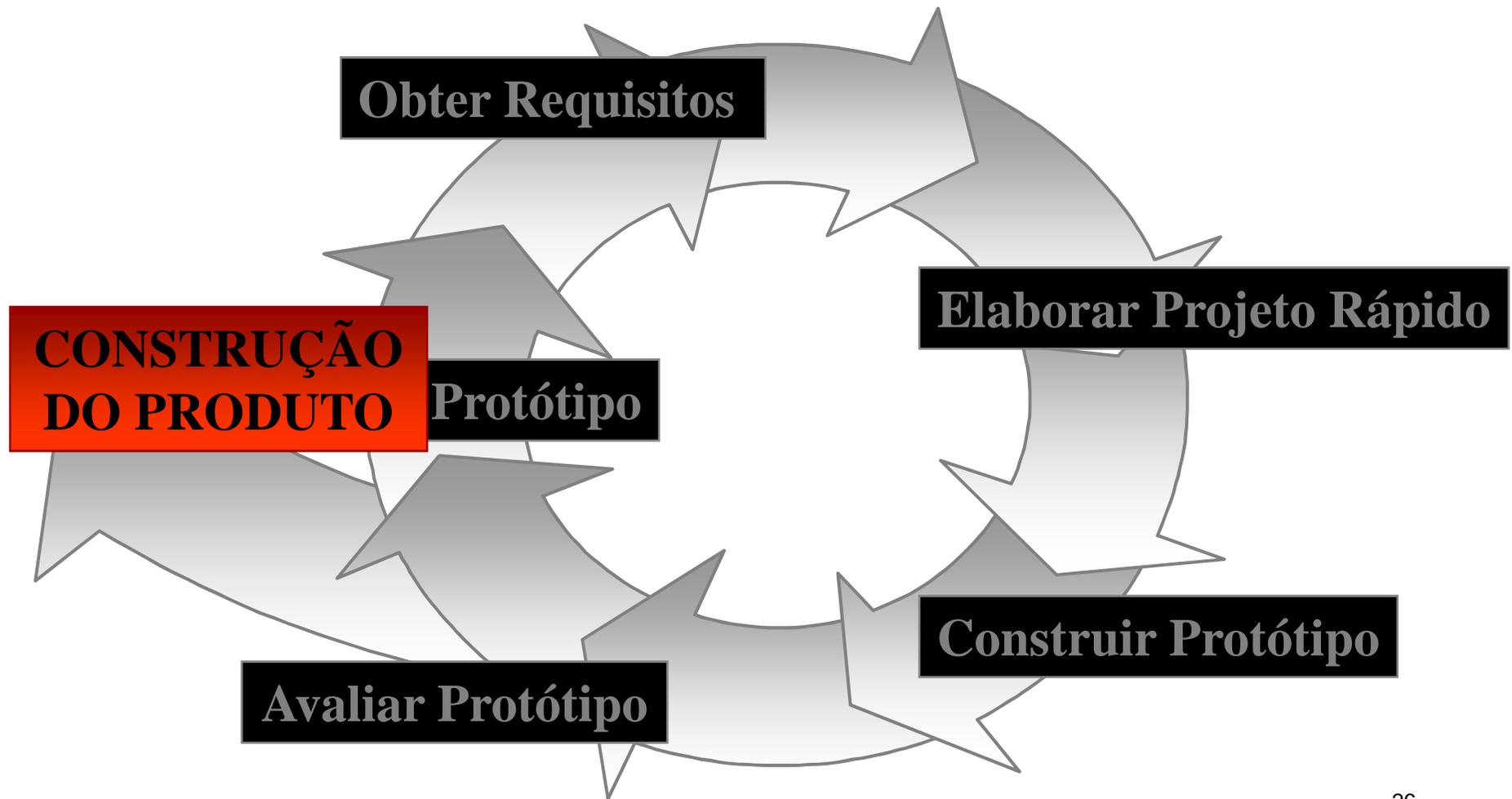
desenvolvedor refinam os requisitos do software a ser desenvolvido.

Rápido

Avaliar Protótipo

O Paradigma de Prototipação

para obtenção dos requisitos



O Paradigma de Prototipação

para obtenção dos requisitos



Obter 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.

Projeto Rápido

Protótipo

Avaliar Protótipo

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

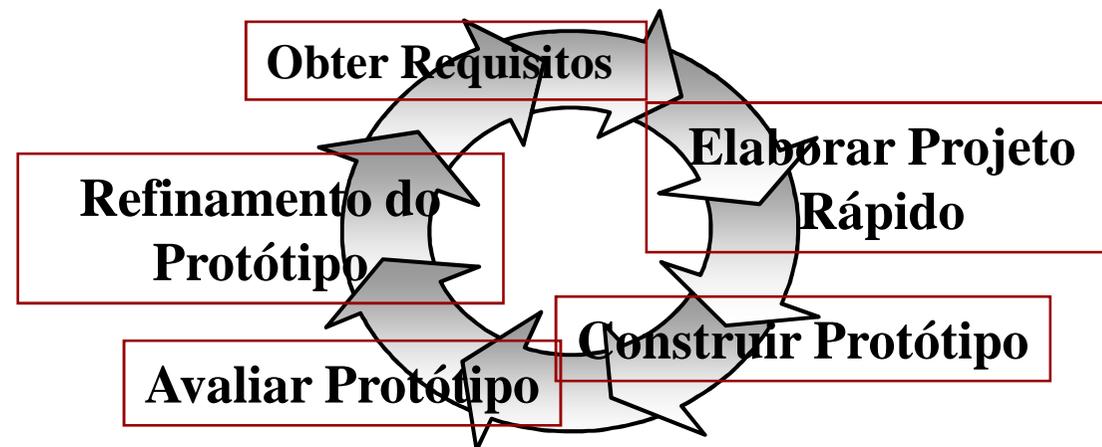


- ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- a chave é definir as regras do jogo logo no começo.
- o cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo para definir os requisitos



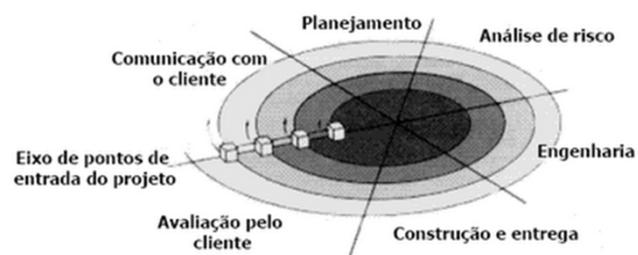
Exercício

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





O Modelo Espiral

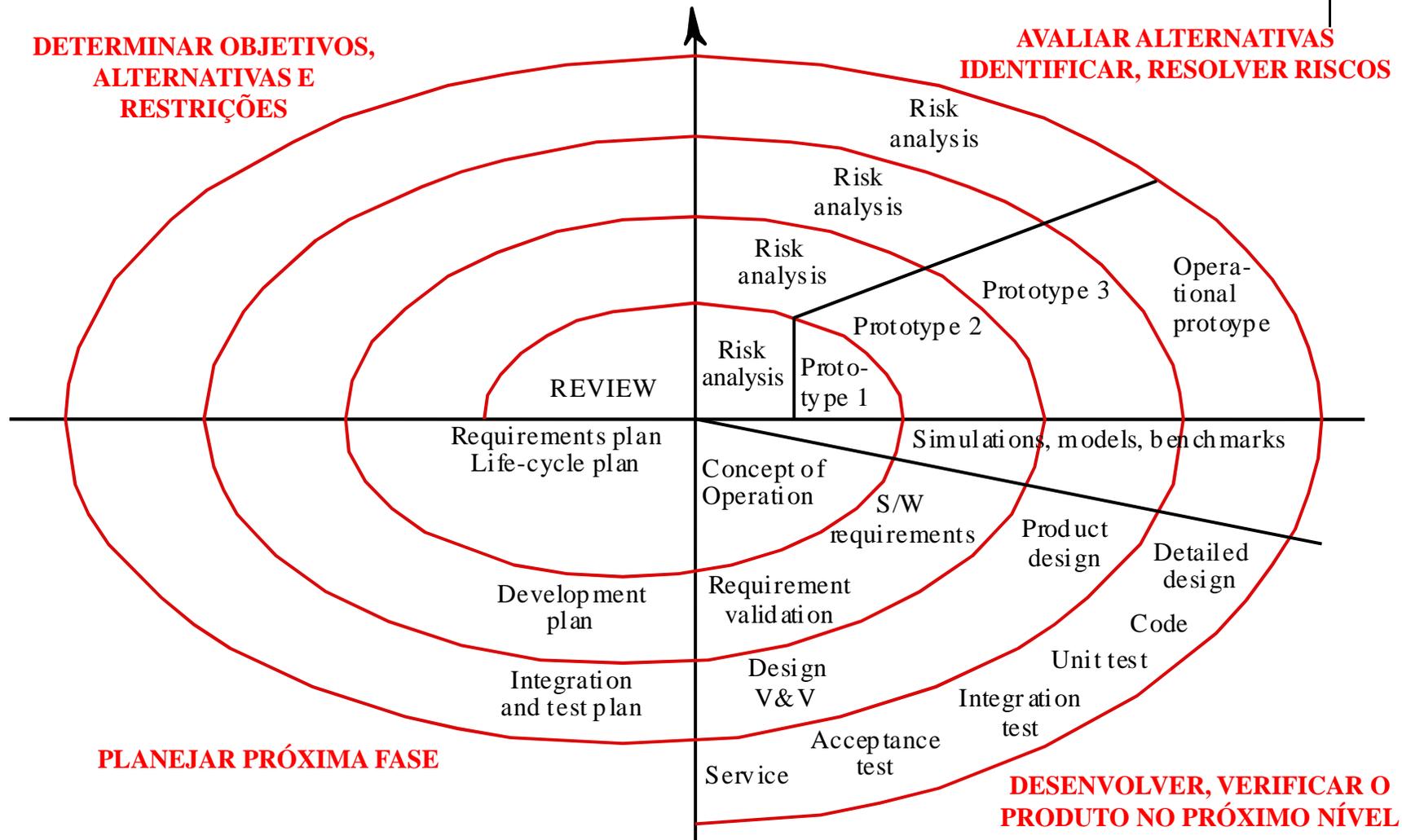


O Modelo Espiral (Boehm, 1986)



- 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)



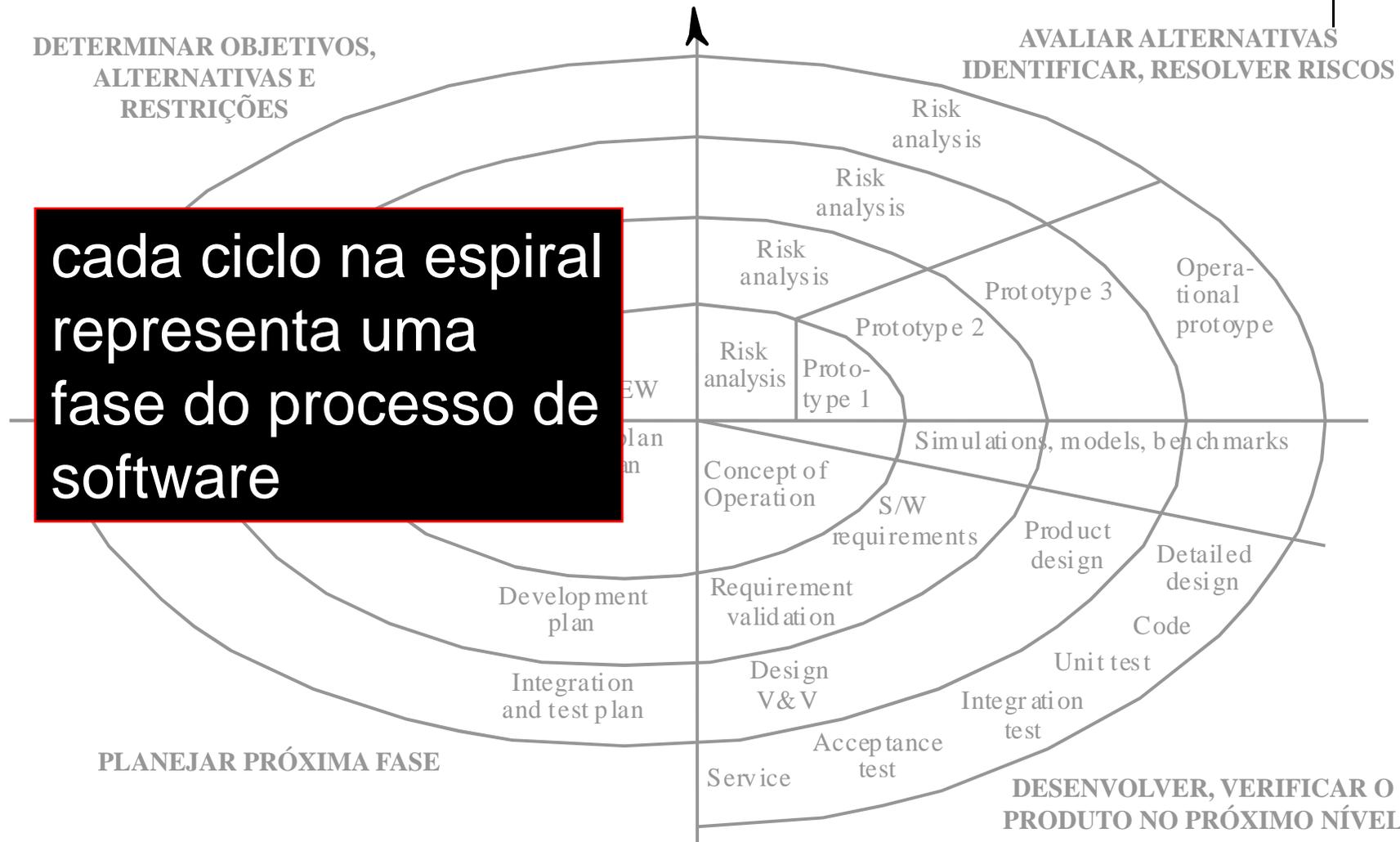
O Modelo Espiral (com 4 regiões)



DETERMINAR OBJETIVOS,
ALTERNATIVAS E
RESTRICÇÕES

AVALIAR ALTERNATIVAS
IDENTIFICAR, RESOLVER RISCOS

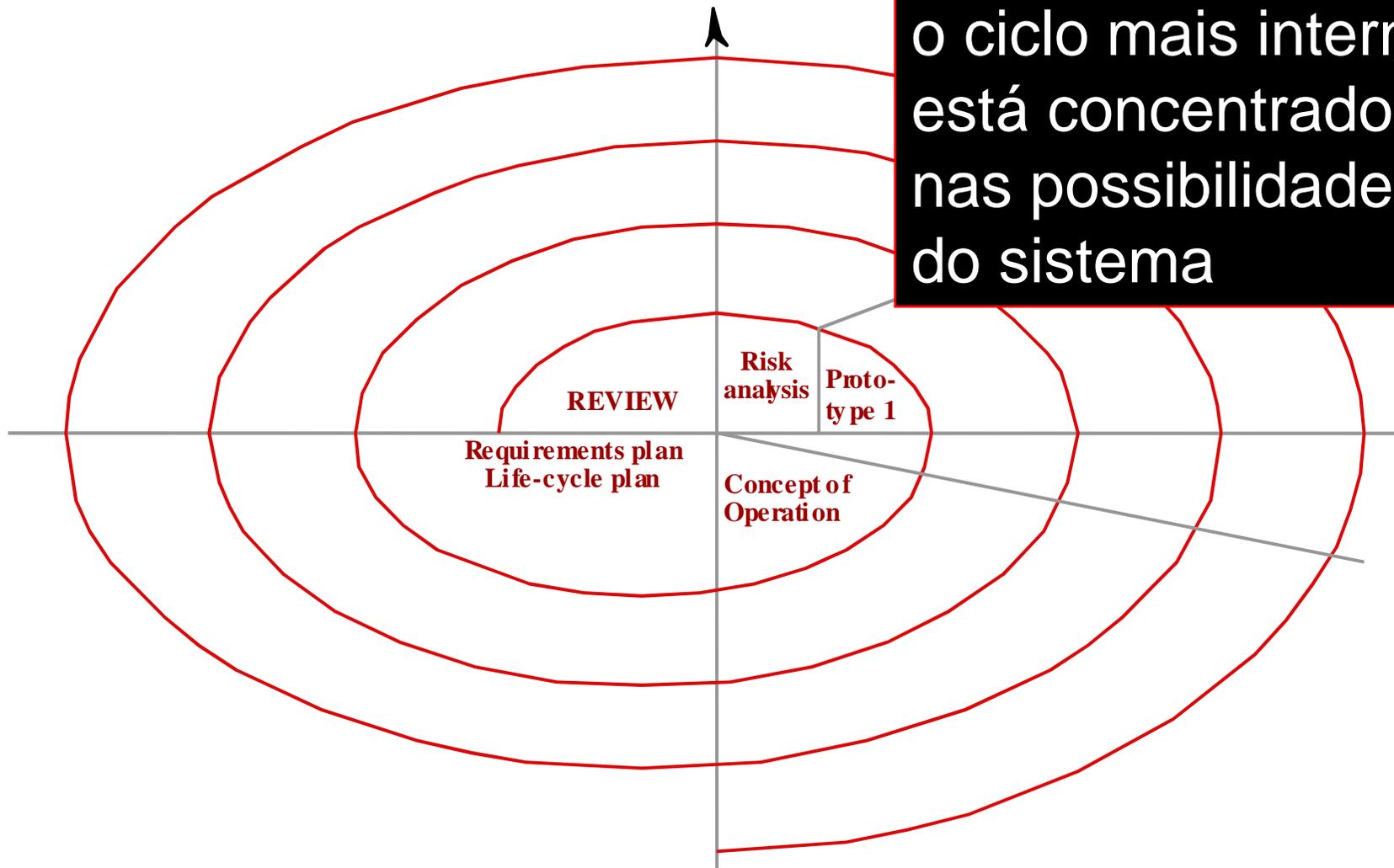
cada ciclo na espiral
representa uma
fase do processo de
software



O Modelo Espiral de Processo de Software



o ciclo mais interno está concentrado nas possibilidades do sistema



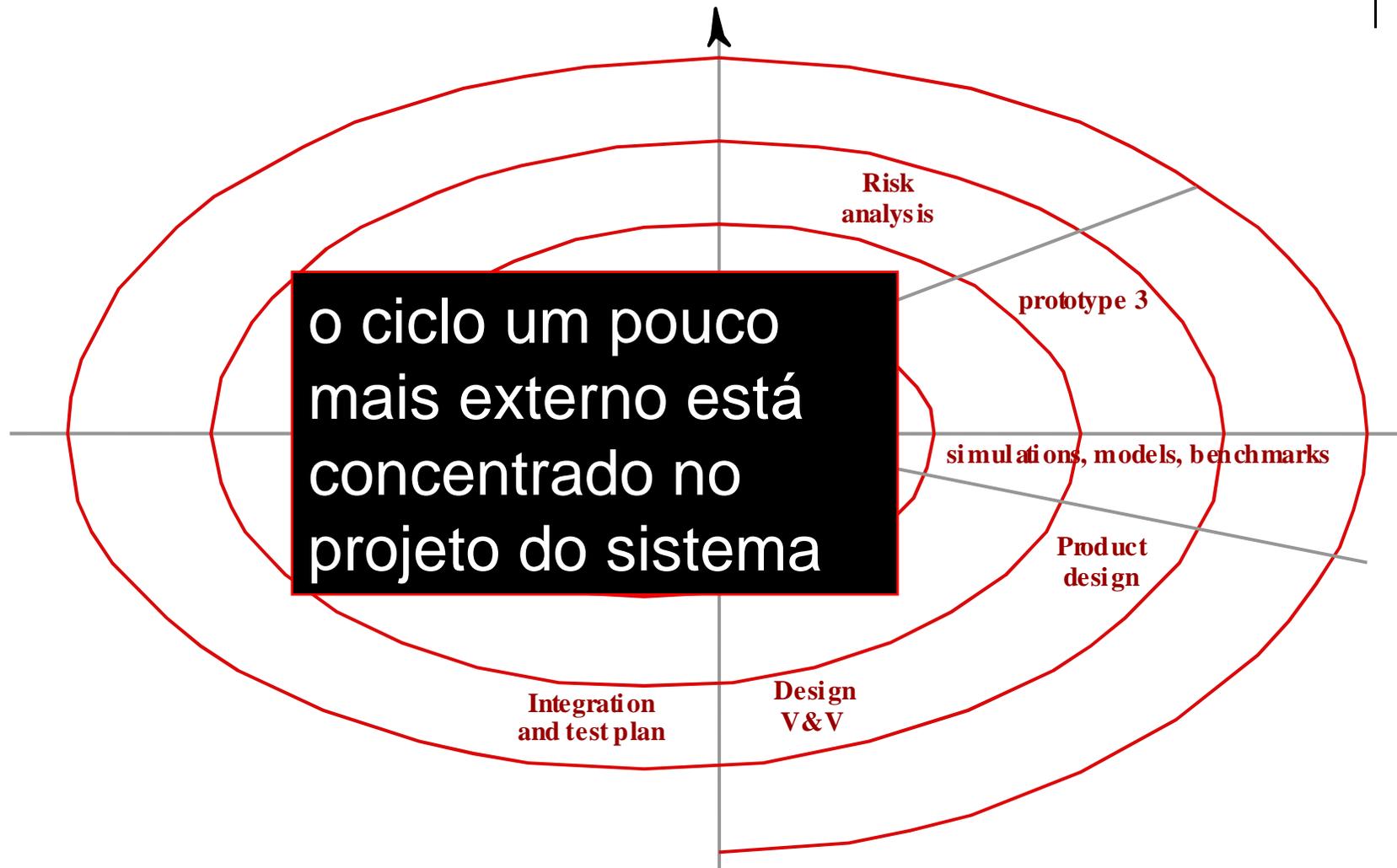
O Modelo Espiral de Processo de Software



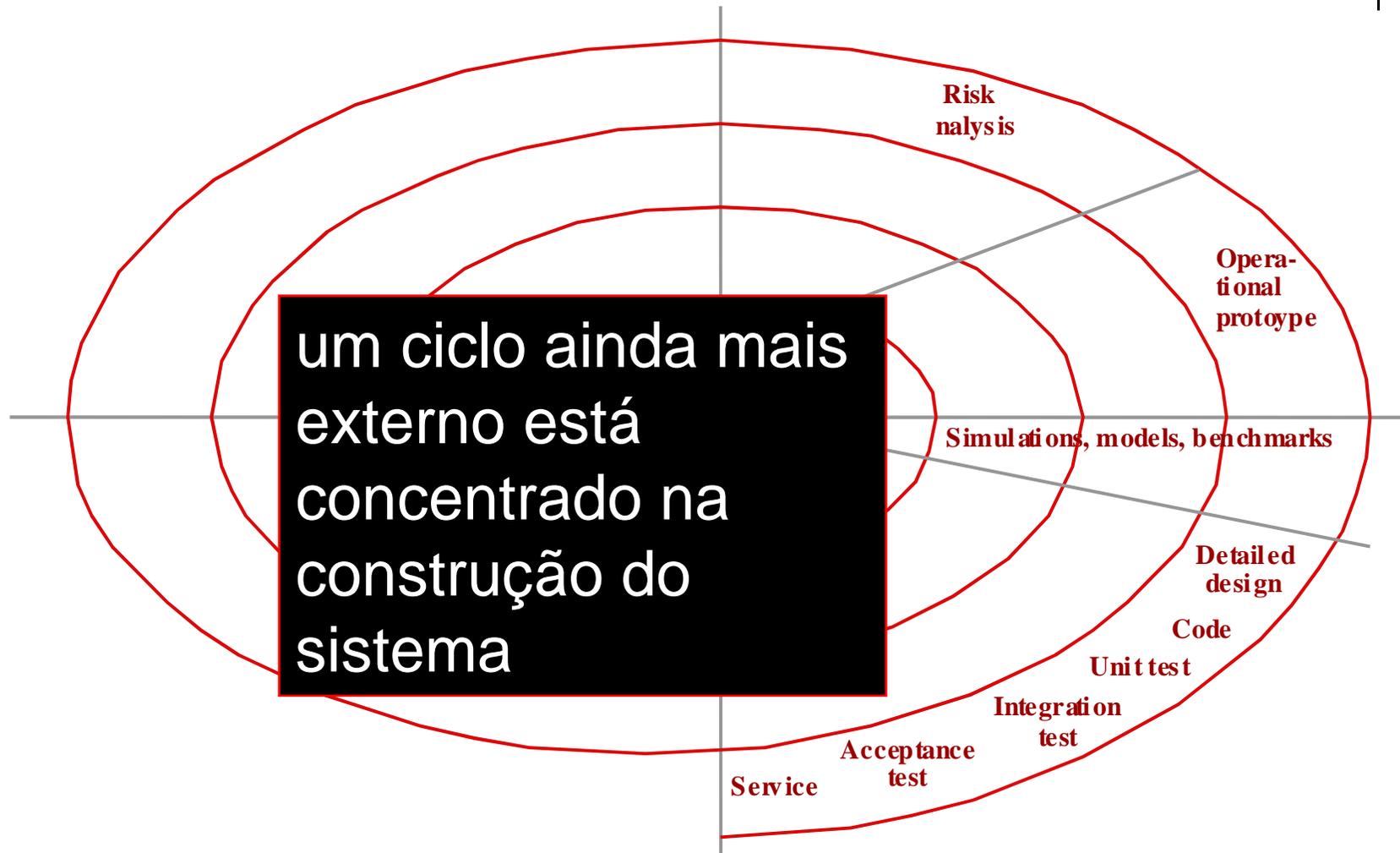
o próximo ciclo está concentrado na definição dos requisitos do sistema



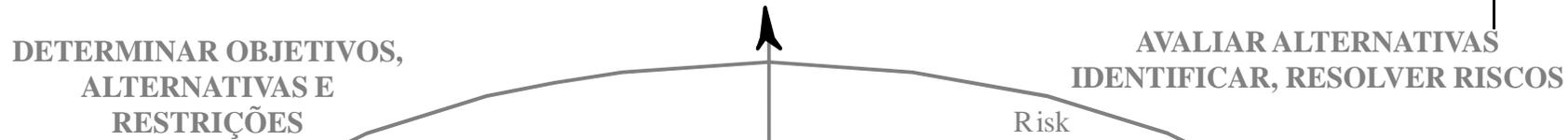
O Modelo Espiral de Processo de Software



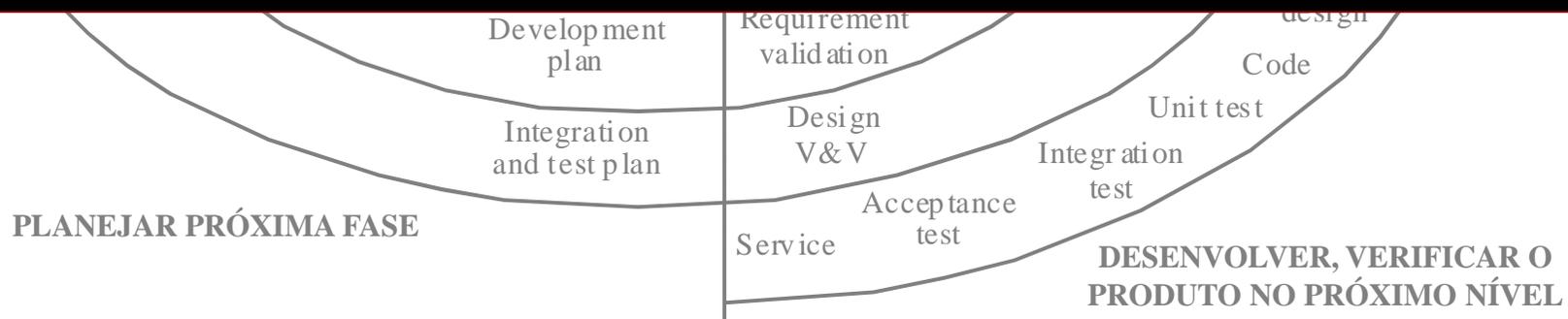
O Modelo Espiral de Processo de Software



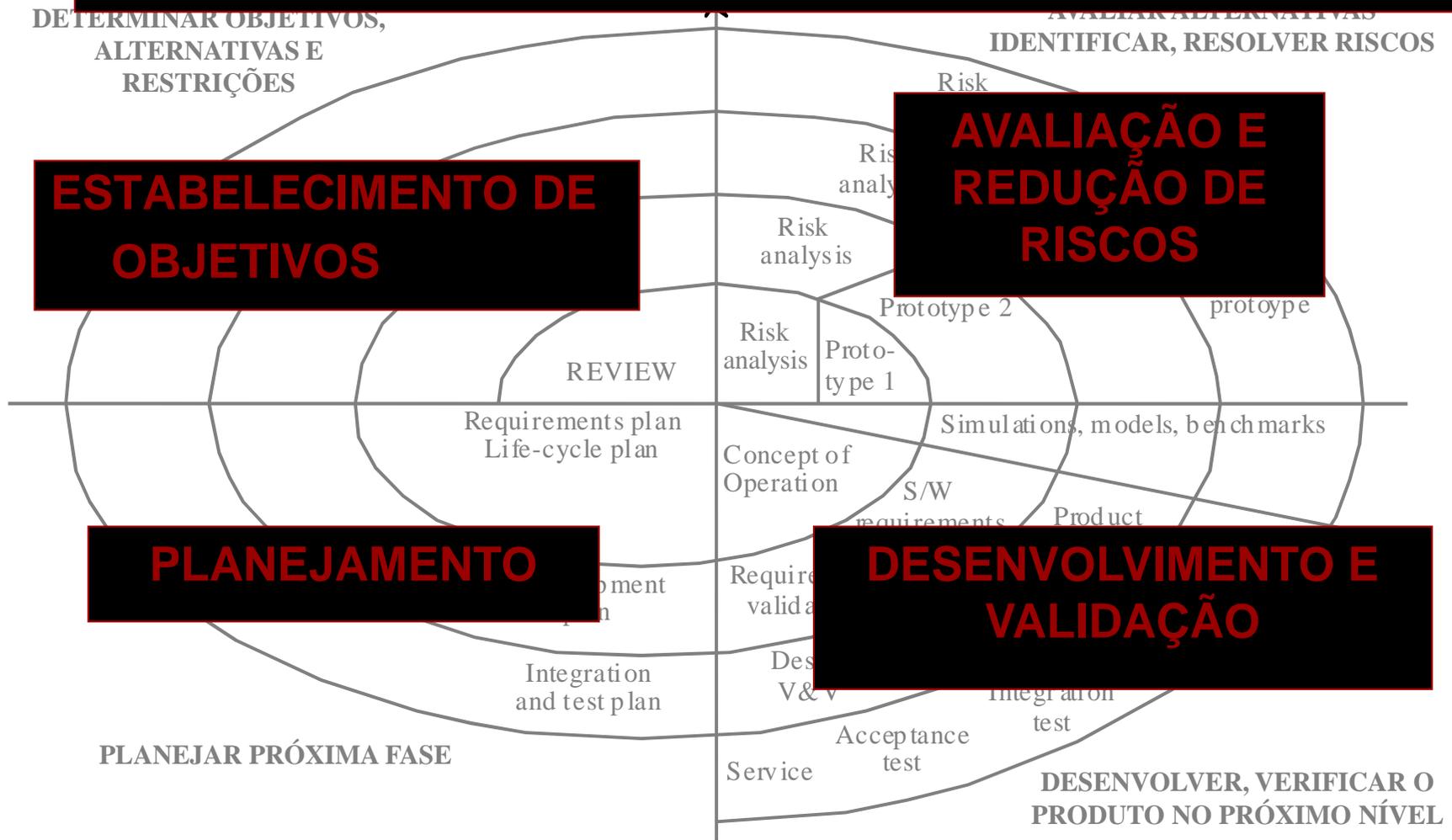
O Modelo Espiral (com 4 regiões)



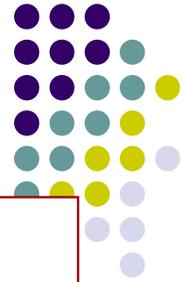
- não existem fases fixas no modelo
- as fases mostradas na figura são meramente exemplos
- a gerência decide como estruturar o projeto em fases



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



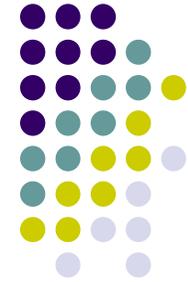
O Modelo Espiral de Processo de Software



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**

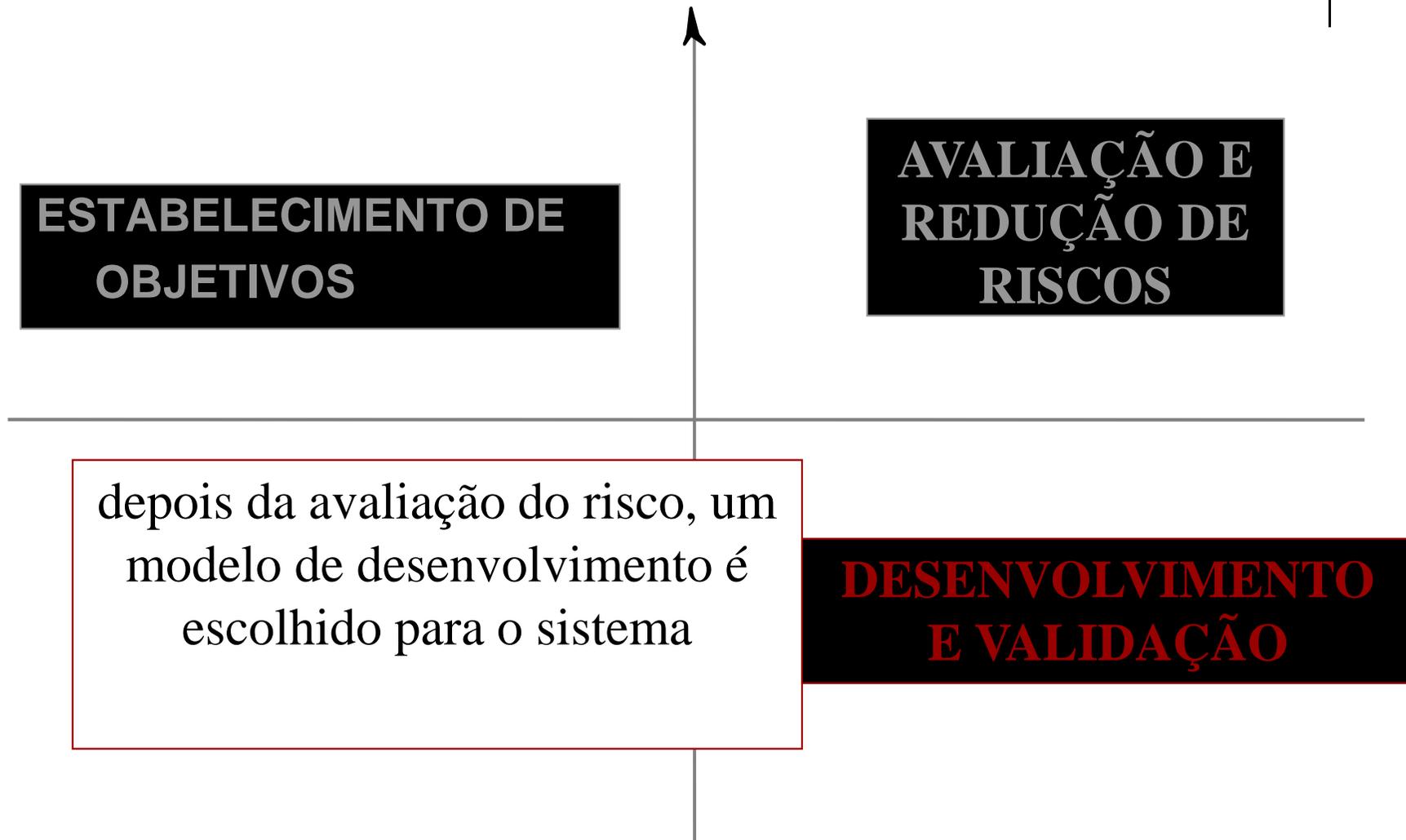
O Modelo Espiral de Processo de Software



para cada um dos riscos identificados, uma análise detalhada é executada. passos são tomados para reduzir o risco

**AVALIAÇÃO E
REDUÇÃO DE
RISCOS**

O Modelo Espiral de Processo de Software



O Modelo Espiral de Processo de Software



COLOCAÇÃO DE OBJETIVOS

AVALIAÇÃO E REDUÇÃO DE RISCOS

PLANEJAMENTO

o projeto é revisto e é tomada uma decisão de continuidade se é decidido continuar, são projetados planos para a próxima fase do projeto (próximo “loop”)

NTO



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

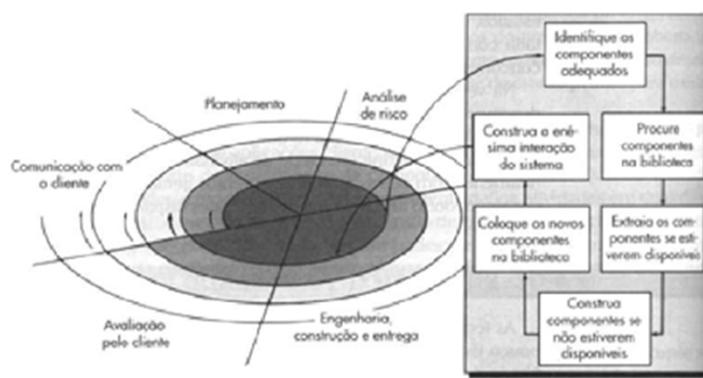
O Modelo Espiral



- 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

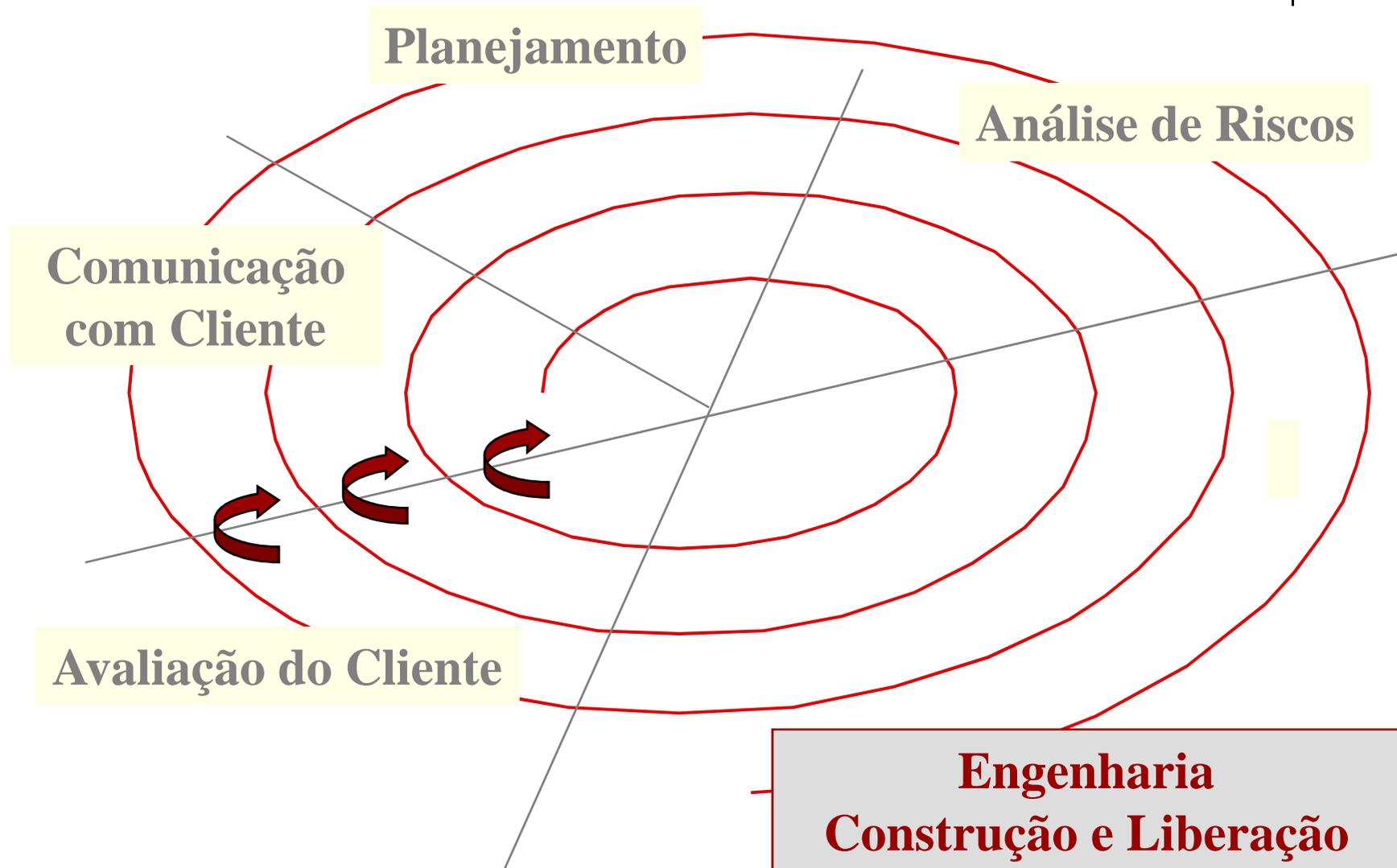


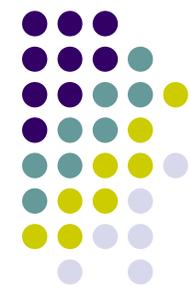
O Modelo Baseado em Componentes



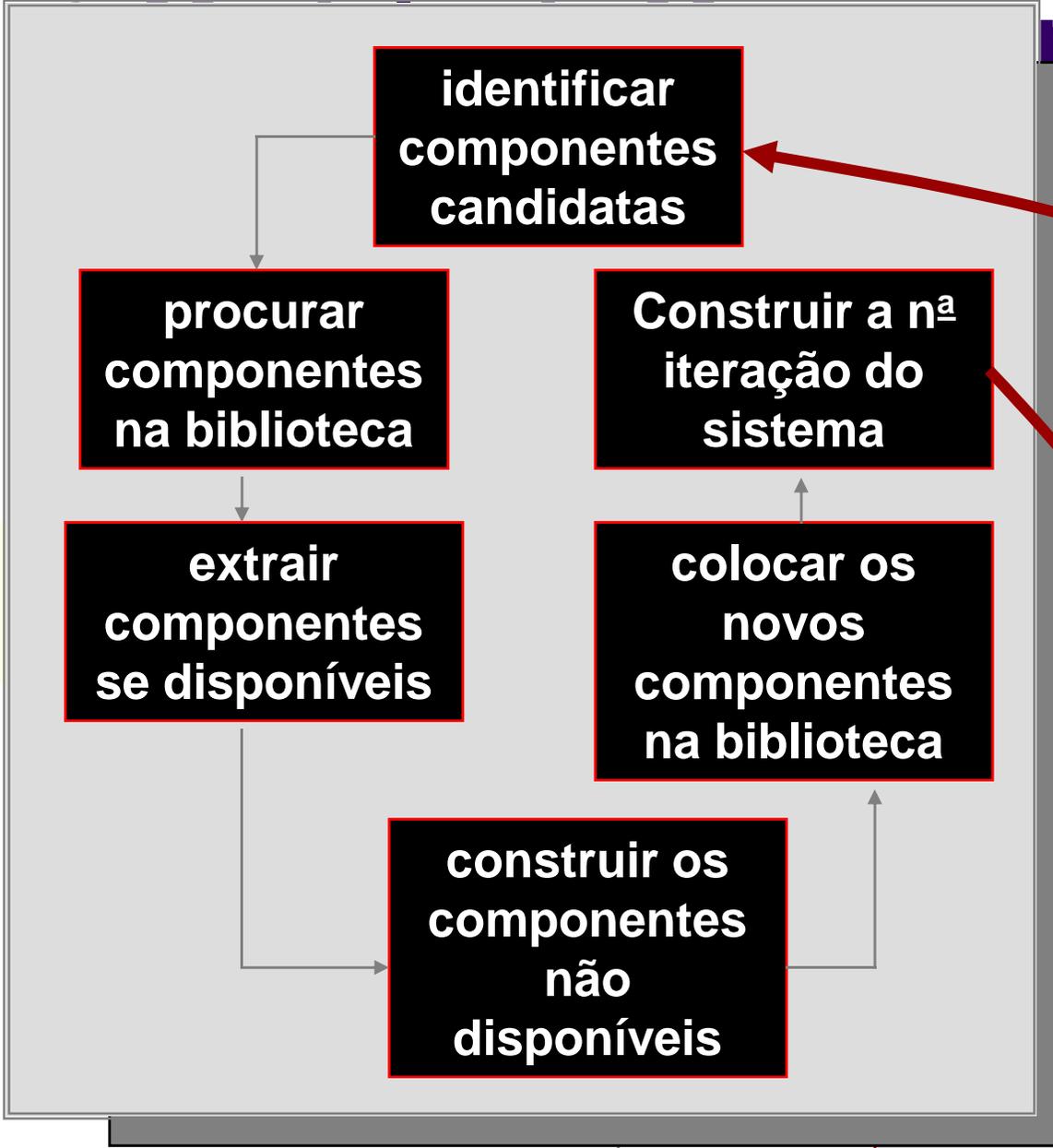
- Utiliza **tecnologias orientadas a objeto**
- Quando projetadas e implementadas apropriadamente as **classes** orientadas a objeto são **reutilizáveis** em diferentes aplicações e arquiteturas de sistema
- O modelo de montagem de componentes incorpora muitas das características do **modelo espiral**.

O Modelo de Montagem de Componentes





m de



Análise de Riscos

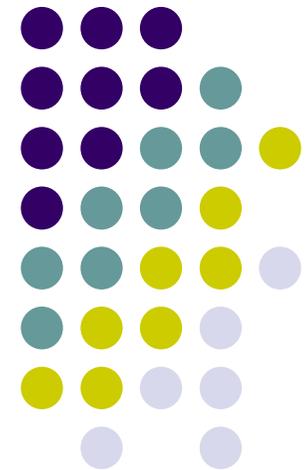
**Engenharia
Construção e Liberação**

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

Processo Unificado da *Rational*





O que é o RUP?

- O nome é uma abreviação de Rational Unified Process
 - mas na verdade é
 - Processo + Métodos + Linguagem (UML)
 - e os autores argumentam que é
 - *Framework para gerar processos*

O que é o RUP?



- Conjunto de atividades
 - bem definidas
 - com responsáveis
 - com artefatos de entrada e saída
 - com dependências entre as mesmas e ordem de execução
 - com modelo de ciclo de vida
 - descrição sistemática de como devem ser realizadas
 - guias (de ferramentas ou não), *templates*
 - utilizando diagramas de UML

Características Principais do RUP

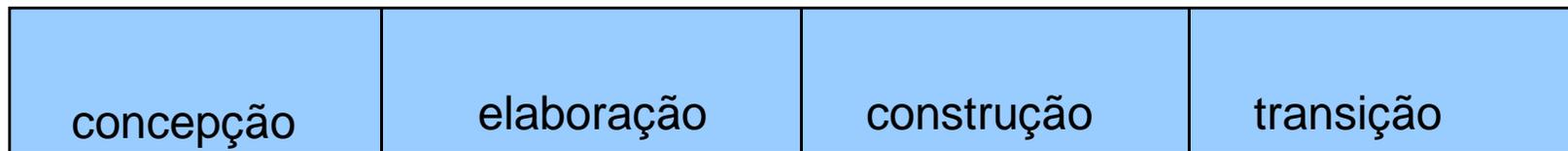


- O desenvolvimento de sistemas seguindo o RUP é
 - Iterativo e incremental
 - Guiado por casos de uso (*use cases*)
 - Baseado na arquitetura do sistema

O RUP é iterativo e incremental



- O ciclo de vida de um sistema consiste de quatro fases:



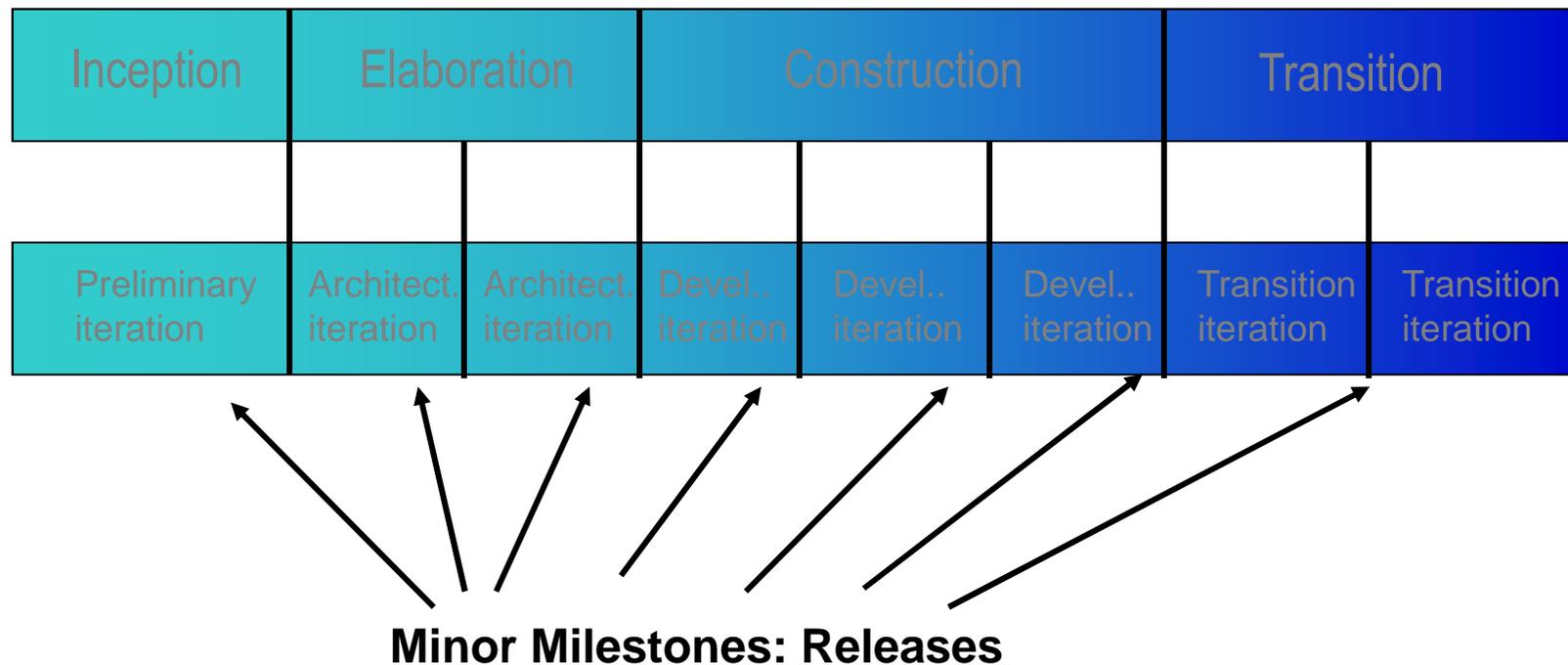
tempo

- Concepção (define o escopo do projeto)
- Elaboração (detalha os requisitos e a arquitetura)
- Construção (desenvolve o sistema)
- Transição (implanta o sistema)

O RUP é iterativo e incremental



- Cada fase é dividida em iterações:

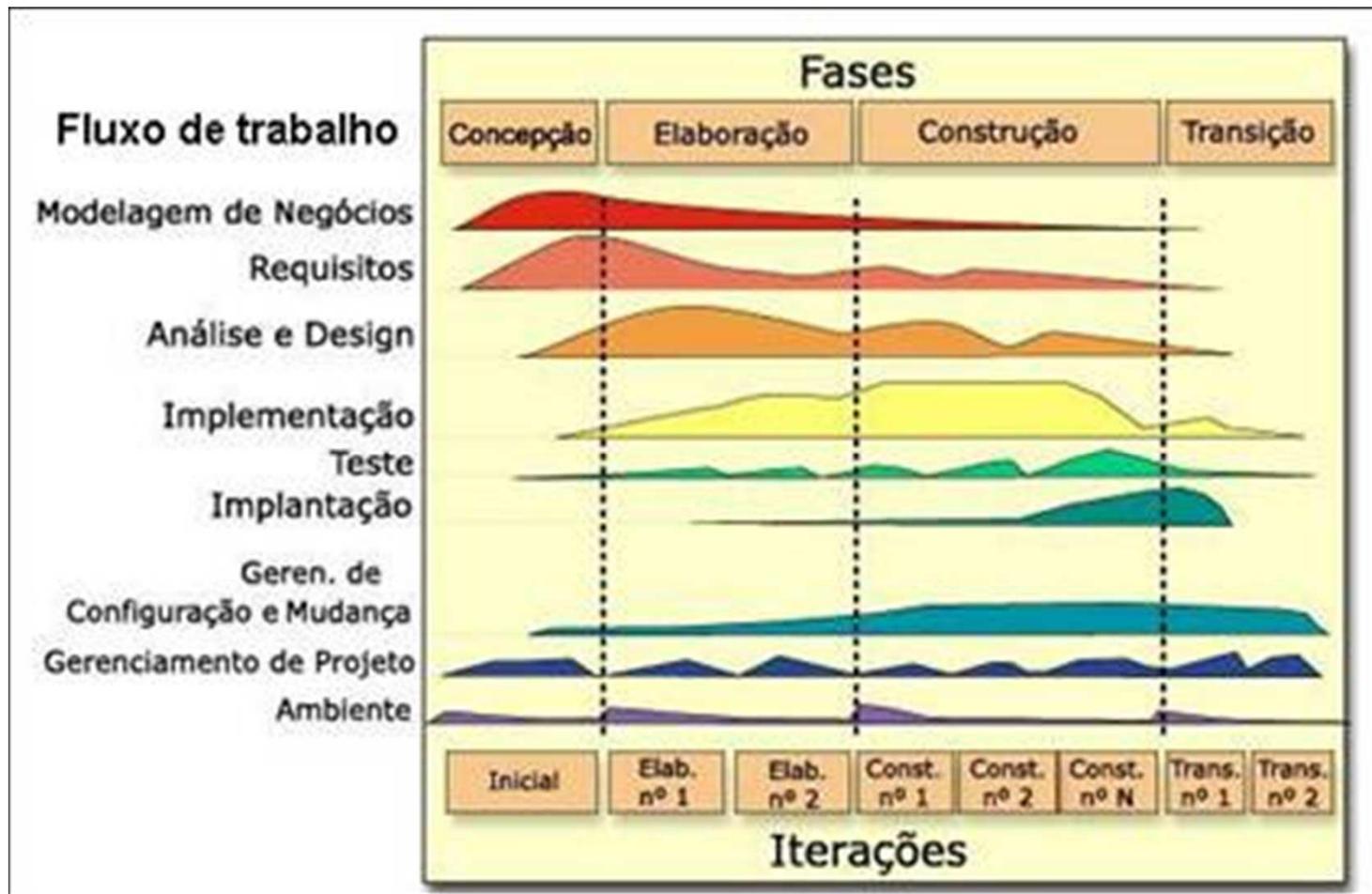


O RUP é iterativo e incremental



- Cada iteração
 - é planejada
 - realiza uma seqüência de atividades (de elicitação de requisitos, análise e projeto, implementação, etc.) distintas
 - geralmente resulta em uma versão executável do sistema
 - é avaliada segundo critérios de sucesso previamente definidos

O RUP é iterativo e incremental



Iteração e Workflow

Passos dentro de uma iteração

Core Workflows

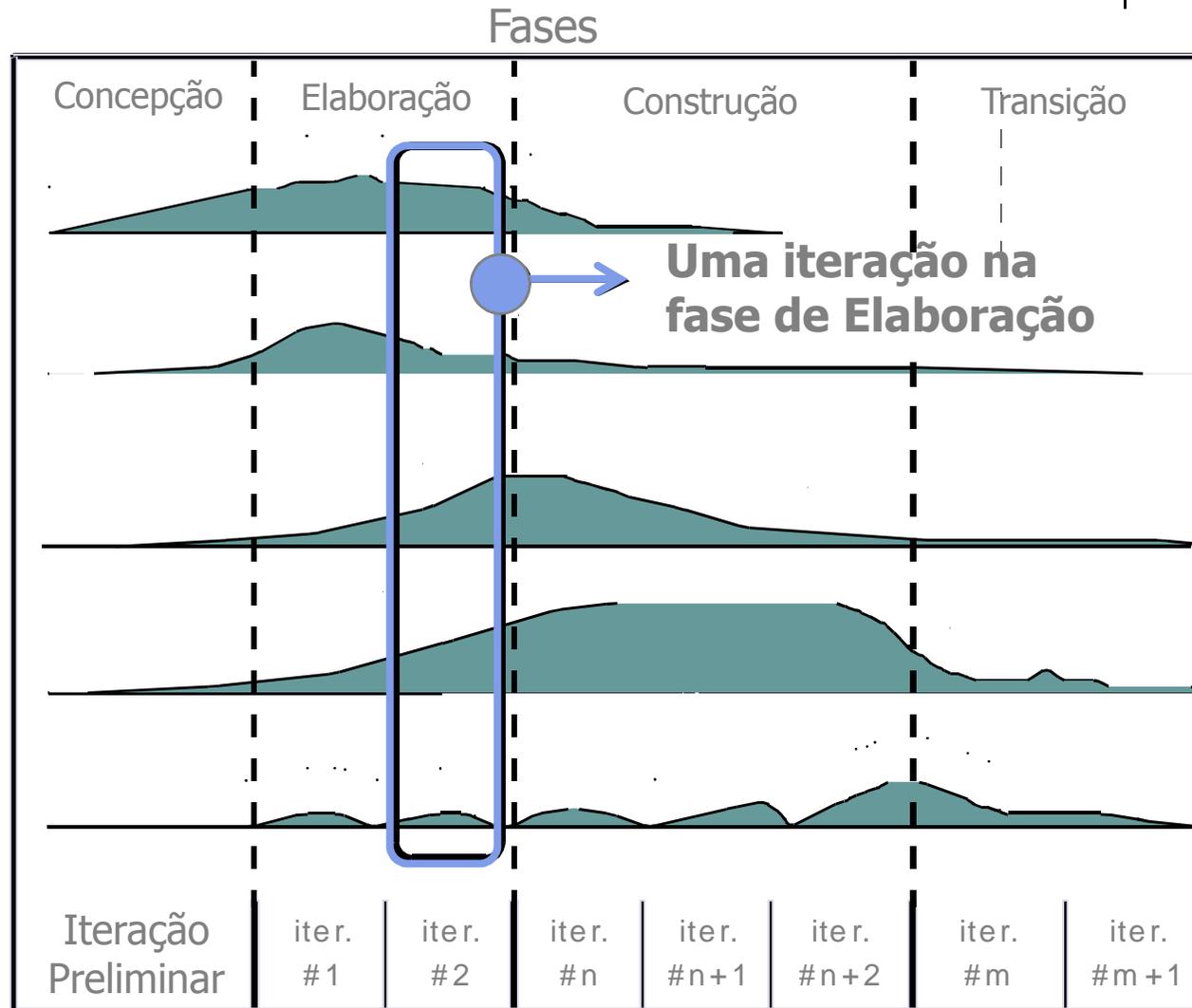
Requisito

Análise

Desenho

Implementação

Teste



O RUP é guiado por casos de uso



- Os casos de uso não servem apenas para definir os requisitos do sistema
- Várias atividades do RUP são guiadas pelos casos de uso:
 - planejamento das iterações
 - criação e validação do modelo de projeto
 - planejamento da integração do sistema
 - definição dos casos de teste

O RUP é baseado na arquitetura do sistema



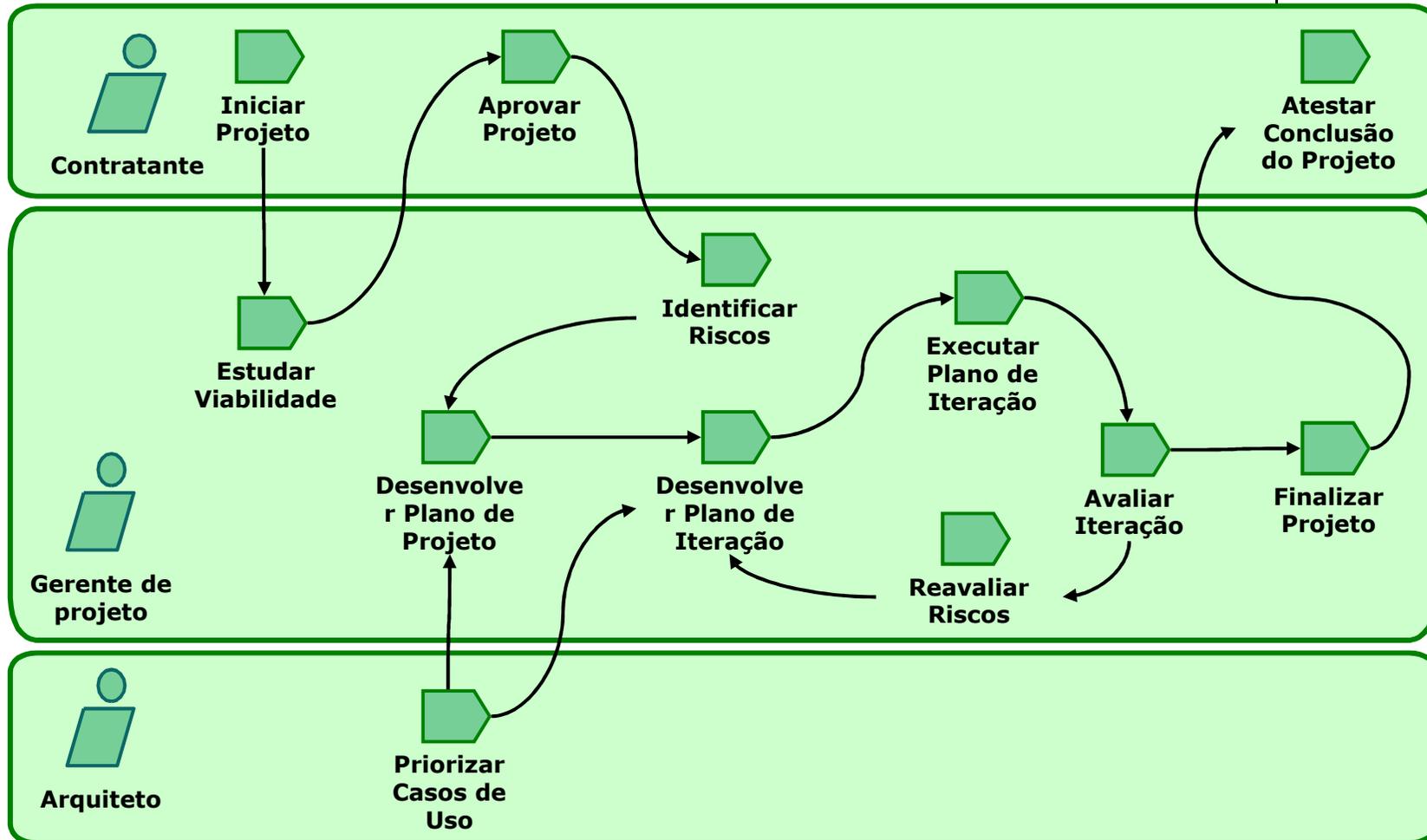
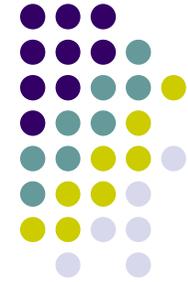
- Arquitetura
 - visão geral do sistema em termos dos seus subsistemas e como estes se relacionam
- A arquitetura é prototipada e definida logo nas primeiras iterações
- O desenvolvimento consiste em complementar a arquitetura
- A arquitetura serve para definir a organização da equipe de desenvolvimento e identificar oportunidades de reuso



Organização do RUP

- Fluxos de atividades
- Atividades
 - passos
 - entradas e saídas
 - guias (de ferramentas ou não), templates
- Responsáveis (papel e perfil, não pessoa)
- Artefatos

Exemplo de Fluxo: Planejamento e Gerenciamento





Resumo

- O RUP é:
- iterativo e incremental
- guiado por casos de uso
- baseado na arquitetura do sistema
- organizado em fases, iterações, fluxos, atividades e passos

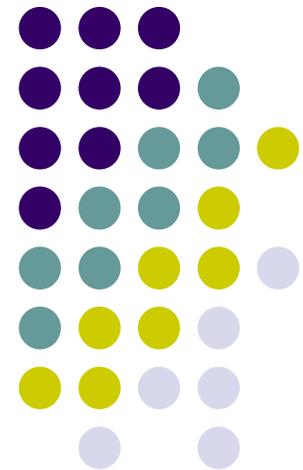
Vamos Navegar pelo RUP



- Entre no site:

<http://www.wthreeex.com/rup/portugues/index.htm>

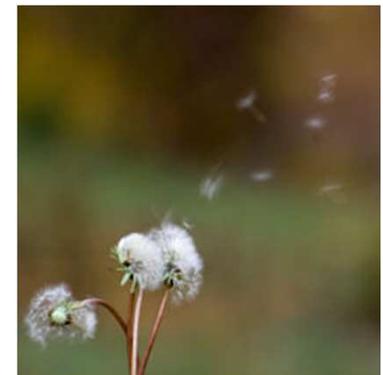
Métodos Ágeis



Novos ventos no mundo do Desenvolvimento de Software



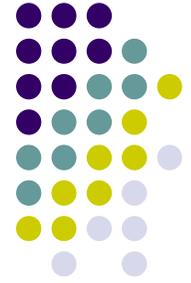
- Sociedade demanda
 - grande quantidade de sistemas/aplicações
 - software complexo, sistemas distribuídos, heterogêneos
 - requisitos mutantes (todo ano, todo mês, todo dia)
- Mas, infelizmente,
 - não há gente suficiente para desenvolver tanto software com qualidade.



Problemas



- Com metodologias de desenvolvimento
 - Supõem que é possível prever o futuro
 - Pouca interação com os clientes
 - Ênfase em burocracias (documentos, formulários, processos, controles rígidos, etc.)
 - Avaliação do progresso baseado na evolução da burocracia e não do código
- Com software
 - Grande quantidade de erros
 - Falta de flexibilidade



Como resolver esse impasse?

- Melhores Tecnologias
 - Padrões de Projeto (reutilização de idéias)
 - Componentes (reutilização de código)
 - Middleware (aumenta a abstração)
- Melhores Metodologias
 - Métodos Ágeis
 - outras... (RUP, relacionadas a CMM, etc.)

Métodos Ágeis de Desenvolvimento de Software



- Movimento iniciado por programadores experientes e consultores em desenvolvimento de software.
- Questionam e se opõe a uma série de mitos/práticas adotadas em abordagens tradicionais de Engenharia de Software e Gerência de Projetos.
- Manifesto Ágil:
 - Assinado por 17 desenvolvedores em Utah em fevereiro/2001.

O Manifesto do *Desenvolvimento Ágil de Software*

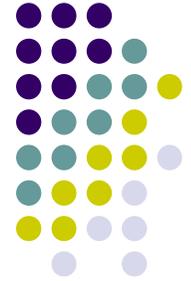


1. **Indivíduos e interações** são mais importantes que *processos e ferramentas*.
2. **Software funcionando** é mais importante do que *documentação completa e detalhada*.
3. **Colaboração com o cliente** é mais importante do que *negociação de contratos*.
4. **Adaptação a mudanças** é mais importante do que *seguir o plano inicial*.



Princípios do Manifesto Ágil

- Objetivo: satisfazer o cliente entregando, rapidamente e com frequência, sistemas com algum valor.
 - Entregar versões funcionais em prazos curtos.
 - Estar preparado para requisitos mutantes.
 - Pessoal de negócios e desenvolvedores juntos.
 - Troca de informações através de conversas diretas.



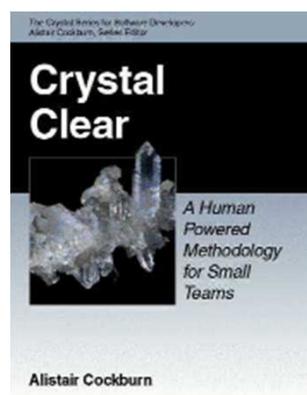
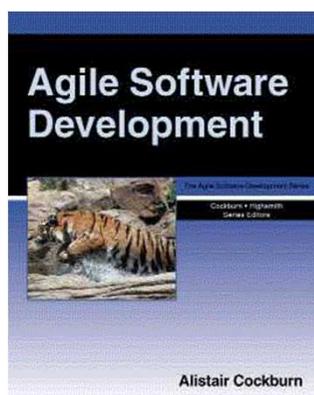
Principais Métodos Ágeis

- Crystal (uma família de métodos)
- Scrum
- Programação eXtrema (XP)
- Adaptive Software Development
- Feature Driven Development
- etc.



A família *Crystal* de Métodos

- Criada por Alistair Cockburn
- <http://alistair.cockburn.us/crystal>
- Editor da série *Agile Software Development* da Addison-Wesley.





Scrum



Definição informal:

Estratégia em um jogo de rugby onde jogadores colocam uma bola quase perdida novamente em jogo através de trabalho em equipe.

Scrum



- Scrum é um esqueleto de processo que contém grupos de práticas e papéis pré-definidos. Os principais papéis são:
 - ScrumMaster, que mantém os processos (normalmente no lugar de um gerente de projeto);
 - Proprietário do Produto, ou Product Owner, que representa os stakeholders e o negócio;
 - a Equipe, ou Team, um grupo multifuncional com cerca de 7 pessoas e que fazem a análise, projeto, implementação, teste etc.

Programação eXtrema XP



- Metodologia de desenvolvimento de software aperfeiçoada nos últimos 5 anos.
- Ganhou notoriedade a partir da OOPSLA'2000.
- Nome principal: Kent Beck
- Também importante: Ward Cunningham

A Quem se Destina os Métodos Ágeis



- Grupos de 2 a 10 programadores
- Projetos de 1 a 36 meses (calendário)
- De 1000 a 250 000 linhas de código

Características Comuns dos Métodos Ágeis



- Coloca o foco
 - Na entrega freqüente de sub-versões do software [funcionando] para o cliente.
 - Nos seres humanos que desenvolvem o software.
- Retira o foco de
 - Processos rígidos e burocratizados.
 - Documentações e contratos detalhados.
 - Ferramentas que são usadas pelos seres humanos.

Para escolha de um Modelo de Processo de Software:



- **natureza** do projeto e da aplicação
- **métodos** e **ferramentas** a serem usados
- **controles** e **produtos** que precisam ser entregues



Exercício

- Você e mais 4 sócios resolveram criar uma empresa de desenvolvimento de Software e você ficou responsável por definir algumas áreas da empresa. Cada questão abaixo será a sua parte para a criação desta empresa.
 - 1) Defina uma abordagem para o desenvolvimento de software que compreenda os benefícios dos modelos de desenvolvimento clássicos (Cascata, prototipação e Espiral). Inclua as características de outros modelos se achar necessários.
 - 2) Defina uma abordagem para fazer a análise da viabilidade de sistemas a serem desenvolvidos. Esta abordagem consiste de uma relação das atividades em ordem cronológica para analisar a viabilidade de qualquer sistema que for desenvolvido pela empresa.
 - 3) O seu primeiro sistema a ser desenvolvido pela nova empresa de desenvolvimento de software será para a Farmácia “Pague Pouco”. Eles querem informatizar o sistema de vendas e controle de estoque, desejam armazenar a seguinte informação: qual vendedor vendeu qual produto para qual cliente. Relate, de acordo com o modelo de software criado na primeira questão e com a abordagem de análise da segunda, quais as atividades deverão ser realizadas para o desenvolvimento do sistema da Farmácia “Pague Pouco”.