Engenharia de Software I
2017.2

Modelos de Processo de Software

CECOMP

Ricardo Argenton Ramos
ricargentonramos@gmail.com

A Engenharia de Software

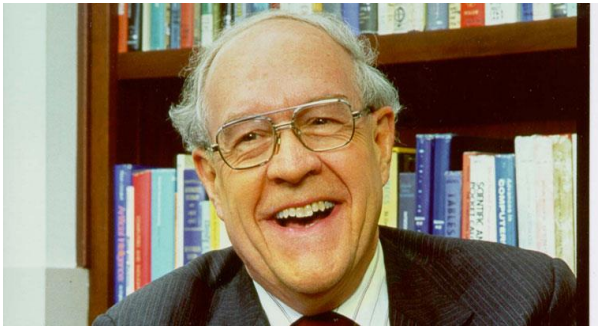
Uma Tecnologia em Camadas

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

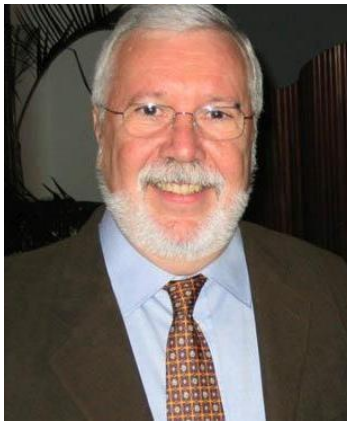
pode ser vista como uma abordagem dedesenvolvimento de software elaborada com disciplina e métodos bem definidos.

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



Fred Brooks, Jr.

Donald D. Cowan



Carlos Lucena

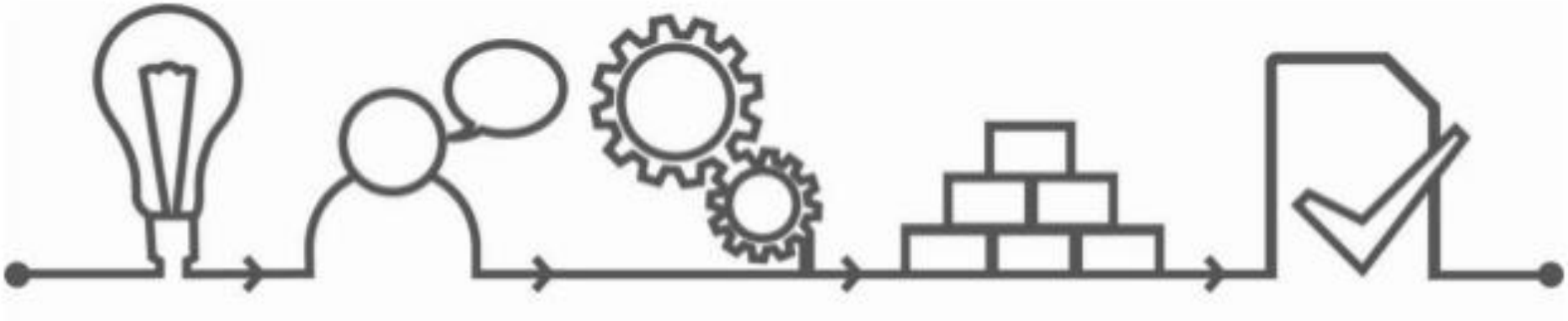
Roger Pressman

Barbara Kitchenham

Shari Lawrence Pfleeger

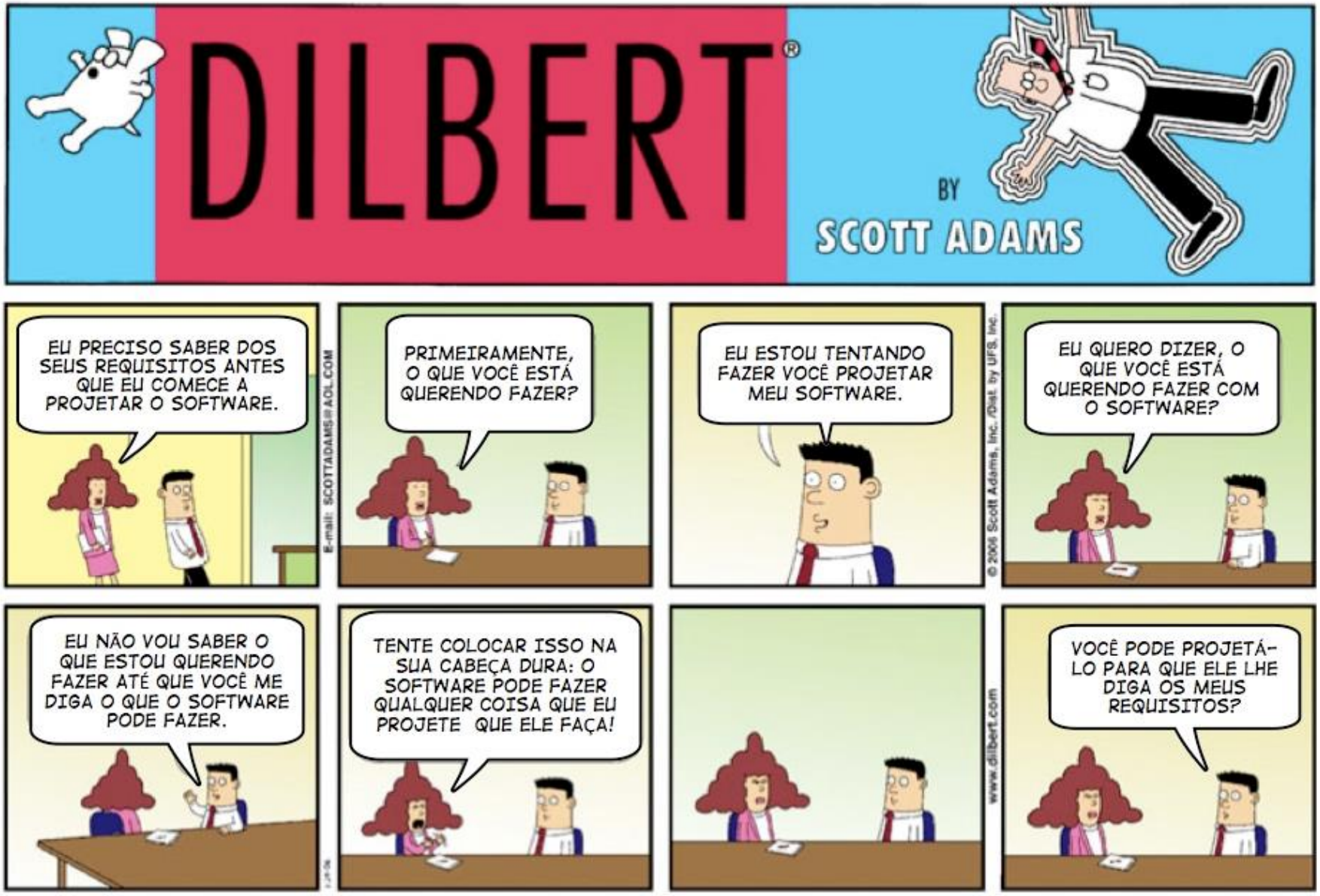
Ian Sommerville

Engenharia de Software



"Engenharia de *Software* é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais"

BAUER, 1960



Como o cliente explicou

Como o lider de projeto entendeu

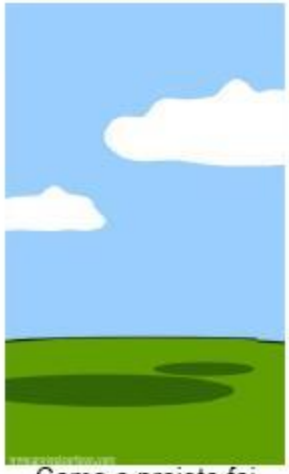
Como o analista planejou

Como o programador codificou

O que os beta testers receberam

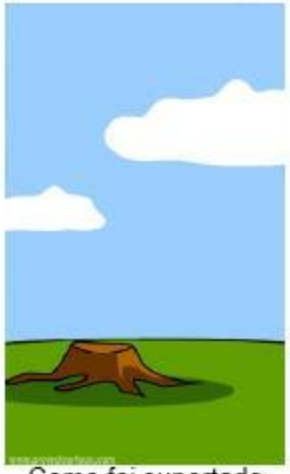
Como o consultor de negocios descreveu

Valor que o cliente pagou



Como o projeto foi documentado

O que a assistencia tecnica instalou



Como foi suportado

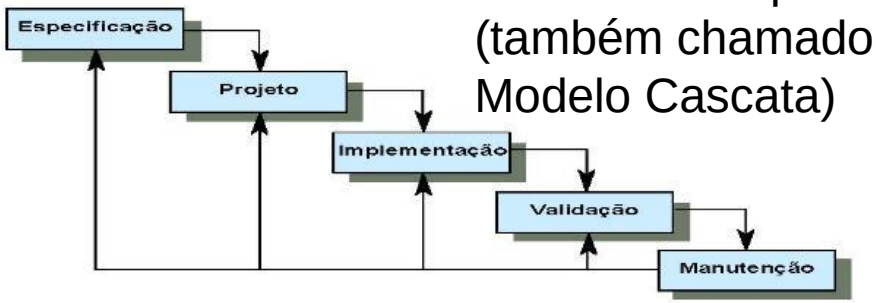
Quando foi entregue

O que o cliente realmente necessitava

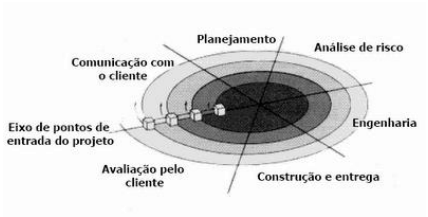
Modelos de Processo de Software

- 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

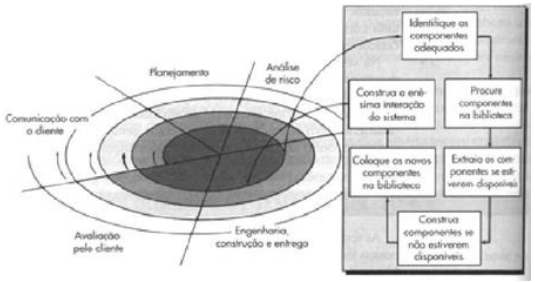
Modelos de Processo de Software



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

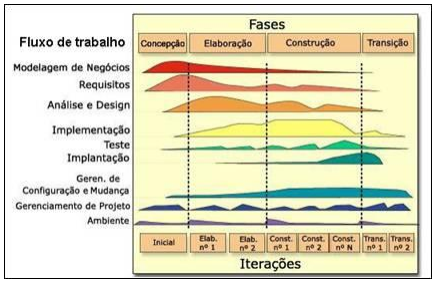


O Modelo Espiral



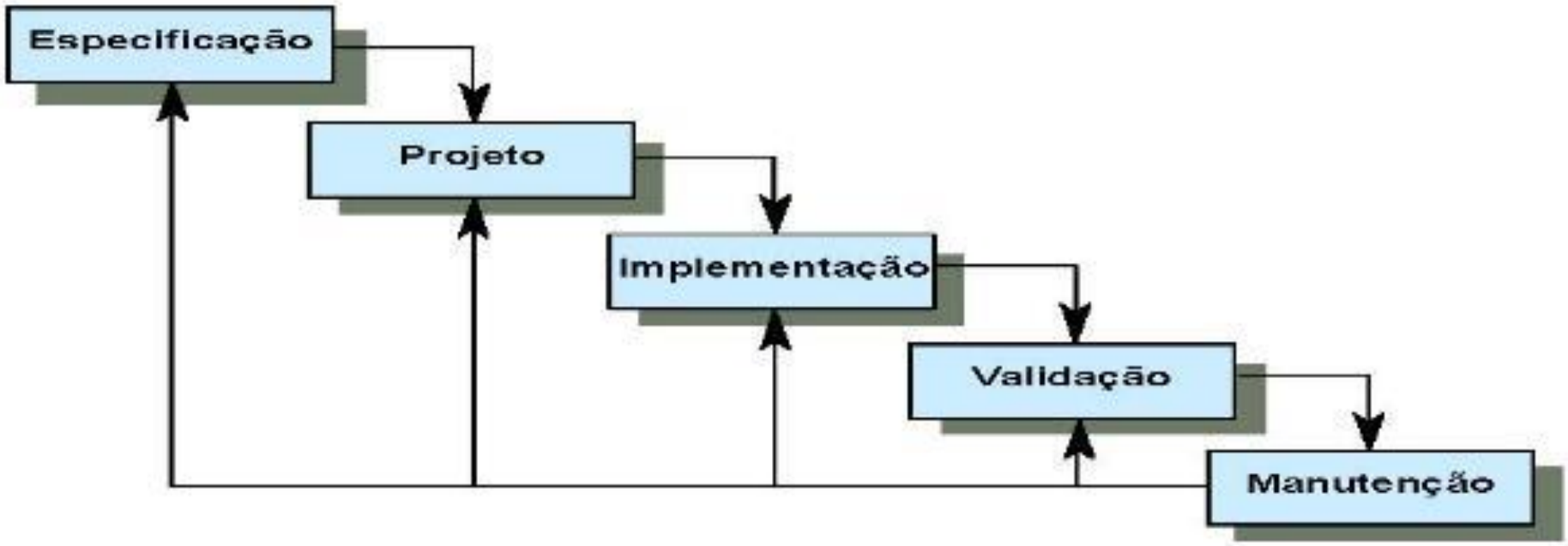
O Modelo Baseado em Componentes

O Paradigma de Prototipação



Processo Unificado

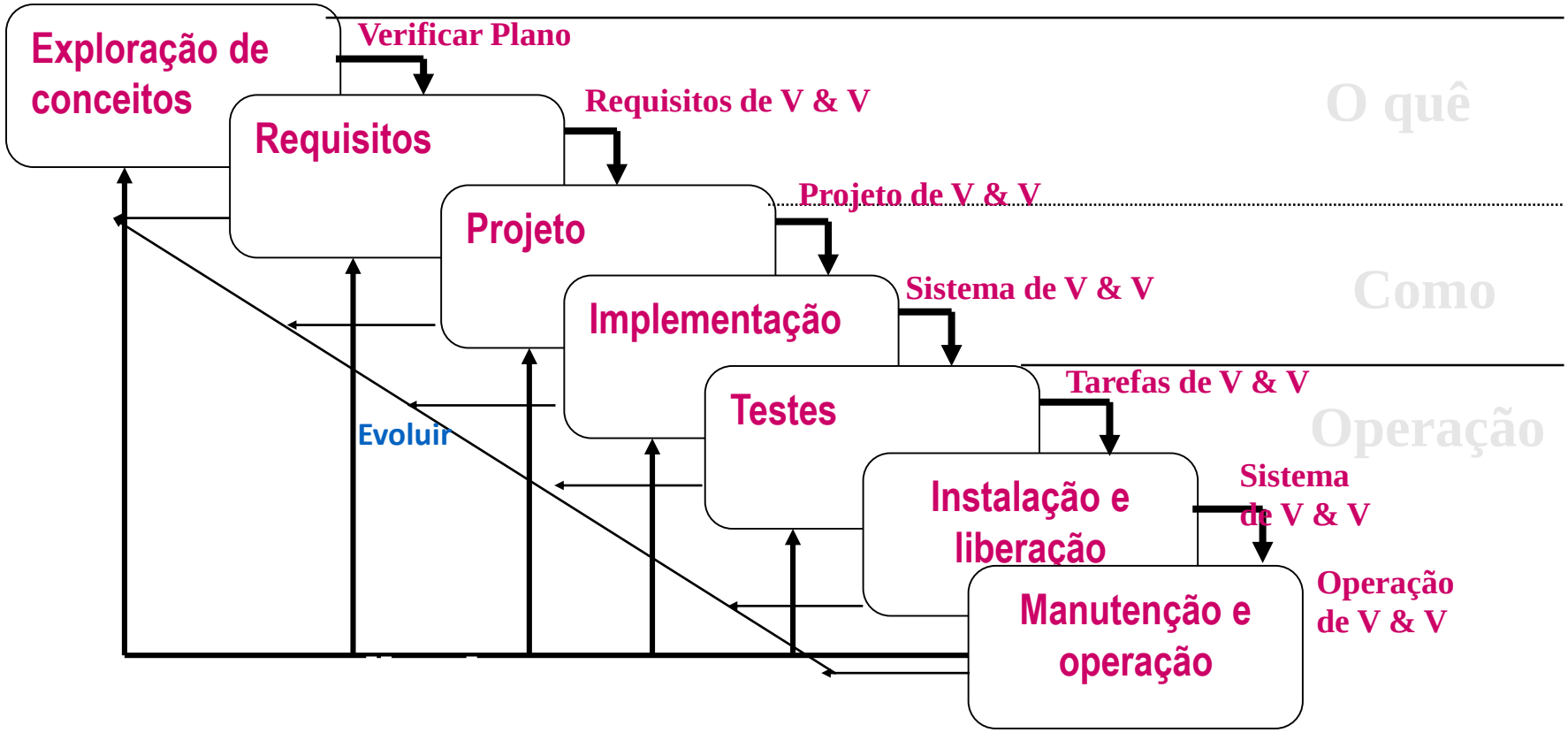
O Modelo Sequencial Linear (também chamado Ciclo de Vida Clássico ou Modelo Cascata)



O Modelo Cascata

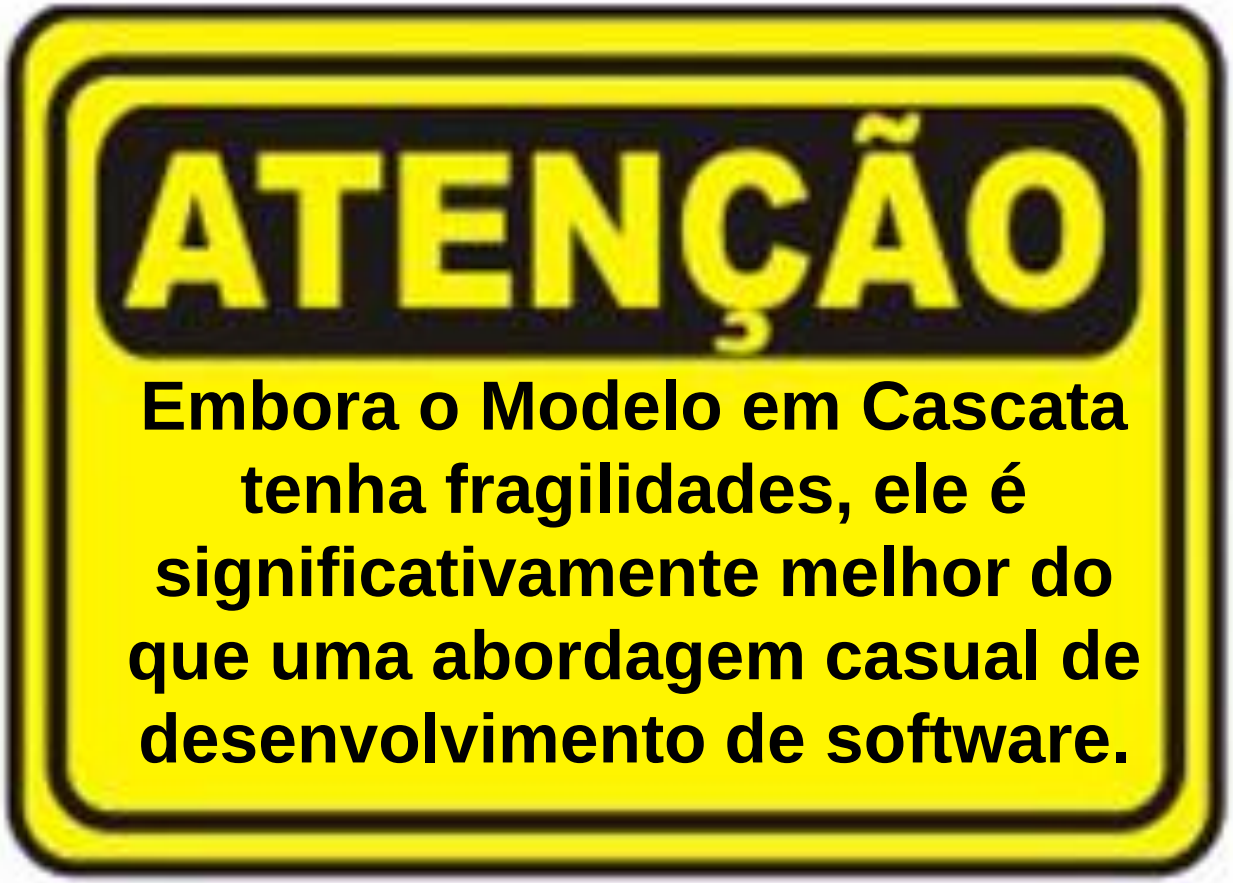
- modelo mais antigo e o mais amplamente usado da engenharia de software
- modelado em função do ciclo da engenharia convencional
- requer uma abordagem sistemática, seqüencial ao desenvolvimento de software
- o resultado de uma fase se constitui na entrada da outra

O Modelo em Cascata



Problemas com o Modelo em Cascata

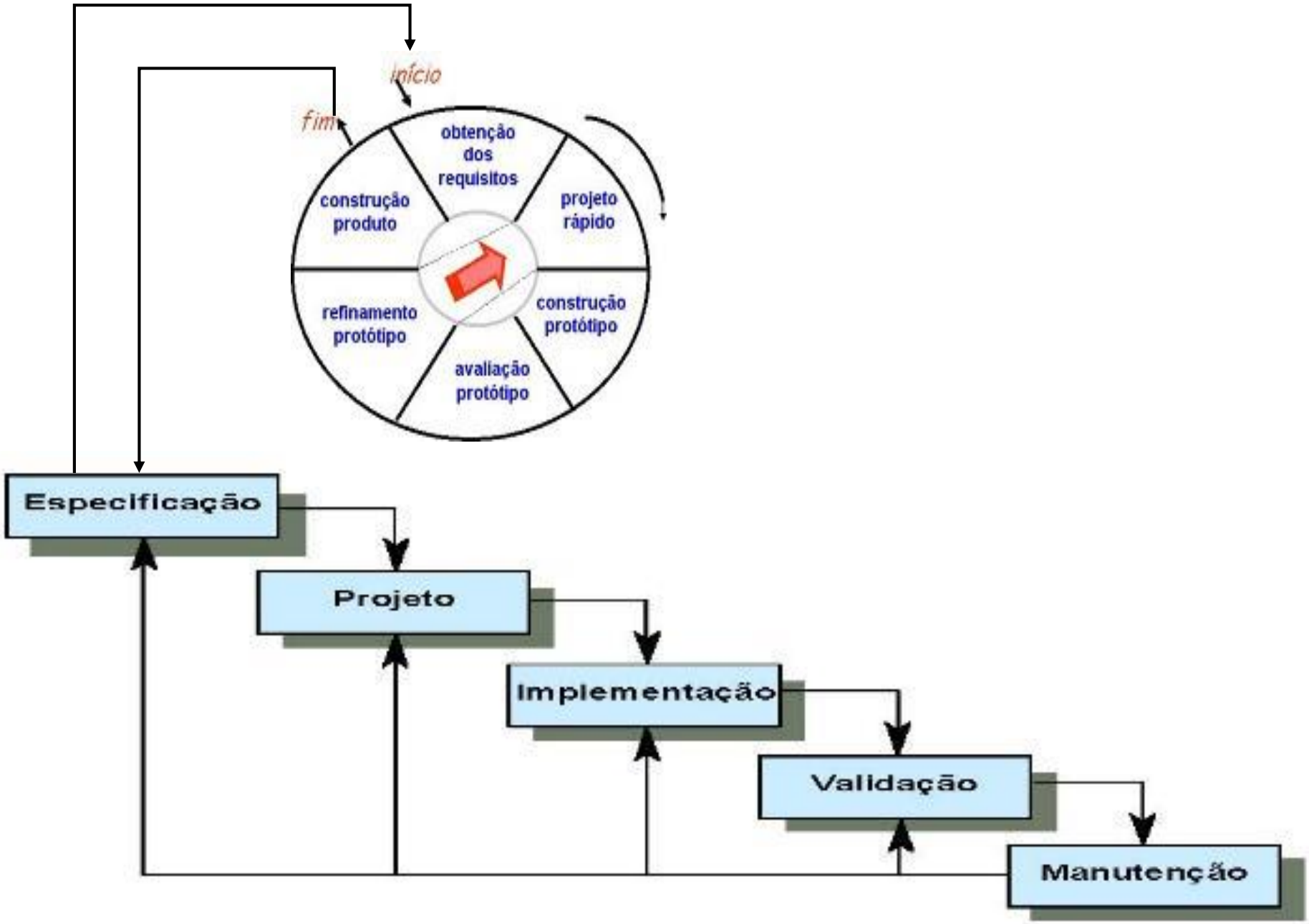
- Projetos reais raramente seguem o fluxo sequencial que o modelo propõe;
- Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural;
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa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


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;

Prototipação


início
fim
obtenção dos requisitos
projeto rápido
construção protótipo
avaliação protótipo
refinamento protótipo
construção produto
Especificação
Projeto
Implementação
Validação
Manutenção

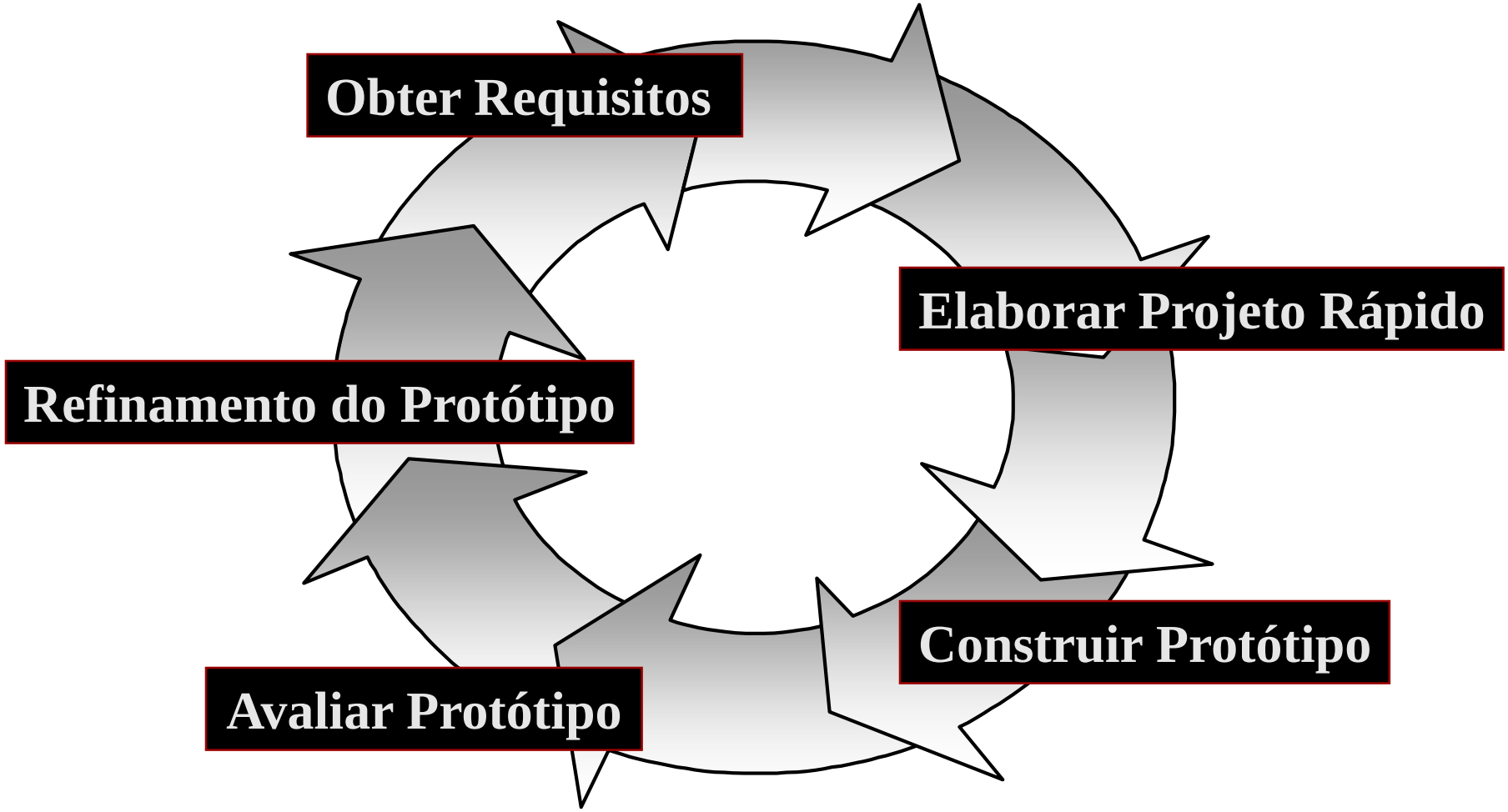
O Modelo de Prototipação

- o objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.

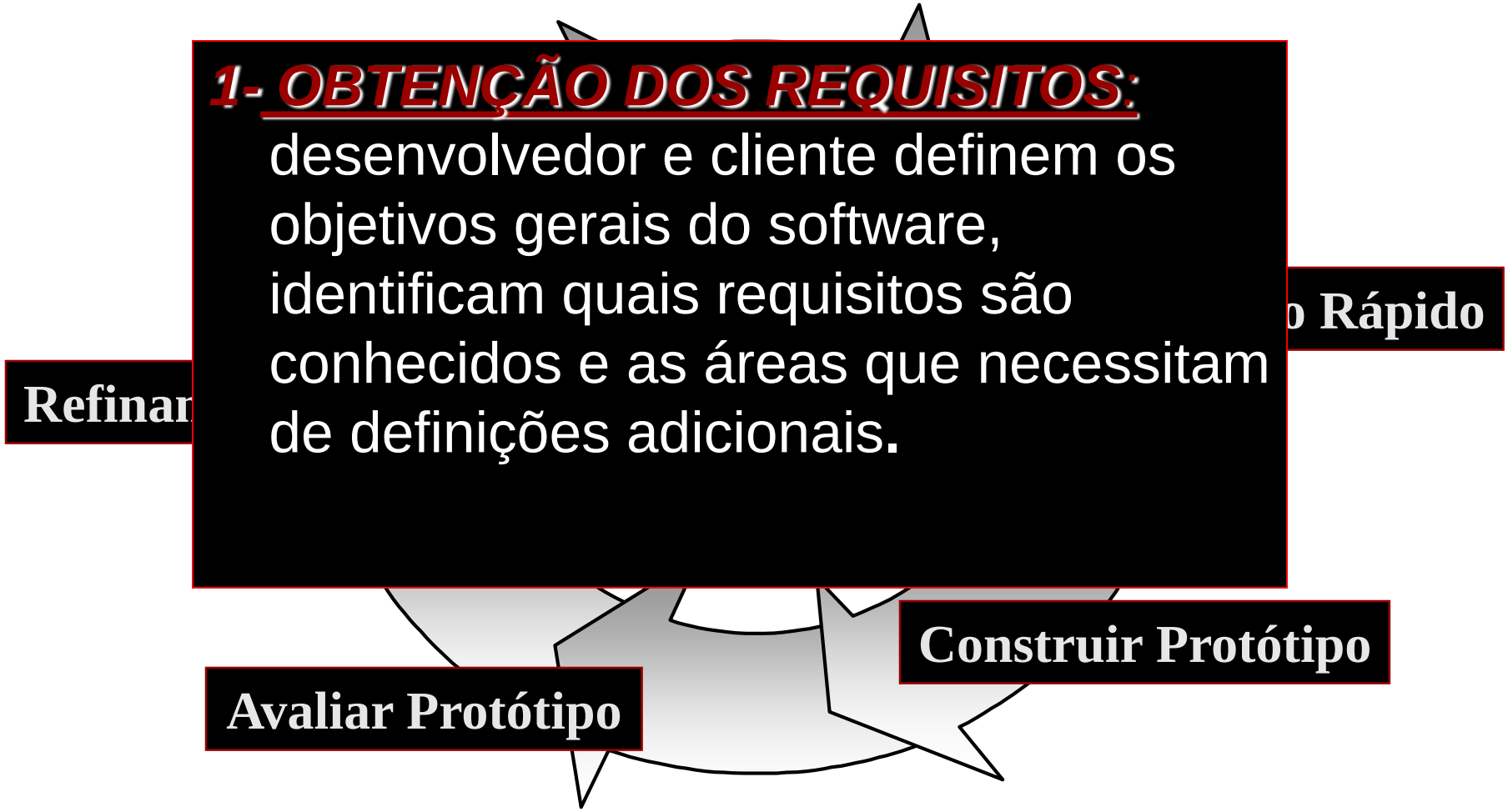
- possibilita que o desenvolvedor crie um modelo (protótipo)do software que deve ser construído

- apropriado para quando o cliente não definiu detalhadamente os requisitos.

O Paradigma de Prototipação para obtenção dos requisitos

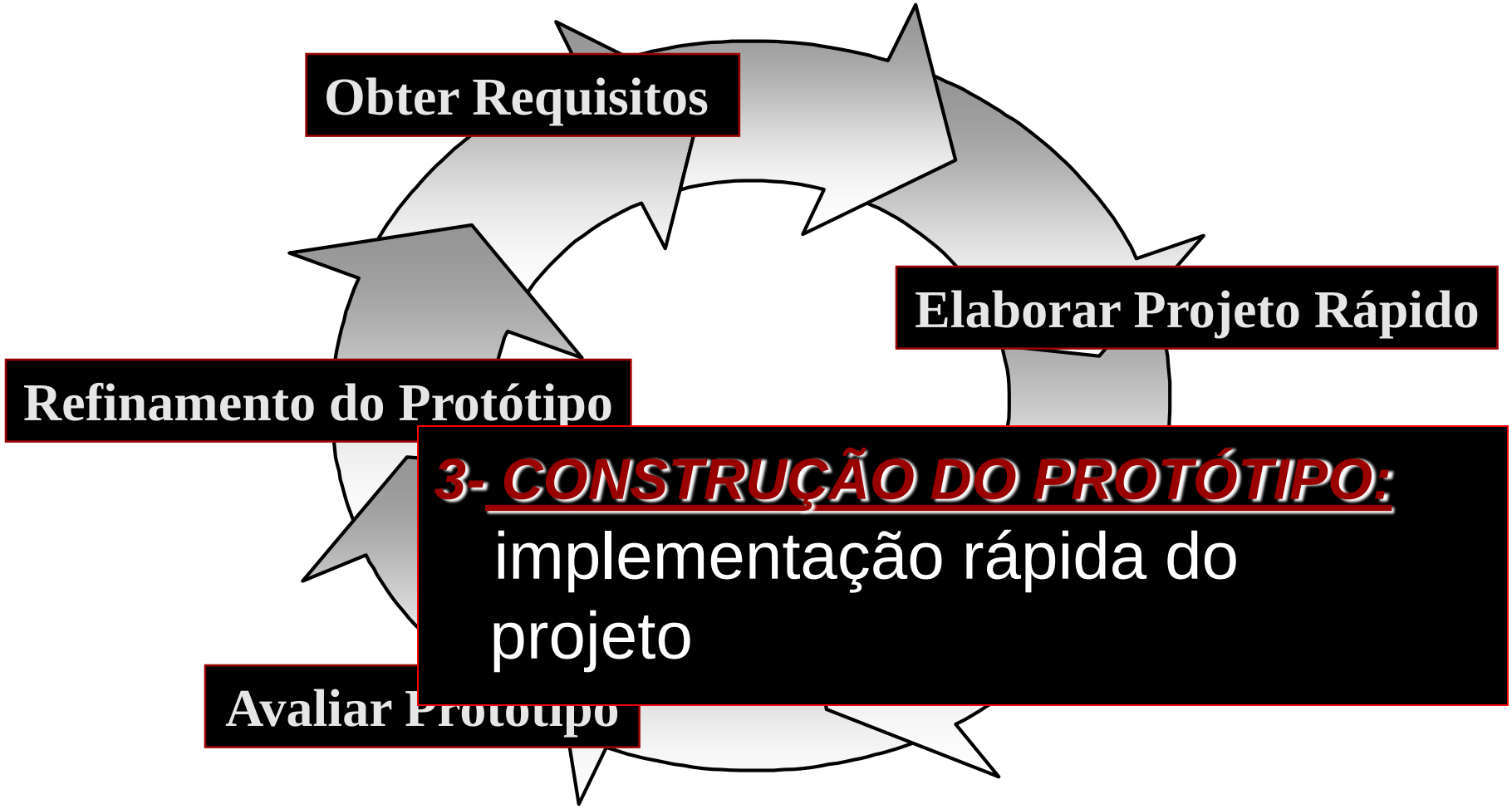


O Paradigma de Prototipação para obtenção dos requisitos

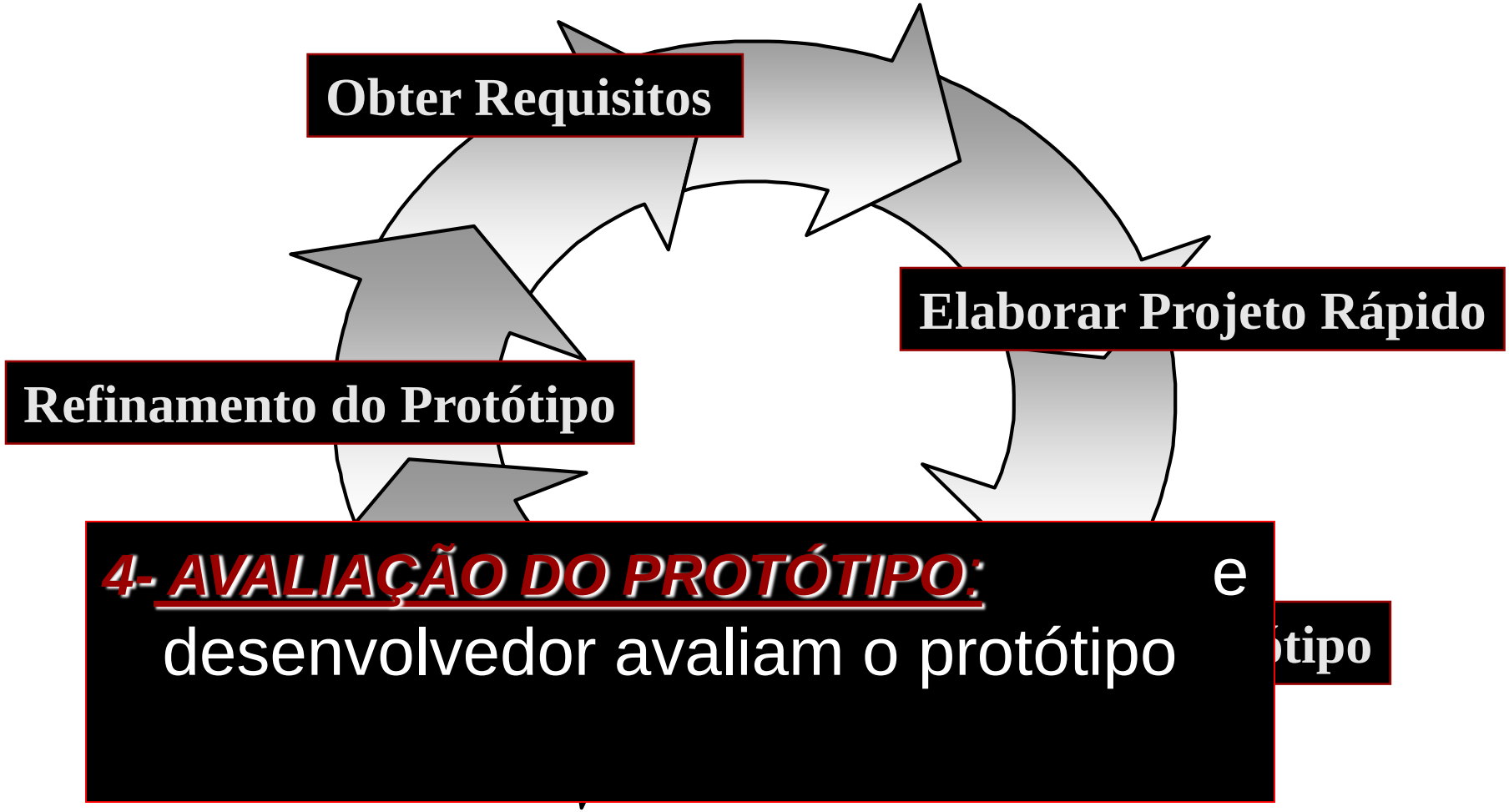


O Paradigma de Prototipação para obtenção dos requisitos

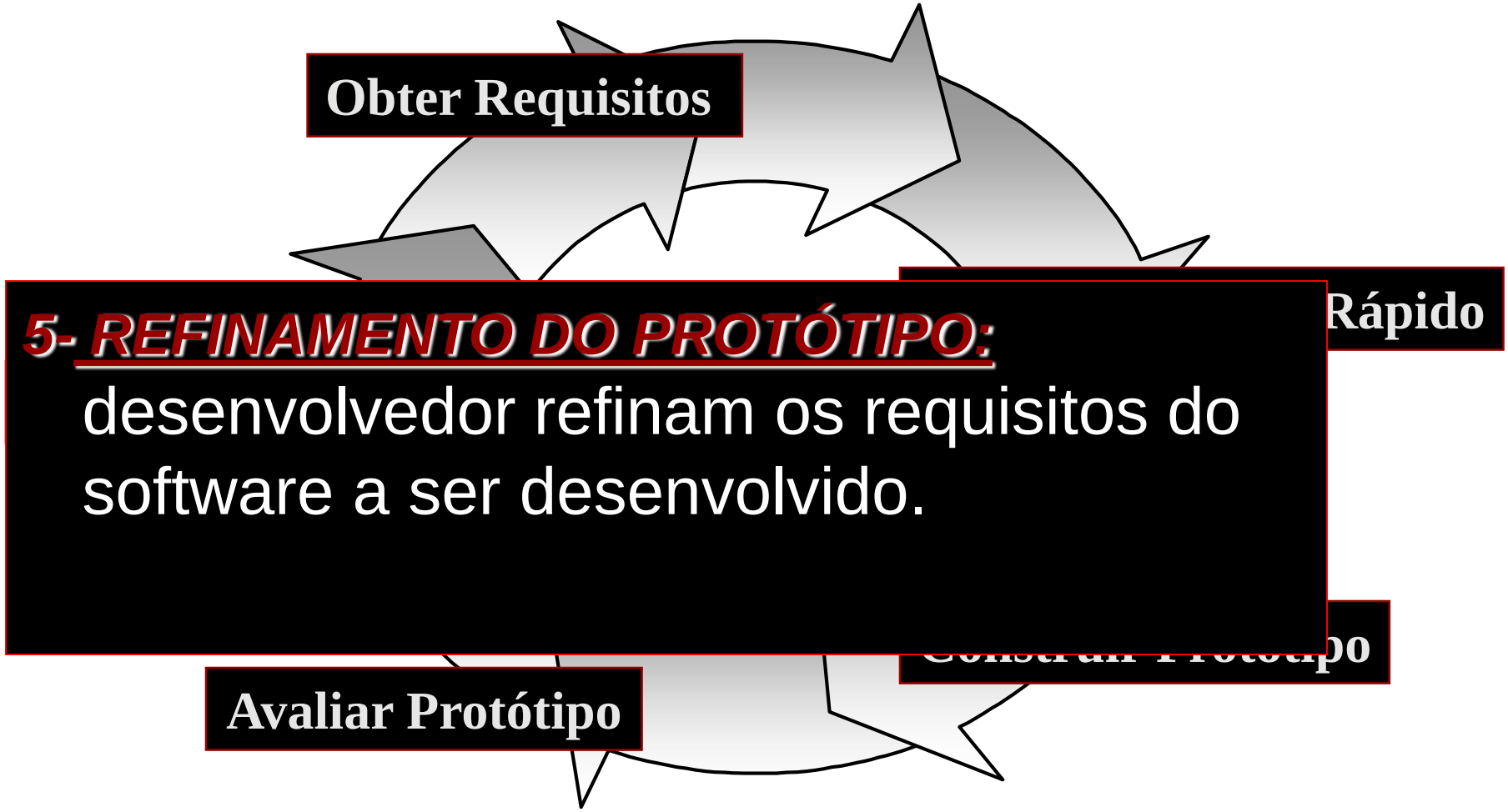
O Paradigma de Prototipação para obtenção dos requisitos



O Paradigma de Prototipação para obtenção dos requisitos



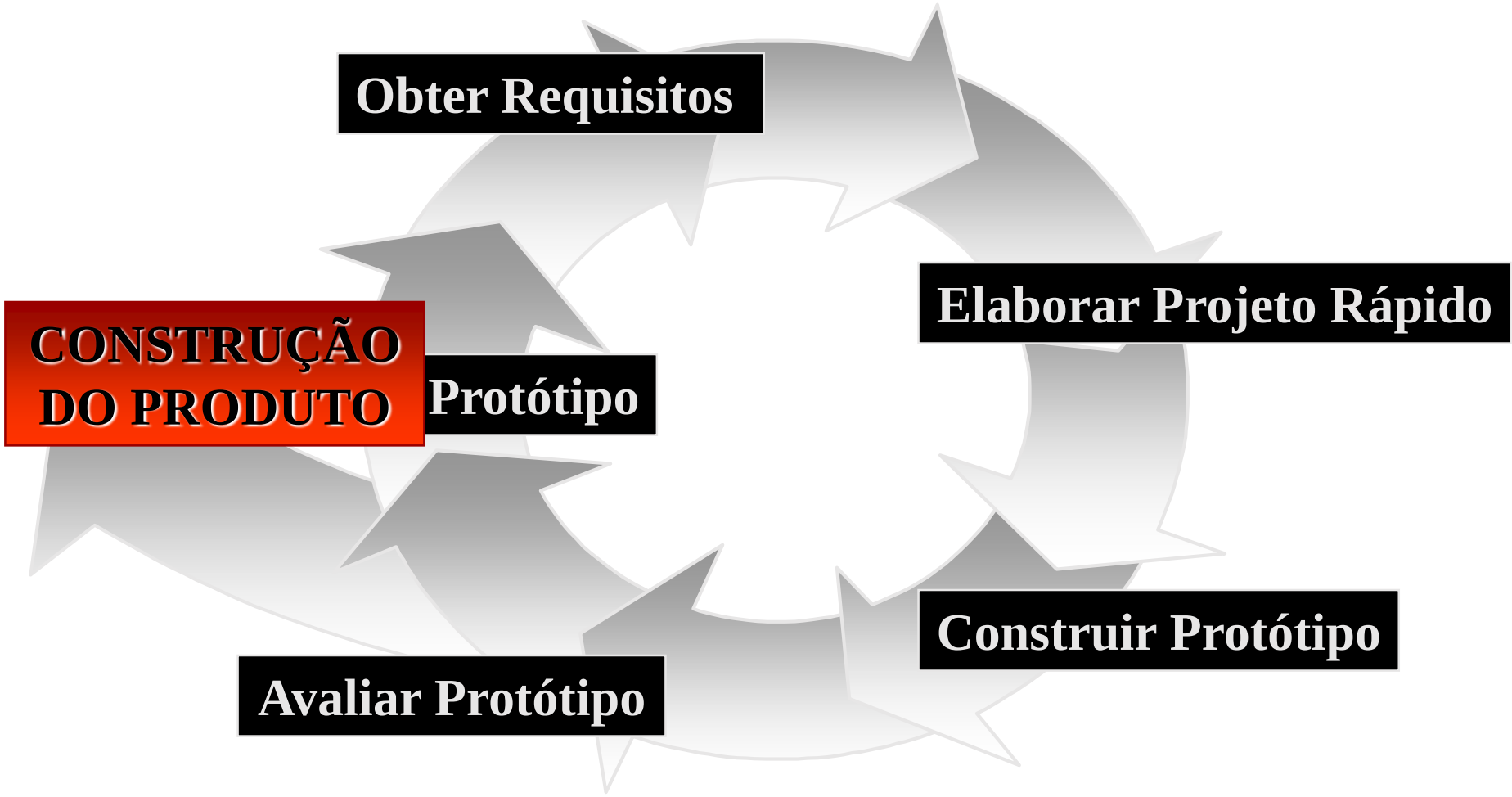
O Paradigma de Prototipação para obtenção dos requisitos



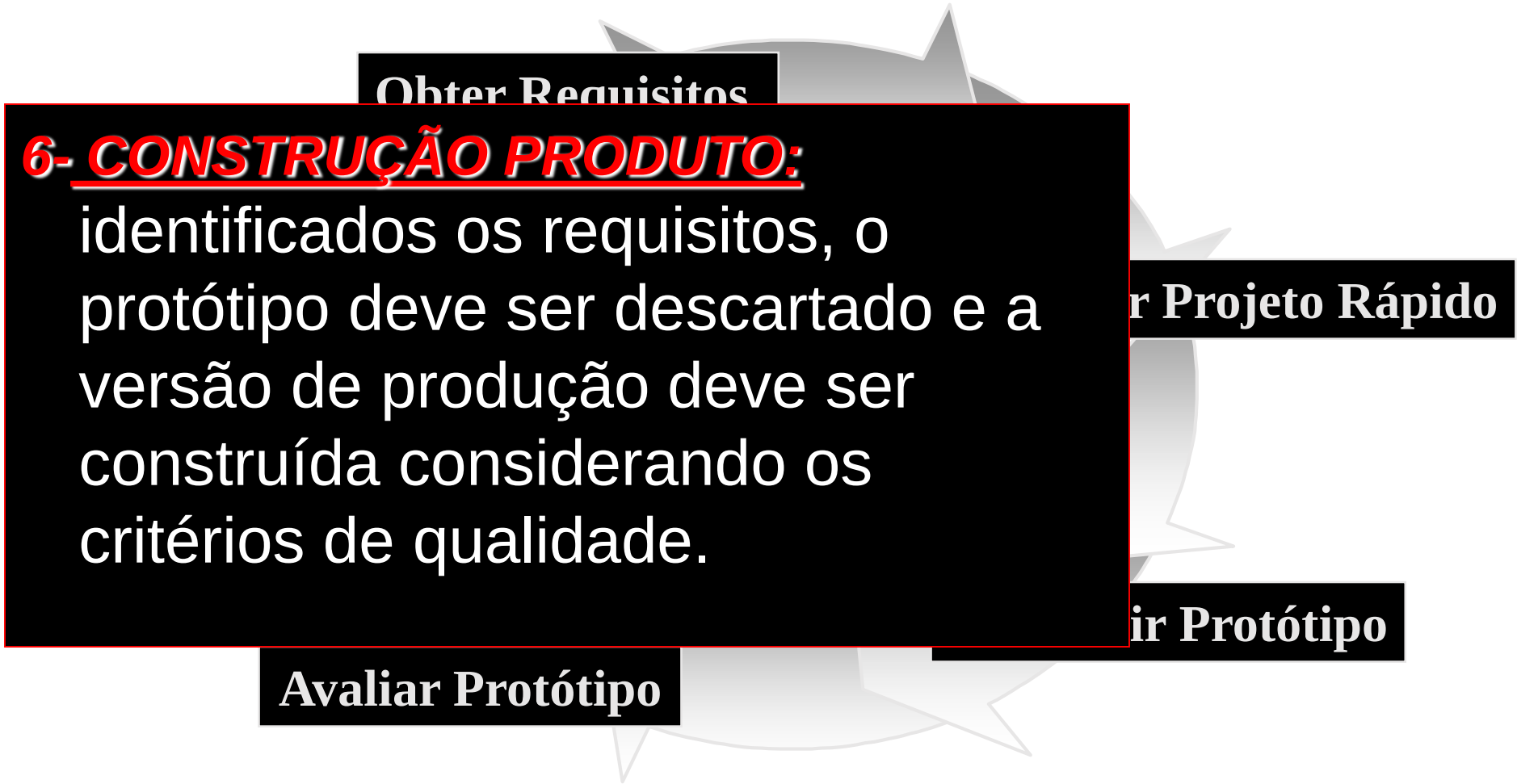
5- REFINAMENTO DO PROTÓTIPO:

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 mantenabilidade a longo prazo;

- desenvolvedor frequentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo.

Comentários sobre o Paradigma de Prototipação

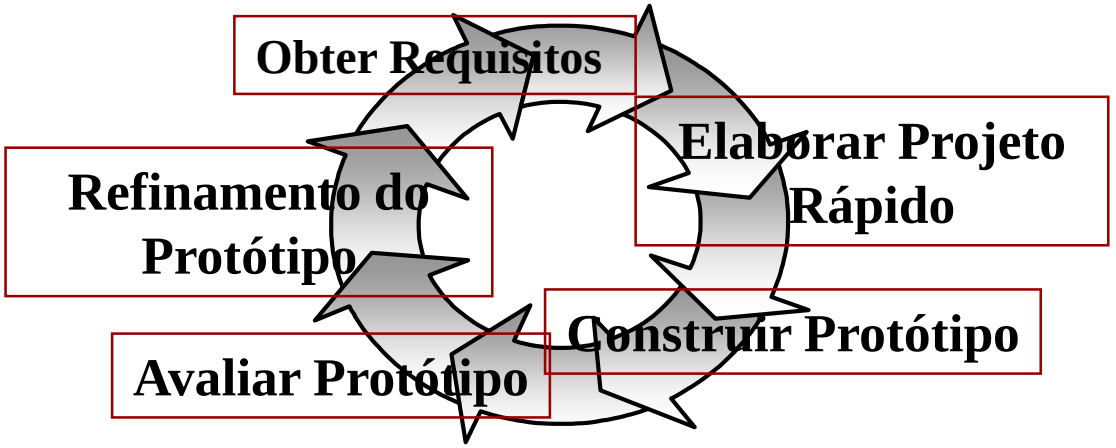
- ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- a chave é definir as regras do jogo logo no começo.
- o cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo para definir os requisitos

Ferramentas

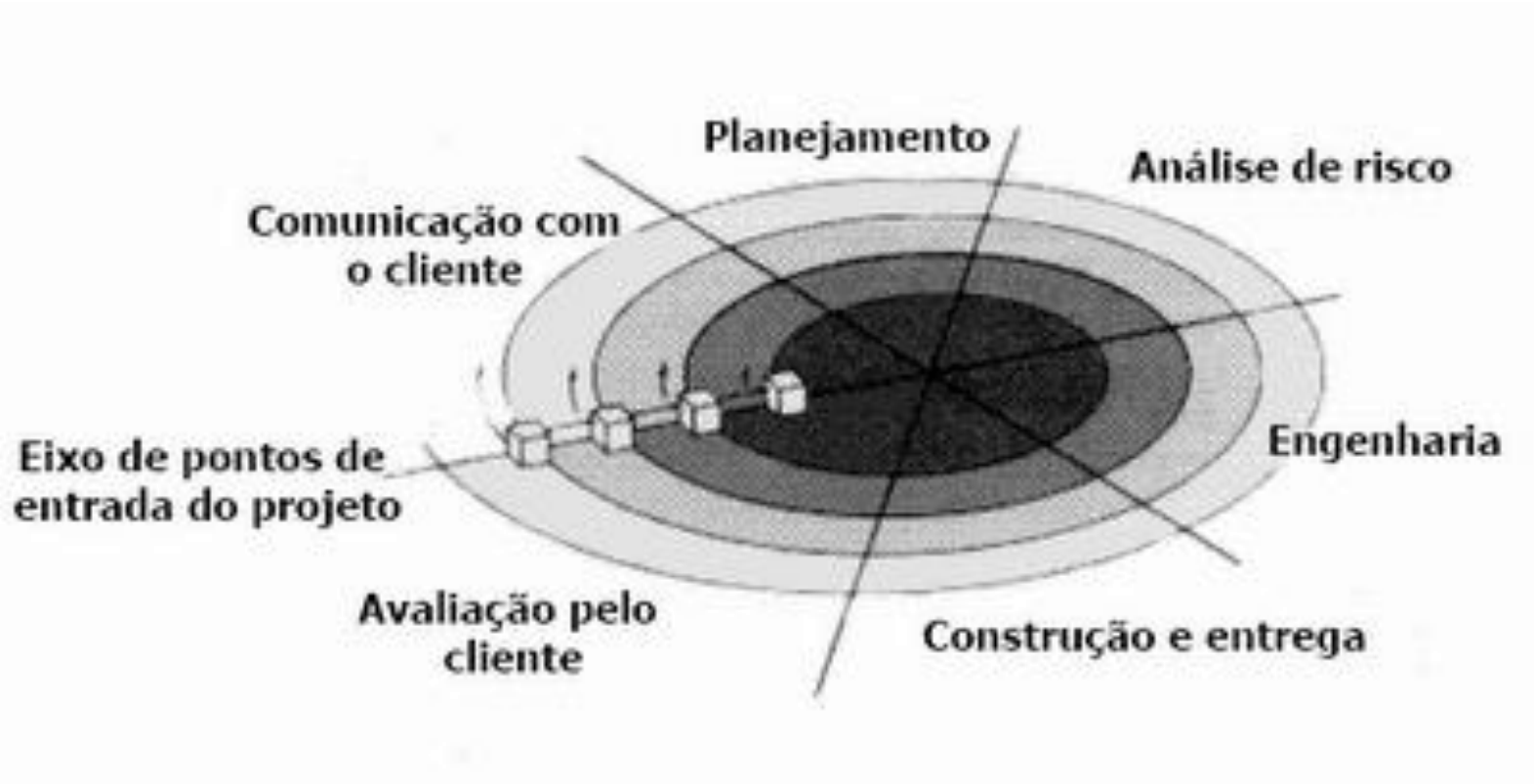
- https://marvelapp.com/

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

- 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

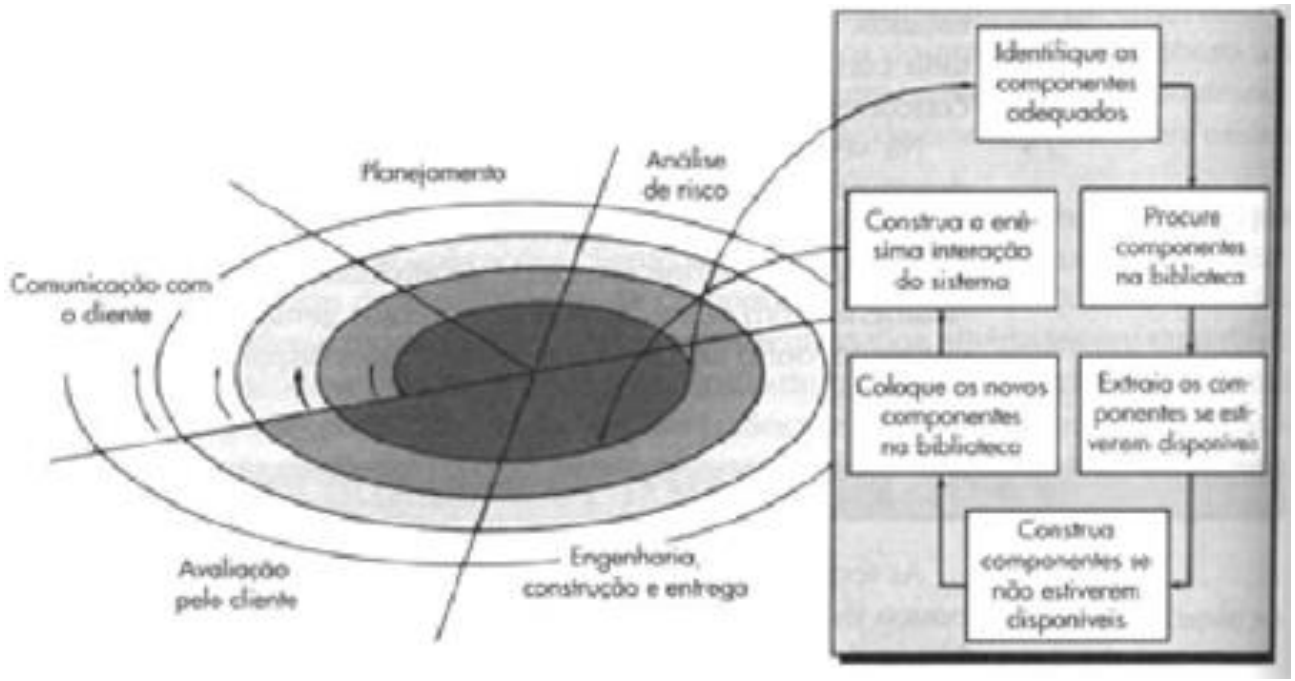
O Modelo Espiral

- 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

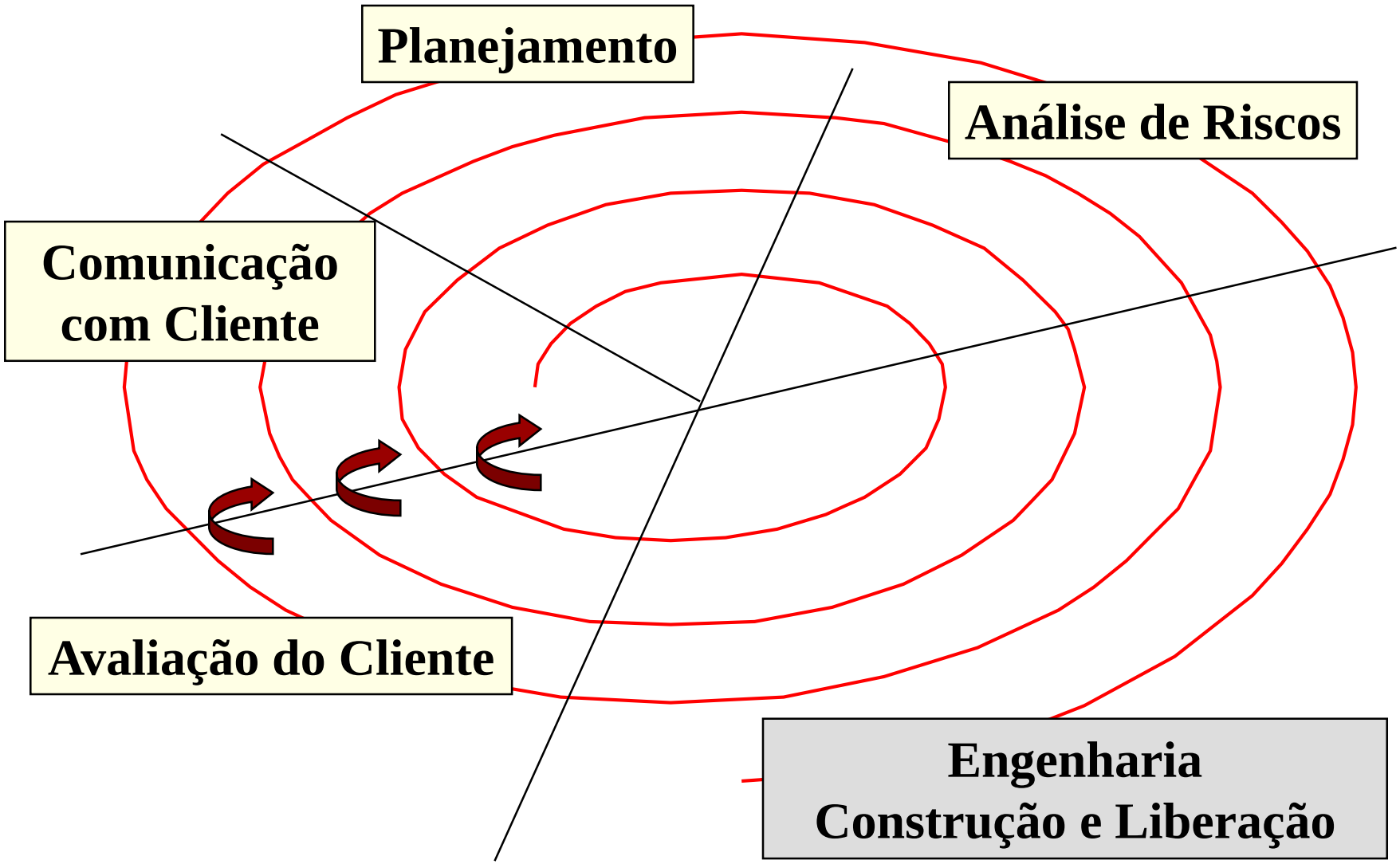
O Modelo BASEADO EM Componentes



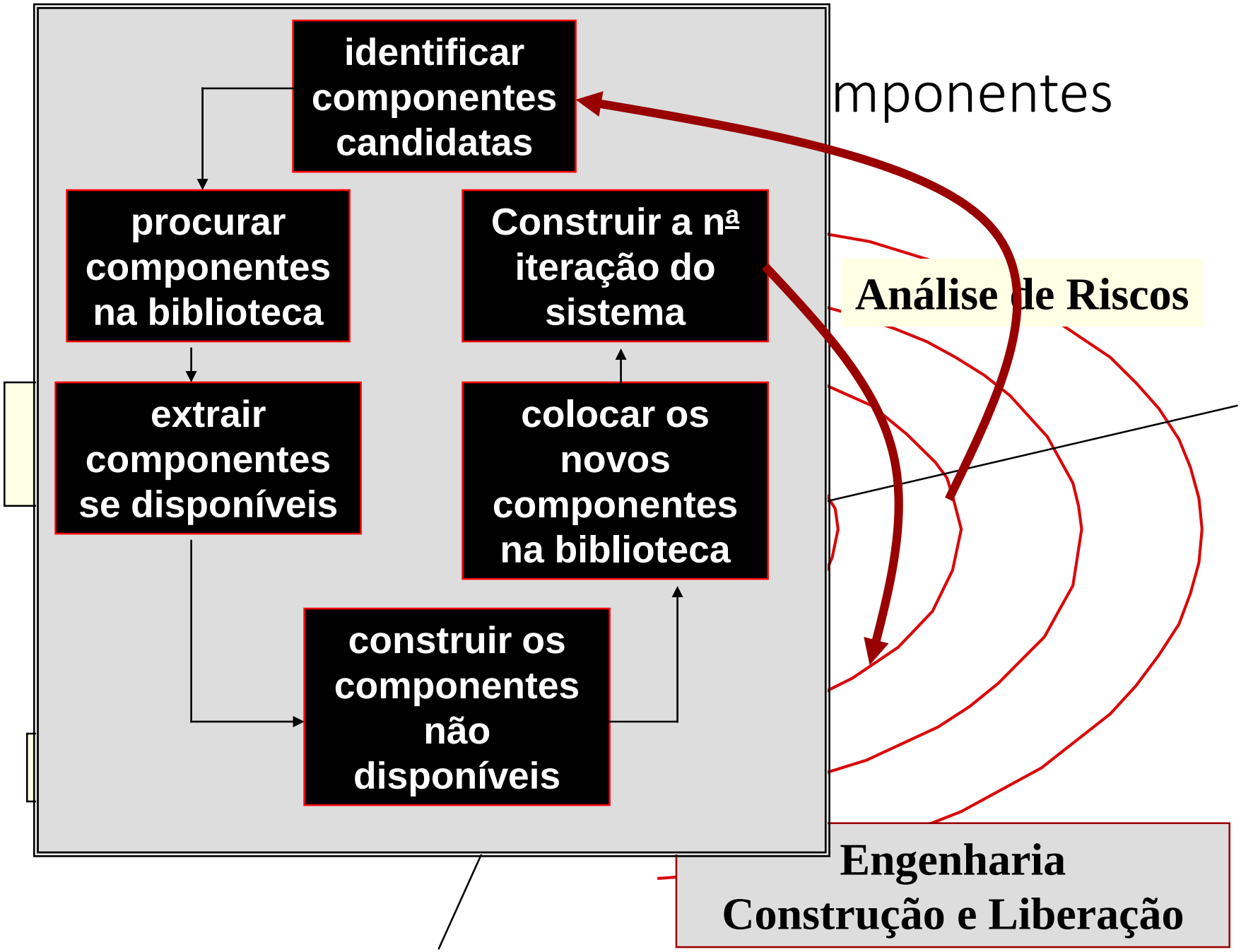
O Modelo Baseado em Componentes

- 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



mponentes


identificar componentes candidatas
procurar componentes na biblioteca
extrair componentes se disponíveis
construir os componentes não disponíveis
colocar os novos componentes na biblioteca
Construir a nª iteração do sistema
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?

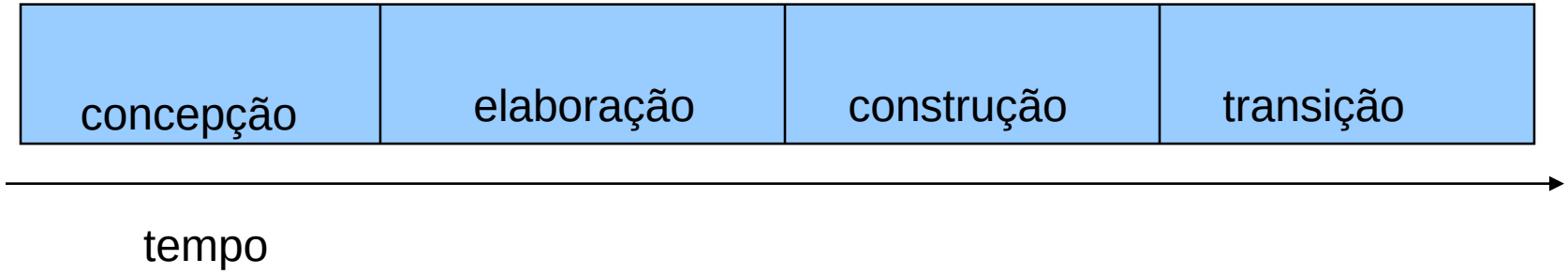
- Conjunto de atividades
  - bem definidas
  - com responsáveis
  - com artefatos de entrada e saída
  - com dependências entre as mesmas e ordem de execução
  - com modelo de ciclo de vida
  - descrição sistemática de como devem ser realizadas
  - guias (de ferramentas ou não), *templates*
  - utilizando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

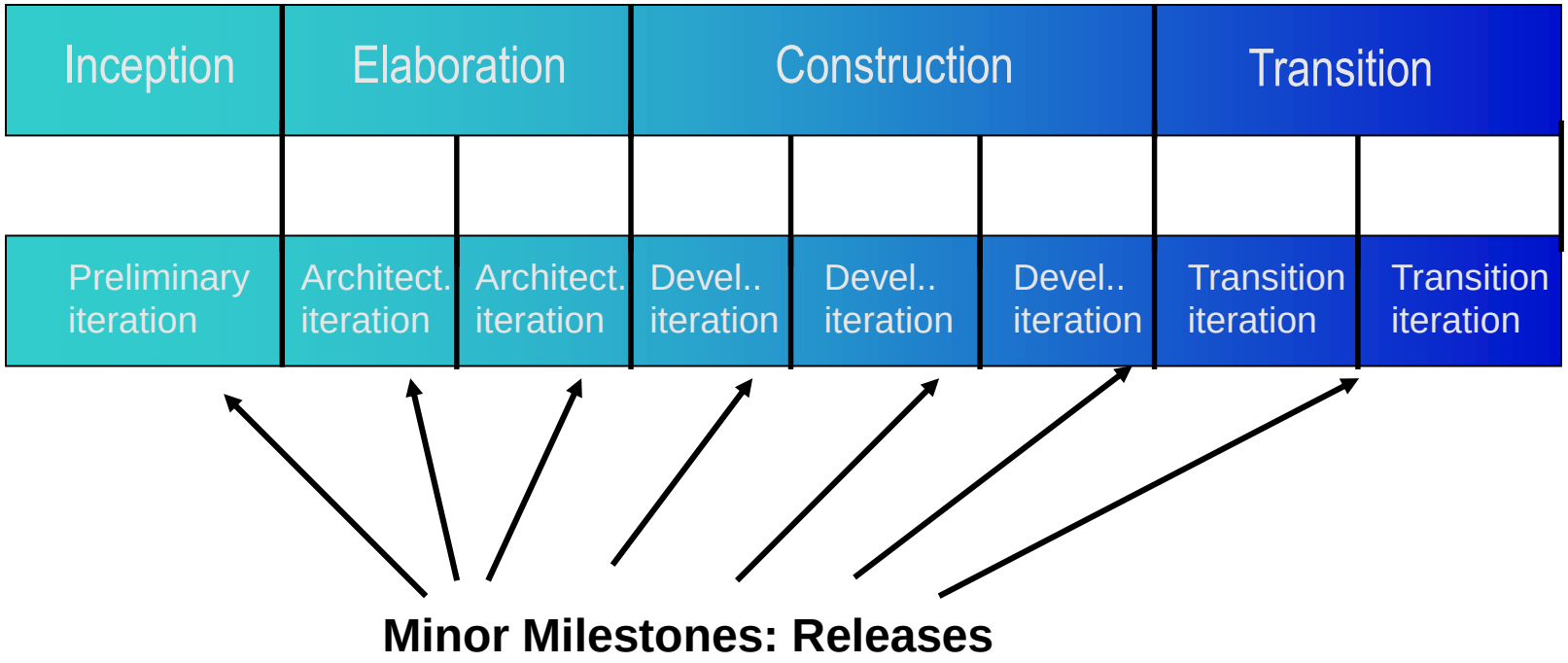
- O ciclo de vida de um sistema consiste de quatro fases:



- 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

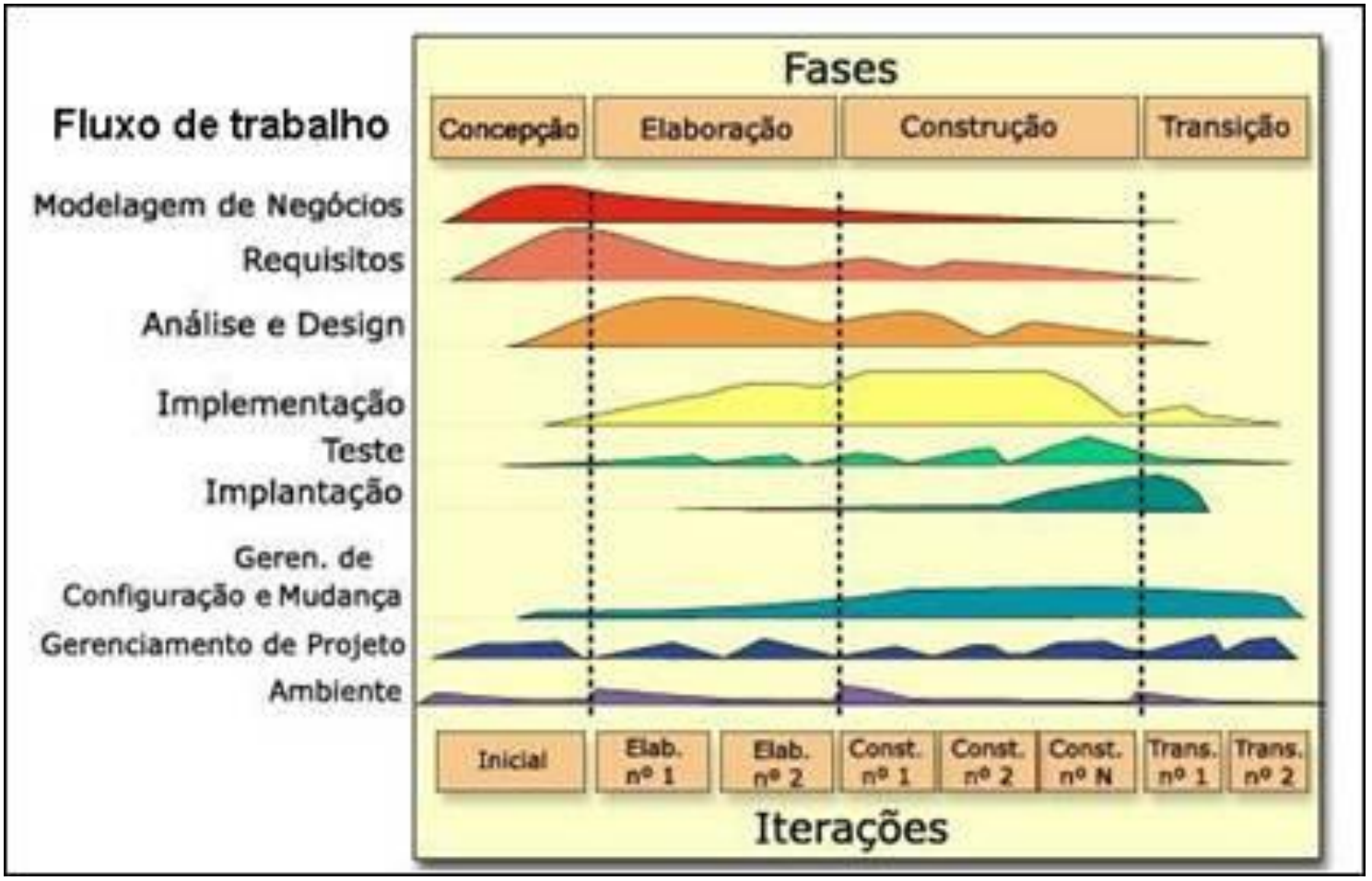
- Cada fase é dividida em iterações:



O RUP é iterativo e incremental

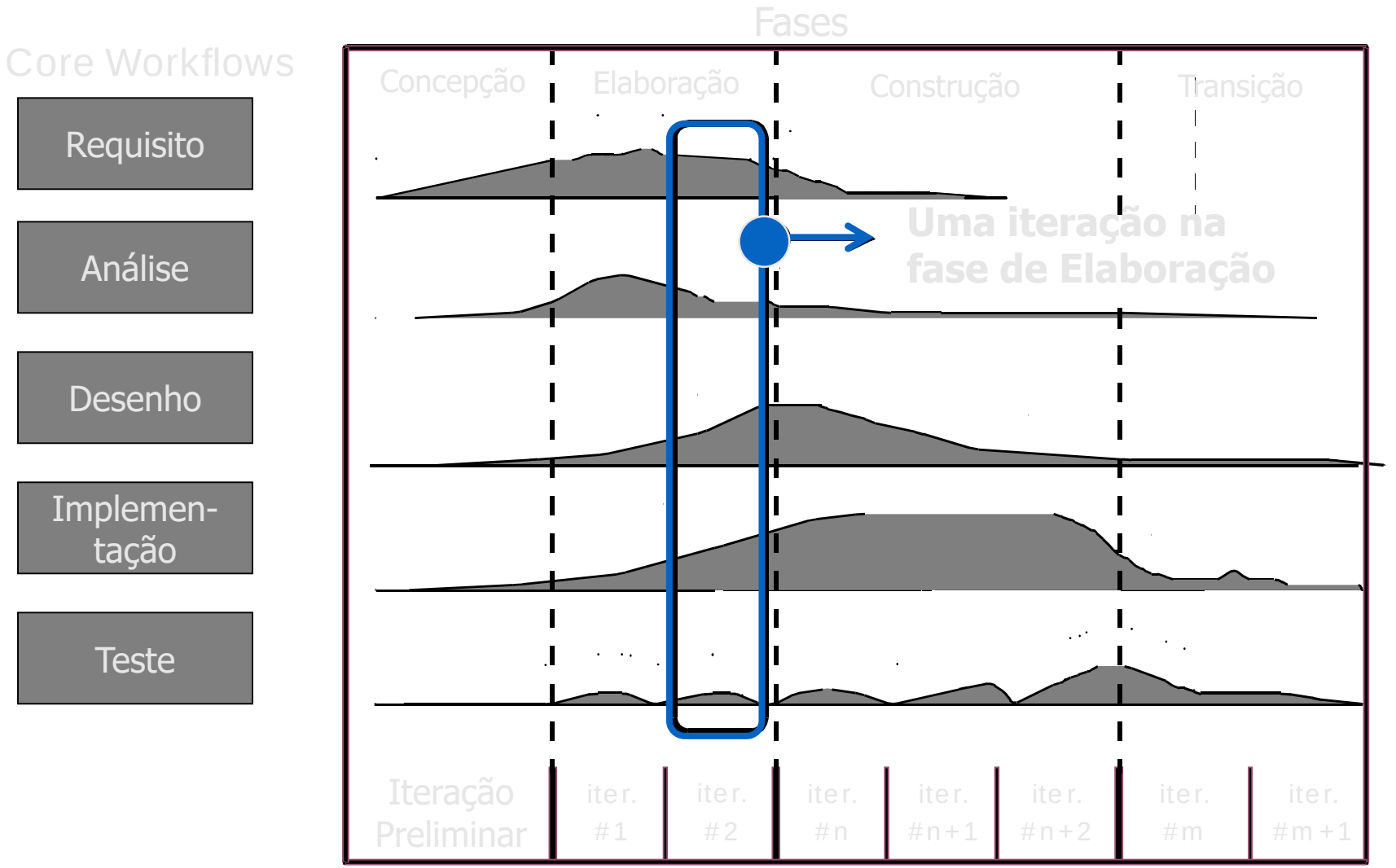
- Cada iteração
  - é planejada
  - realiza uma seqüência de atividades (de elicitação de requisitos, análise e projeto, implementação, etc.) distintas
  - geralmente resulta em uma versão executável do sistema
  - é avaliada segundo critérios de sucesso previamente definidos

O RUP é iterativo e incremental



Iteração e Workflow

Passos dentro de uma iteração



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

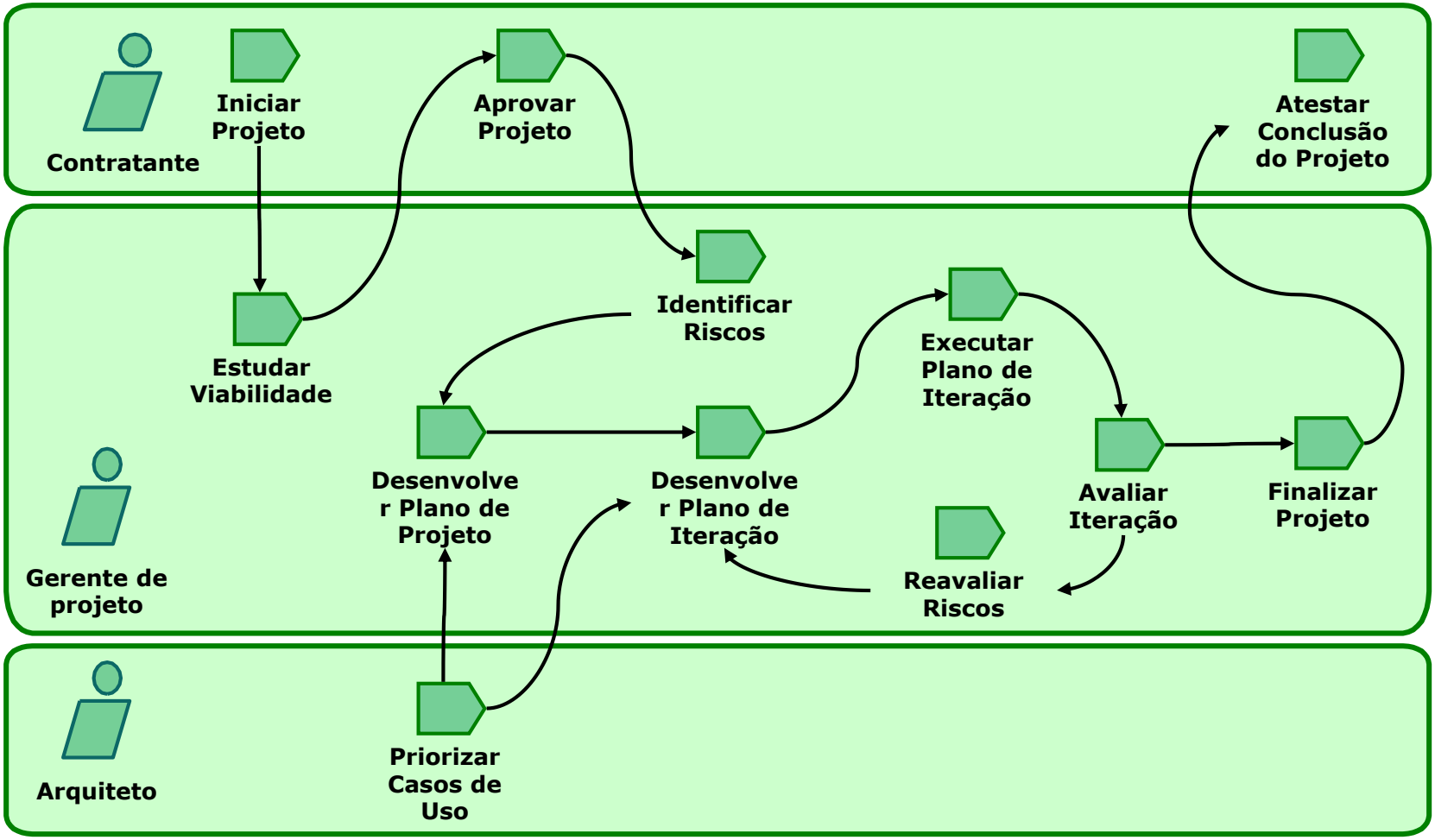
O RUP é baseado na arquitetura do sistema

- 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.funpar.ufpr.br:8080/rup/process/ovu_proc.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

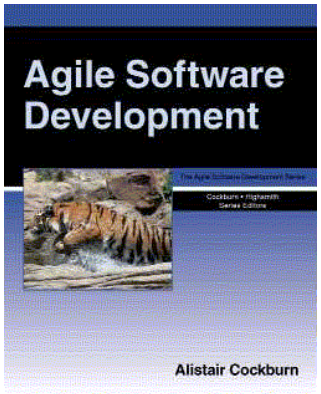
- Objetivo: satisfazer o cliente entregando, rapidamente e com freqüência, sistemas com algum valor.
  - Entregar versões funcionais em prazos curtos.
  - Estar preparado para requisitos mutantes.
  - Pessoal de negócios e desenvolvedores juntos.
  - Troca de informações através de conversas diretas.

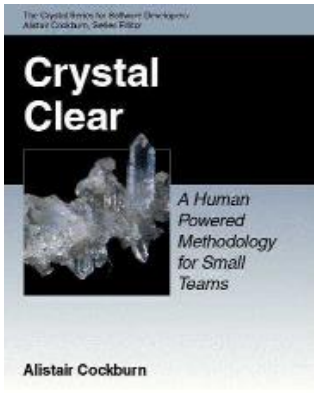
Principais Métodos Ágeis

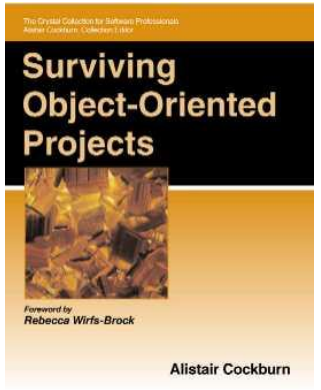
- Crystal (uma família de métodos)
- Scrum
- Programação eXtrema (XP)
- Adaptive Software Development
- Feature Driven Development
- etc.

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.

Como escolher um Modelo para o Meu Processo de Desenvolvimento de Software

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**

Grupos e PodCast, Vídeos e textos

- O Manifesto Ágil (4 alunos):
  - http://www.manifestoagil.com.br/ (texto)
  - https://hipsters.tech/agilidade-hipsters-05/
  - http://www.agilcoop.org.br/files/Agilcast01-intro.mp3
- Programação Extrema (4 alunos):
  - http://www.fernandocosta.com.br/2007/10/08/extreme-programa-xp-programacao-extrema/ (texto)
  - http://www.agilcoop.org.br/files/Agilcast02-praticasXP.mp3
  - https://www.youtube.com/watch?v=UipebTN25Hc
- SCRUM (4 alunos):
  - http://gc.blog.br/2008/05/27/como-estamos-indo-com-a-adocao-de-scrum-na-globocom/ (texto)
  - https://hipsters.tech/scrum-do-zero-ao-sprint-hipsters-54/
  - https://www.youtube.com/watch?v=vg1S1WYZa6o

Grupos e PodCast, Vídeos e textos

- Métodos ágeis e documentação de software / Product Owner(4 alunos)
  - https://www.youtube.com/watch?v=3Smbhnmue7Y
  - https://www.youtube.com/watch?v=7lhnYbmovb4
  - https://www.youtube.com/watch?v=72ilcGqfvjU
- Kanban (4 alunos):
  - https://www.culturaagil.com.br/kanban-do-inicio-ao-fim/ (texto)
  - https://www.youtube.com/watch?v=LJOiFRsp0Z8
- Padrões para Introduzir Novas Ideias (4 alunos):
  - https://www.youtube.com/watch?v=8Kjv6gI8kJQ (1/3)
  - https://www.youtube.com/watch?v=wXBiHK0763Y (2/3)
  - https://www.youtube.com/watch?v=MsC5bKlpEno (3/3)
  - http://www.agilcoop.org.br/files/Agilcast11-Padroes%20para%20introduzir%20novas%20ideias.mp3

Para Entregar na Próxima aula (07/12/2017)

- Cada grupo deverá fazer uma apresentação em Power Point sobre os temas que foram passados pelos links.
- A apresentação deverá ter a duração de no máximo 10 minutos.

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 analise 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 analise da segunda, quais a as atividades deverão ser realizadas para o desenvolvimento do sistema da Farmácia “Pague Pouco”.

Para a Prova, veja os capítulos seguintes:

- Livro de Engenharia de Software, autor: Ian Sommerville:
  - 8º edição – Capítulo 4.1, 4.2, 4.3 e 4.4

  ou
  - 9ª edição – Capítulo 2.1, 2.2 e 2.4
- Livro de Engenharia de Software, autor: Roger Pressman:
  - 6ª edição – Capítulo 3.1, 3.2, 3.4 e 3.6