PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

Processo de Desenvolvimento de Software

MODELOS DE PROCESSO & ITERAÇÃO DE PROCESSO

Prof. Carlos Eduardo de B. Paes
Departamento de Computação
Pontifícia Universidade Católica de São Paulo
carlosp@pucsp.br

Questões que determinam a escolha de um modelo

Quão bem os analistas e o cliente podem conhecer os requisitos do sistema?

Quão bem é compreendida a arquitetura do sistema?

Qual o grau de confiabilidade necessário em relação ao cronograma?

Quanto planejamento é efetivamente necessário?

Qual o grau de risco que este projeto apresenta?

Existe alguma restrição de cronograma?

Será necessário entregar partes do sistema funcionando antes de terminar o projeto todo?

Qual o grau de treinamento e adaptação necessários para a equipe poder utilizar o ciclo de vida que parece mais adequado ao projeto?

Será desenvolvido um único sistema ou uma família de sistemas semelhantes?

Qual o tamanho do projeto?

Modelo de Processo

Conhecer os modelos de processo de software é fundamental para uma correta aplicação de um determinado processo

Os modelos são adequados em função do tipo de aplicação a ser desenvolvida

Modelos de processo podem ser considerados como a “Arquitetura” do processo

Na realidade a maioria dos grandes sistemas são desenvolvidos usando um processo que incorpora elementos dos principais modelos

Representação abstrata de um processo de software

Modelos de Processo

Code and Fix

Cascata (Watherfall)

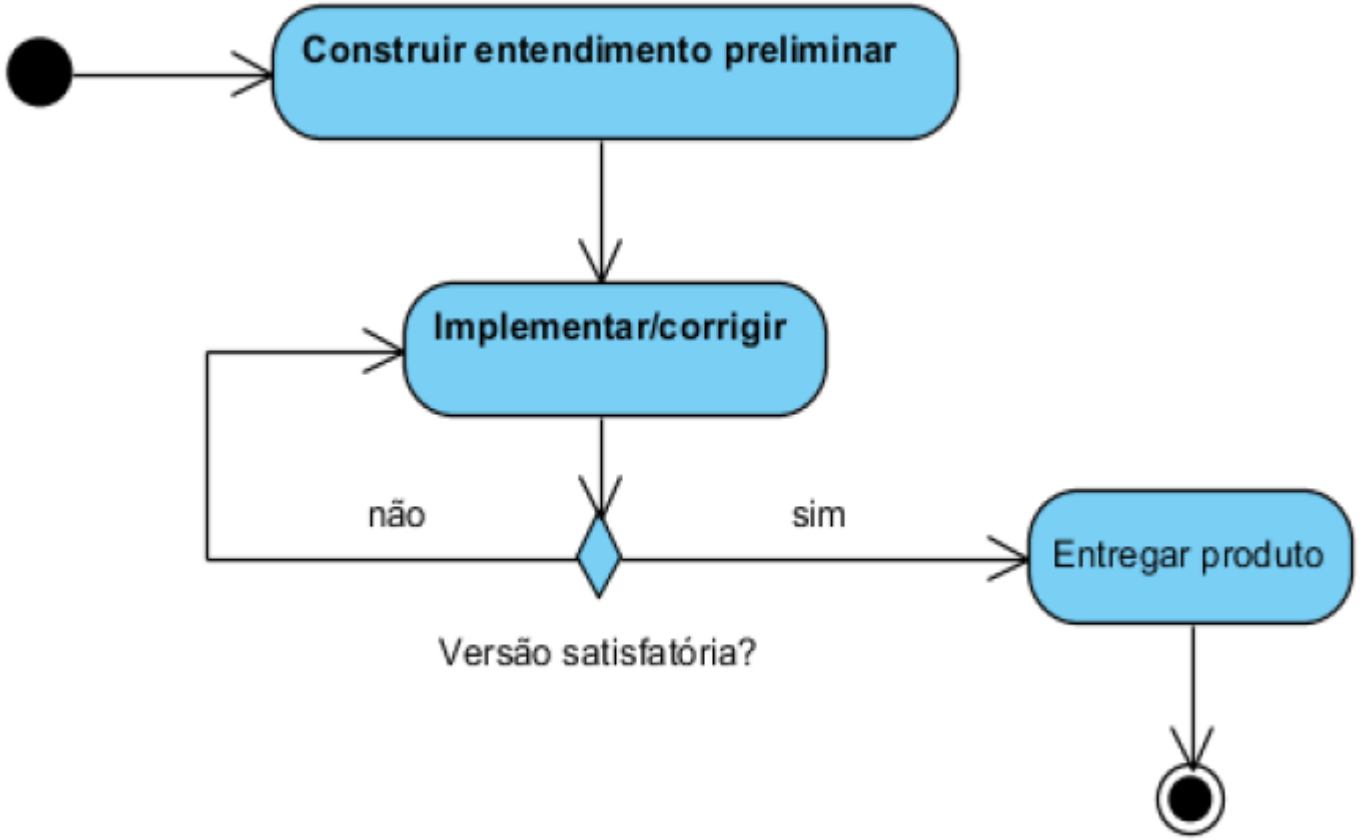
Desenvolvimento Evolucionário

Desenvolvimento Formal de Sistema

Desenvolvimento orientado a reuso

Primeiro Modelo!

Codificar e Consertar (Code and Fix)



Modelo Cascata (Waterfall)

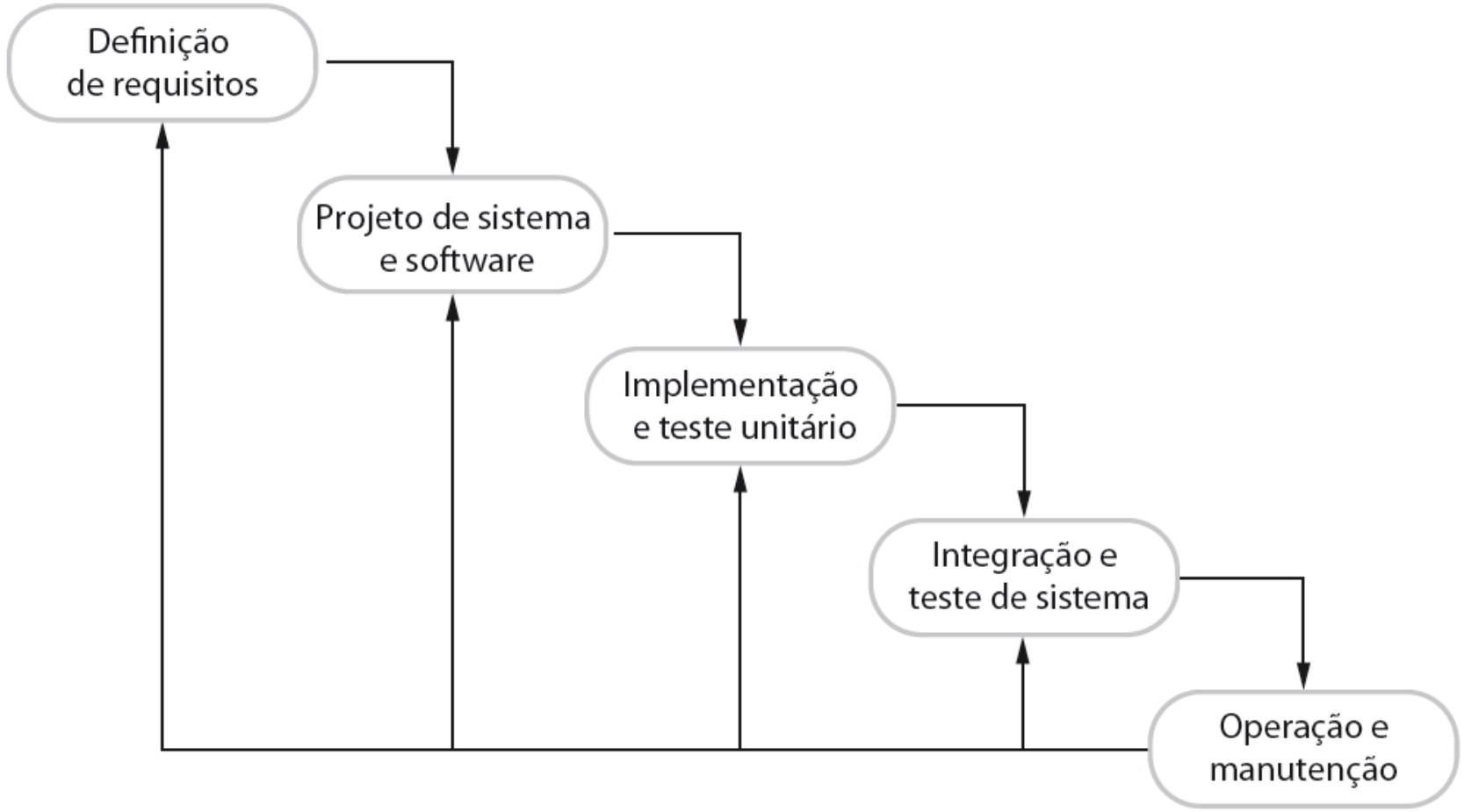
Modelo mais antigo e o mais amplamente usado da engenharia de software

Modelado em função do ciclo da engenharia convencional

Requer uma abordagem sistemática, sequencial ao desenvolvimento de software

Muitas organizações que usam esse modelo, aplicam-no de forma estritamente linear

Modelo Cascata (Waterfall)



Modelo Cascata (Waterfall)

Existem fases identificadas e separadas no modelo cascata:

- Análise e definição de requisitos
- Projeto de sistema e software
- Implementação e teste de unidade
- Integração e teste de sistema
- Operação e manutenção

O principal problema do modelo cascata é a dificuldade de acomodação de mudanças depois que o processo já foi iniciado. Em princípio, uma fase precisa ser completada antes de se mover para a próxima fase

Modelo Cascata (Waterfall) Problemas

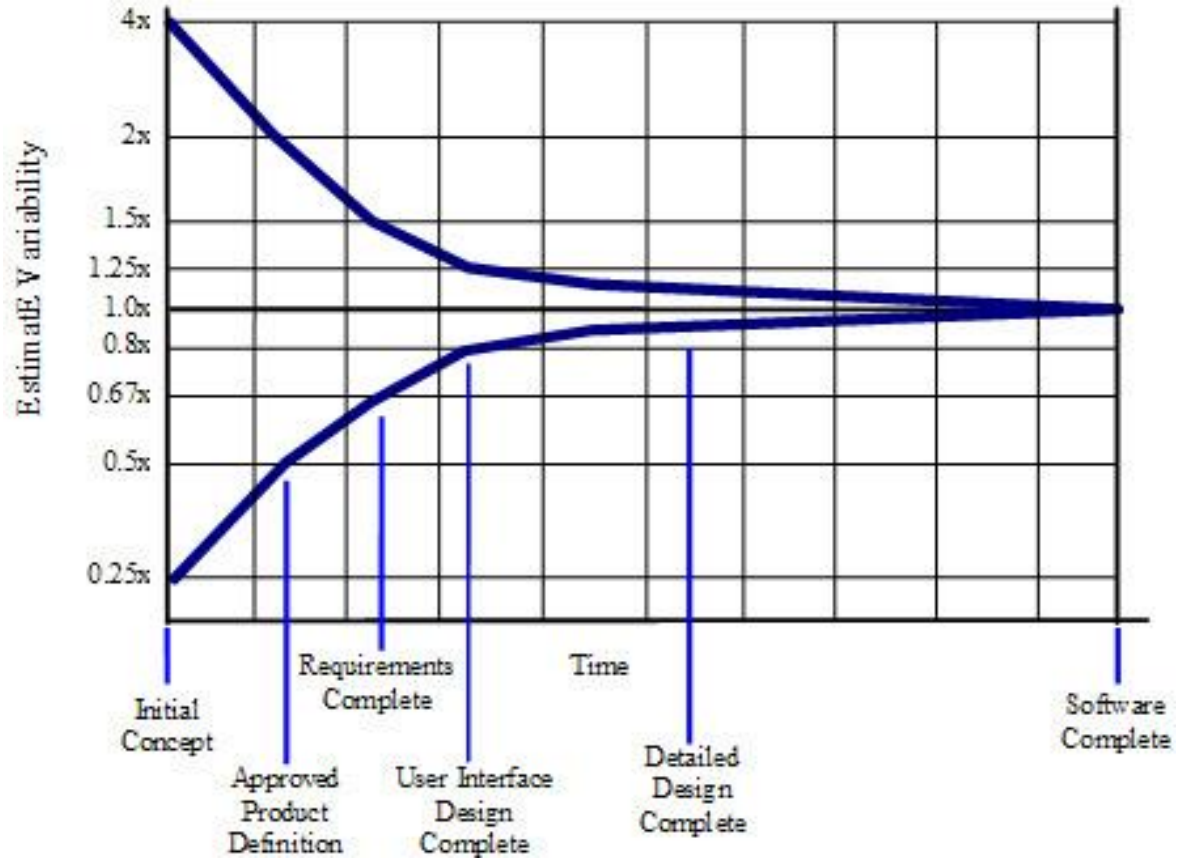
Projetos reais raramente seguem o fluxo sequencial que o modelo propõe

Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural (Cone da Incerteza!)

O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento

Muitas vezes os desenvolvedores ficam ociosos desnecessariamente, devido a estados bloqueadores (quando existem tarefas dependentes, membros da equipe devem aguardar que outros terminem)

Cone da Incerteza



Modelo Cascata (Waterfall) Problemas

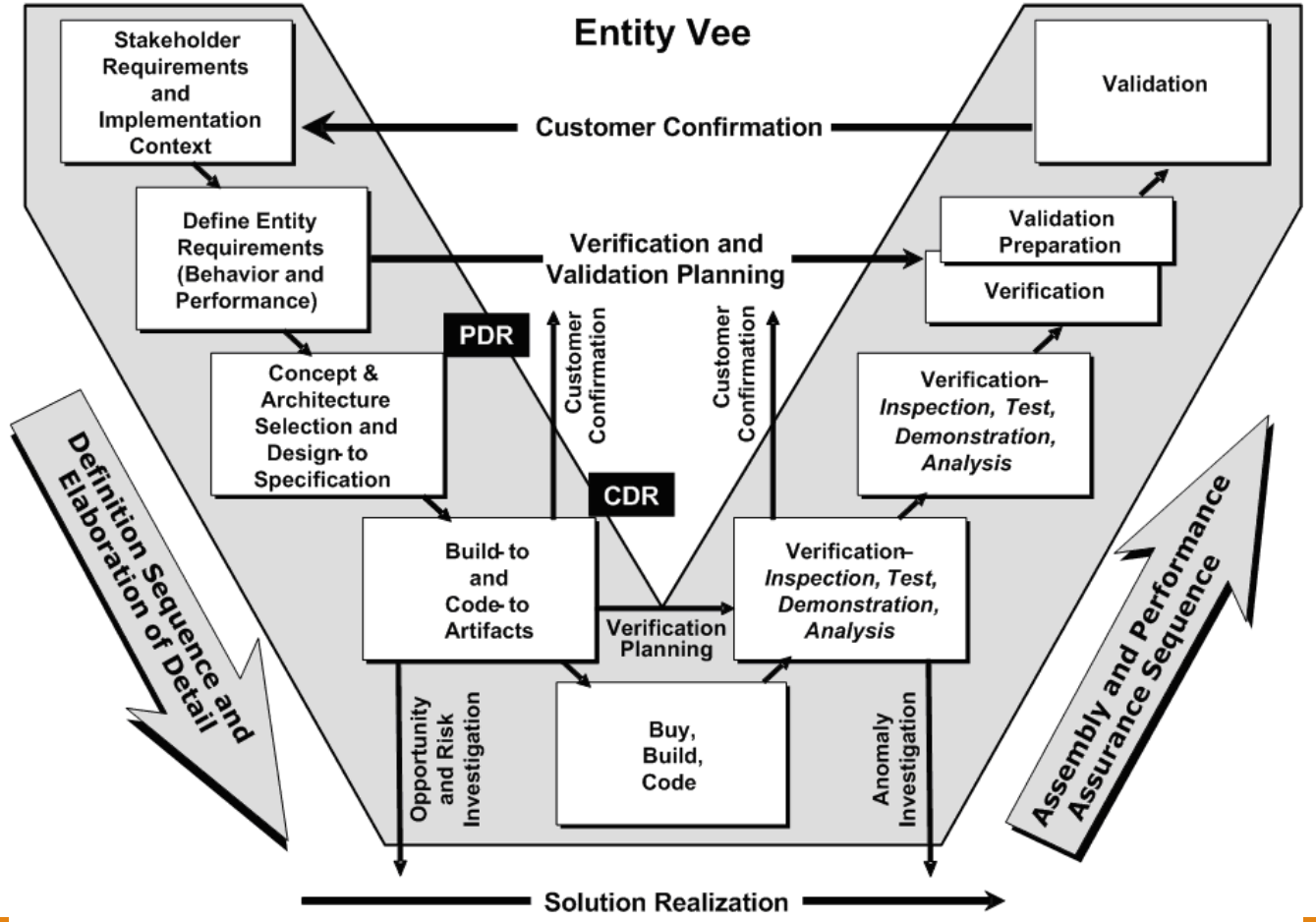
Divisão inflexível do projeto em estágios distintos torna difícil responder às mudanças nos requisitos do cliente.

- Por isso esse modelo só é apropriado quando os requisitos são bem entendidos e as mudanças durante o processo de projeto serão limitadas.
- Poucos sistemas de negócio possuem requisitos estáveis.

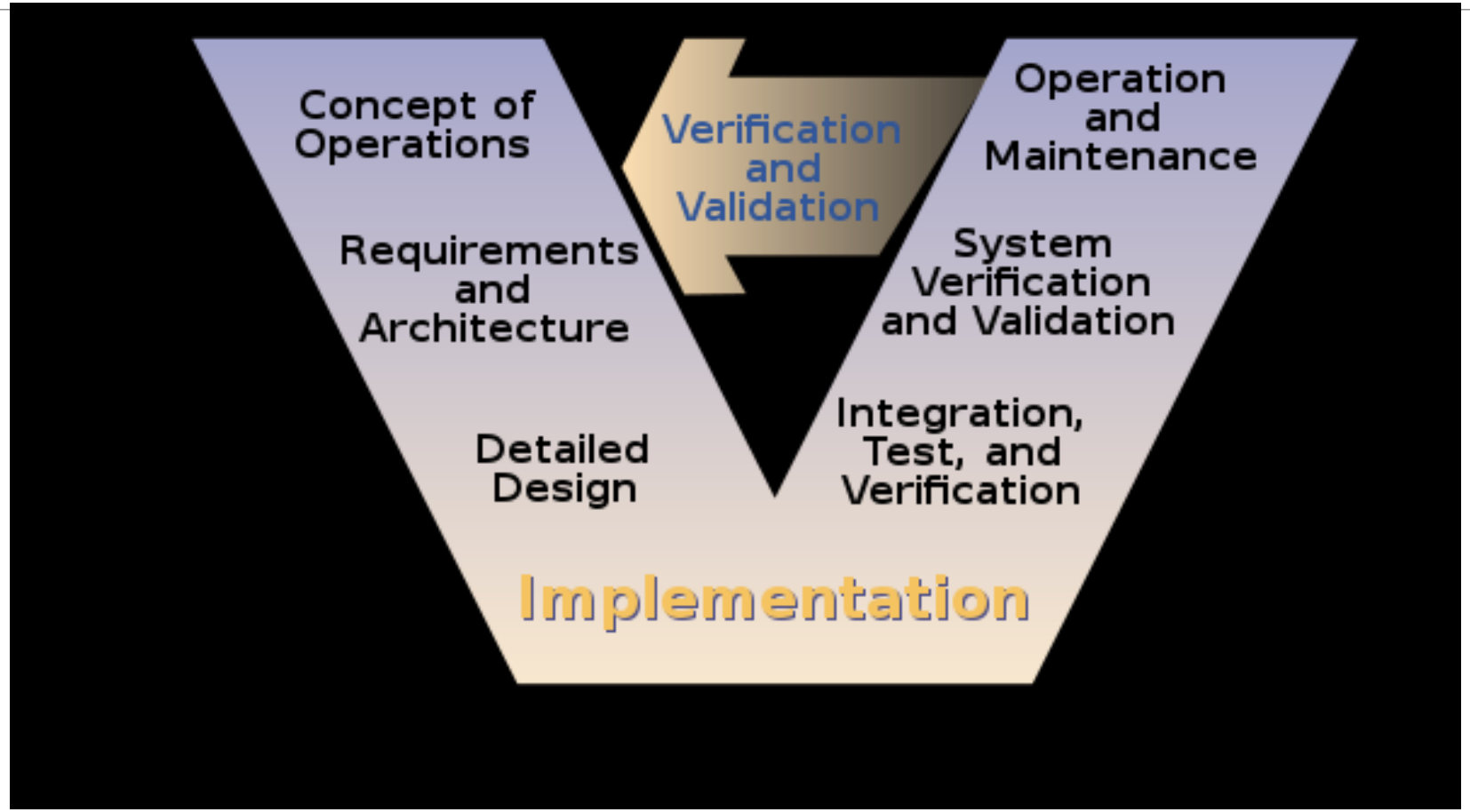
O modelo cascata é mais usado em projetos de engenharia de grandes sistemas onde o sistema é desenvolvido em vários locais (Engenharia de Sistemas).

- Nessas circunstâncias, a natureza do modelo cascata dirigida a planos ajuda a coordenar o trabalho.

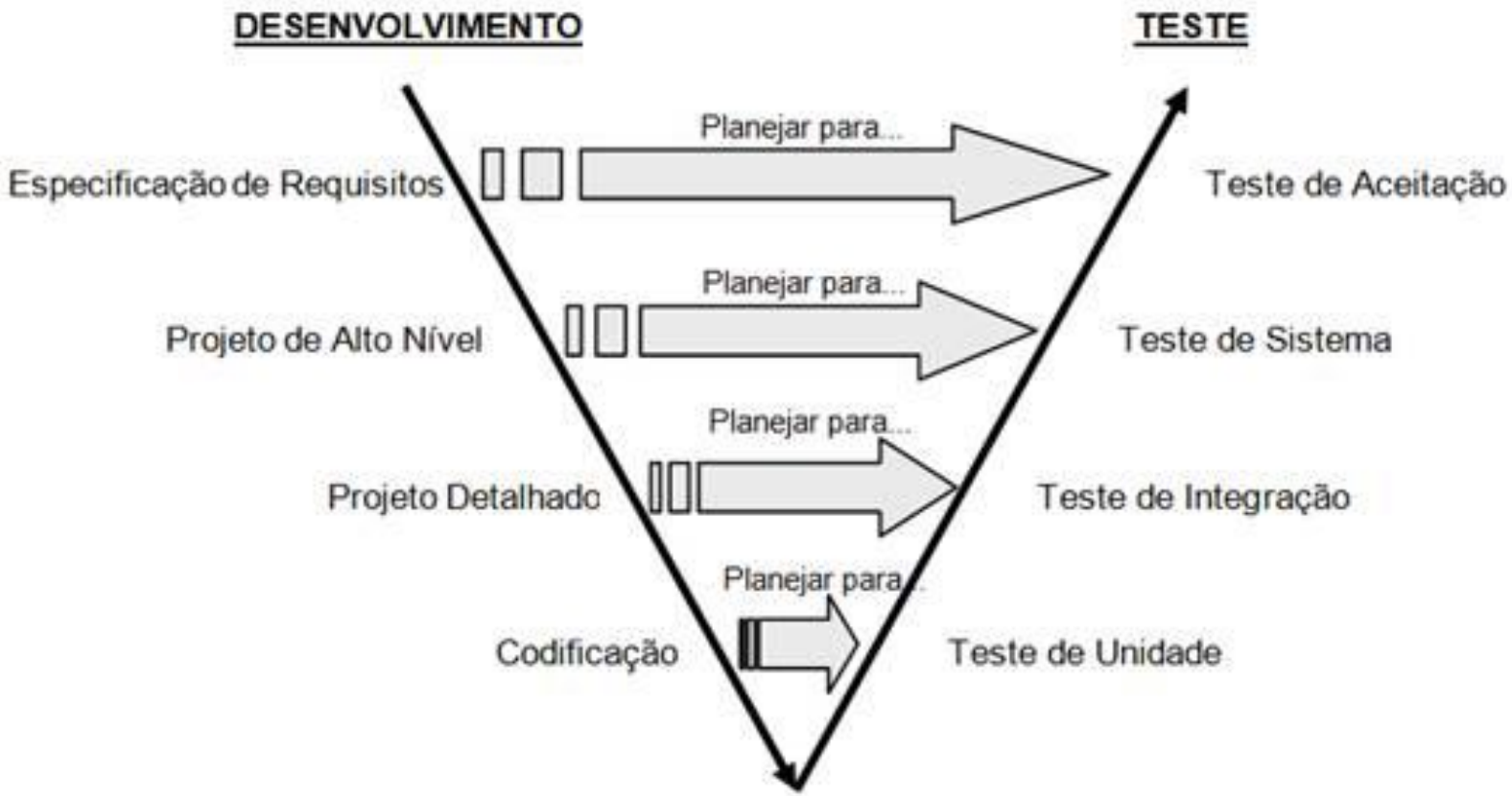
Modelo V (Vee-Model)



Modelo V (Vee-Model)



Modelo V (Vee-Model)



Desenvolvimento Evolucionário

Tem com base a ideia de desenvolver uma implementação inicial, expor o resultado ao comentário do usuário e fazer seu aprimoramento por meio de muitas versões, até que tenha sido desenvolvido

A especificação, desenvolvimento e validação são executados concorrentemente para gerar um retorno rápido

Desenvolvimento Evolucionário

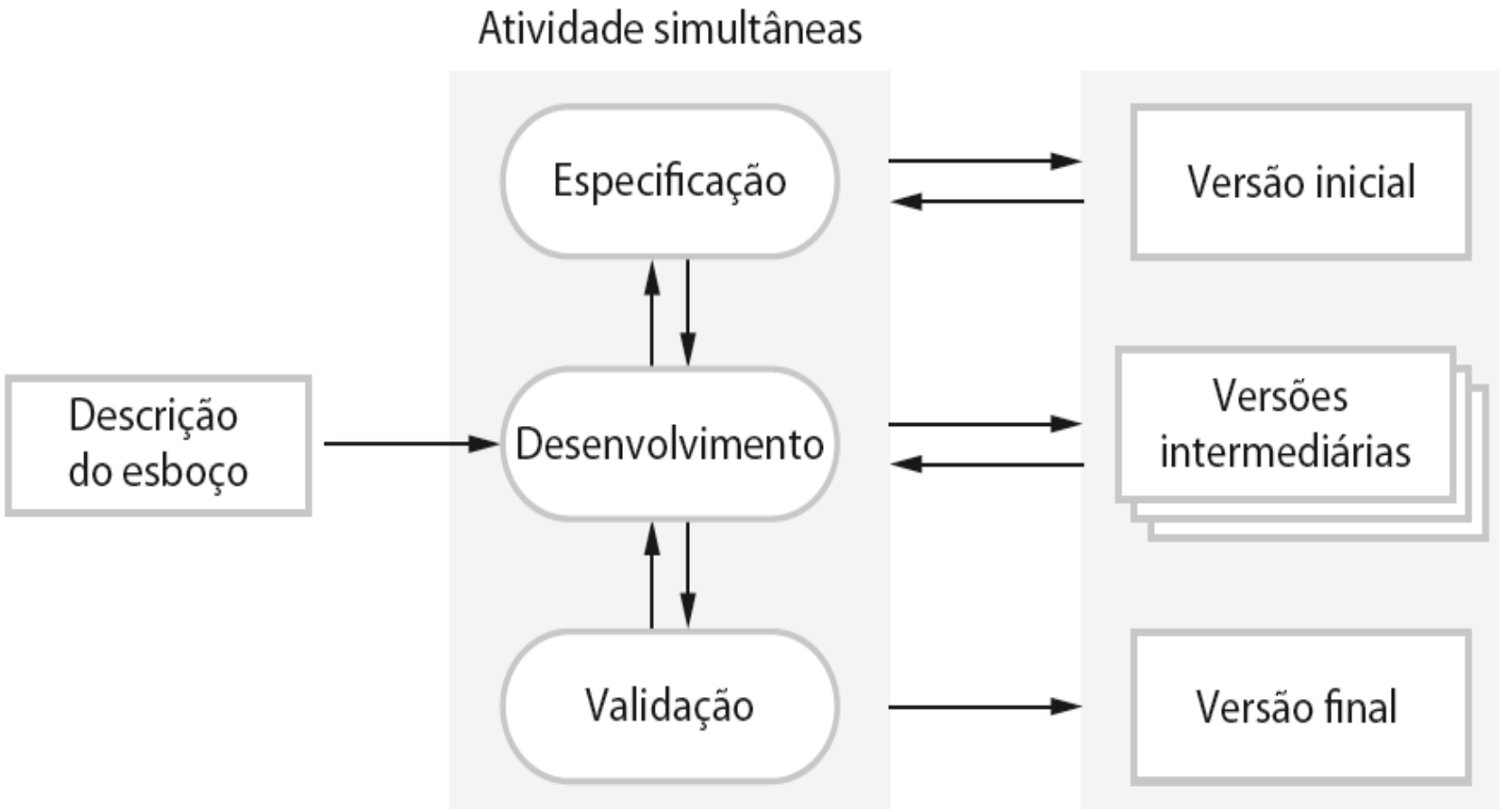
Desenvolvimento exploratório

- O objetivo é trabalhar com o cliente, a fim de explorar seus requisitos e entregar um sistema final.
- Inicia-se com partes do sistema que são compreendidas
- O Sistema evolui com o acréscimo de novas características a medida que elas são propostas pelo cliente

Prototipação (descartável)

- Tenta compreender os melhor os requisitos a partir de protótipos e então desenvolver uma especificação de requisitos completa

Desenvolvimento Evolucionário



Desenvolvimento Evolucionário

Problemas:

- O processo não é visível: como o sistema é desenvolvido rapidamente, não há tempo de documentar as versões;
- Os sistemas são mal estruturados: mudanças constantes podem corromper a estrutura do software;
- Requer ferramentas e técnicas especiais: que nem sempre são disponíveis ou são aplicáveis ao caso.

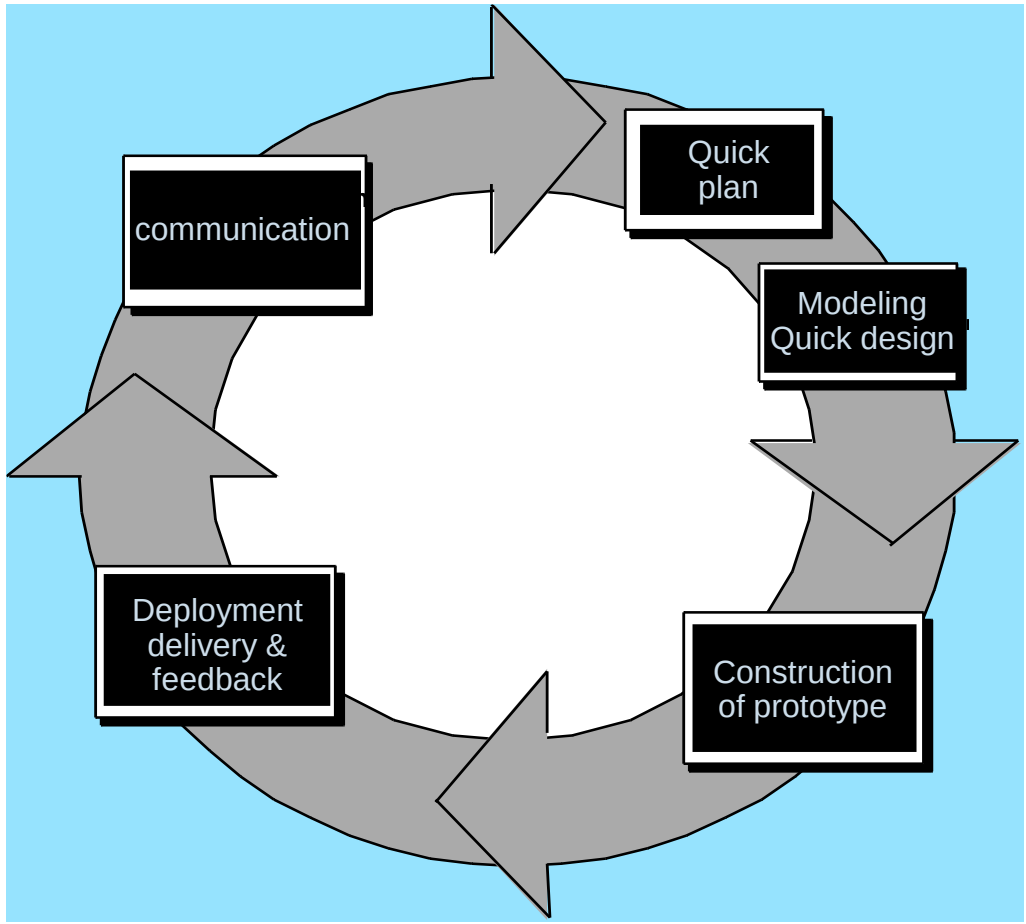
Prototipação

Processo que possibilita que o desenvolvedor crie um modelo do software que deve ser construído.

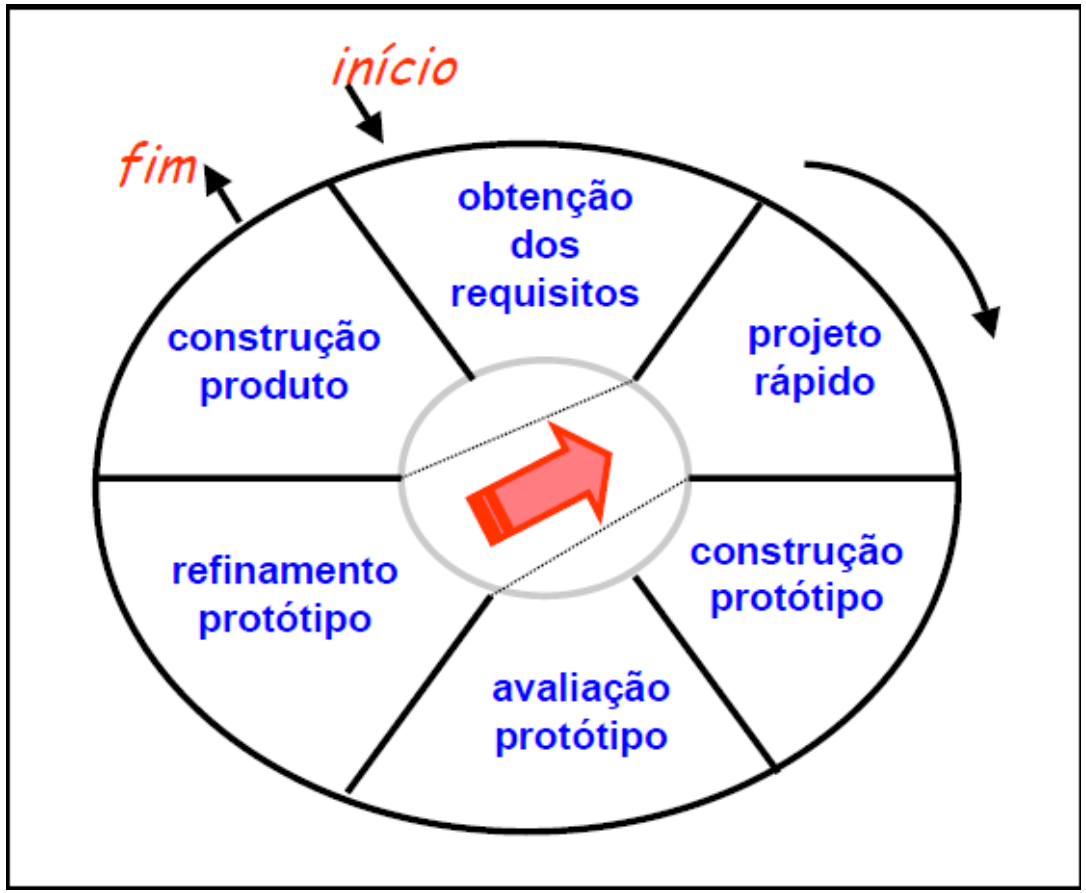
Idealmente, o modelo(protótipo) serve como um mecanismo para identificar os requisitos de software.

Apropriado para quando o cliente definiu um conjunto de objetivos gerais para o software, mas não identificou requisitos de entrada, processamento e saída com detalhes.

Prototipação



Prototipação



Problemas da Prototipação

Cliente não sabe que o software que ele vê não considerou, durante o desenvolvimento, a qualidade global e a *manutenibilidade* a longo prazo.Não aceita bem a ideia de que a versão final do software vai ser construída e "força" a utilização do protótipo como produto final

Desenvolvedor frequentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo. Depois de um tempo ele se familiariza com essas escolhas, e esquece que elas não são apropriadas para o produto final.

Considerações Importantes

Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente

A chave é definirem-se 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 a fim de definir os requisitos

Desenvolvimento Formal de Sistemas

É uma abordagem do desenvolvimento de software que tem algo em comum com o modelo em cascata, mas cujo o processo de desenvolvimento se baseia na transformação matemática formal de uma especificação do sistema em um programa executável

Difere do Modelo em Cascata:

- A especificação de requisitos de software é redefinida em uma especificação formal detalhada, que é expressa em notação matemática;
- O processo de desenvolvimento de projeto, implementação e teste de unidade são substituídos por um processo de desenvolvimento transformacional, em que a especificação é refinada, por meio de uma série de transformações, em um programa

Desenvolvimento Formal de Sistemas

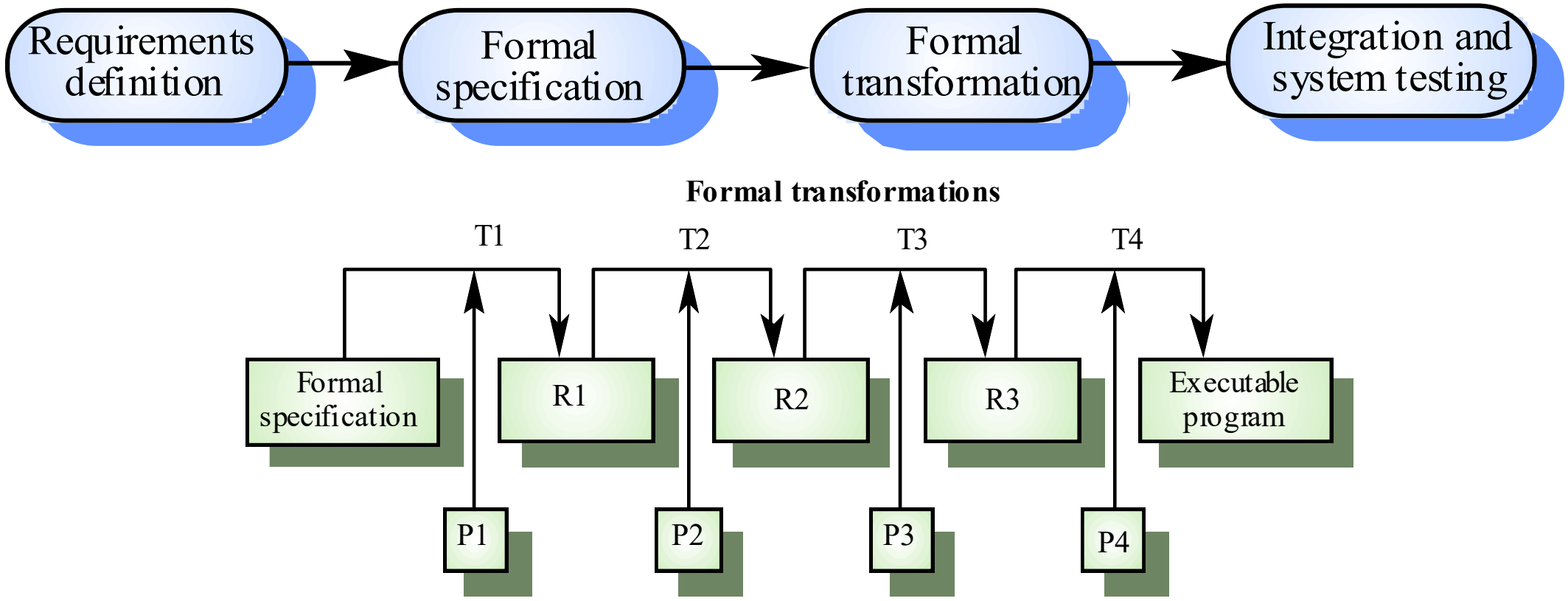
Especificação, desenvolvimento e verificação → aplicação de uma notação matemática

Elimina-se problemas de ambiguidade, incompletude e inconsistência → aplicação de análise matemática

Aplicação: sistemas críticos!

Exemplos: Linguagem Z, CSP (*Communicating Sequential Processes*), π-calculus, Redes de Petri e etc..

Desenvolvimento Formal de Sistemas



Desenvolvimento Formal de Sistemas

Essa abordagem é particularmente adequada ao desenvolvimento de sistemas que tenham rigorosas exigências de segurança, confiabilidade e garantia. No entanto, para a maioria dos sistemas não há um ganho significativo de custo ou qualidade.

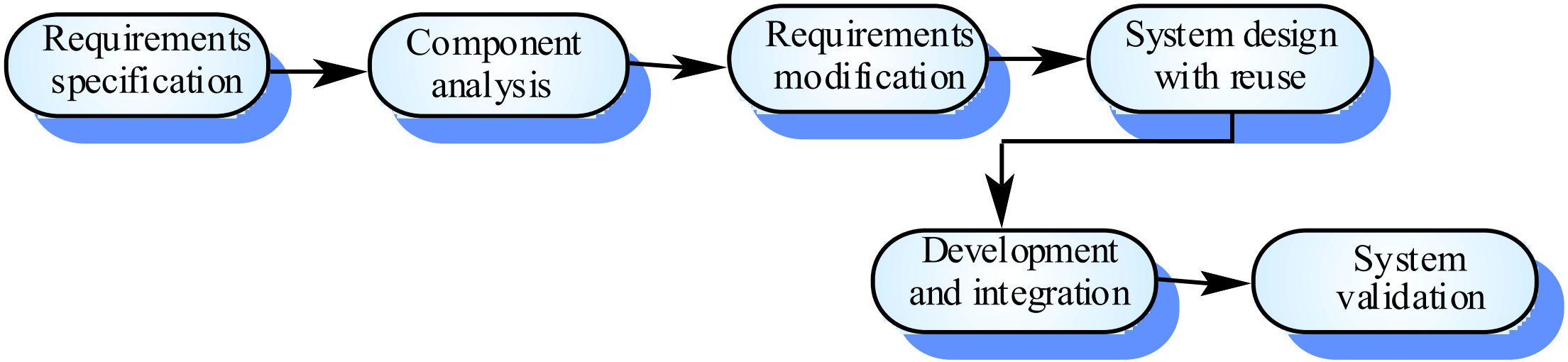
Necessita de perícia especializada.

Desenvolvimento Orientado a Reuso

O reuso é comum no projeto. Mas a engenharia de software formaliza o reuso com uma ampla base de componentes reutilizáveis e com alguma infraestrutura de integração para estes componentes

Acelera a produção de resultados, especialmente em prototipagens

Desenvolvimento Orientado a Reuso



Desenvolvimento Orientado a Reuso

Análise de componentes: considerando a especificação dos requisitos, é feita uma busca de componentes para implementar essa especificação

Modificação dos requisitos: adequação dos requisitos aos componentes encontrados. Se não puderem ser alterados, repete-se a análise

Projeto de sistema com reuso: é desenvolvida uma infraestrutura ou reutilizada uma preexistente. Organizam os componentes que serão utilizados e projetam o que faltam;

Desenvolvimento e integração: os componentes faltantes serão desenvolvidos e todos integrados a fim de criar um sistema.

Desenvolvimento Orientado a Reuso

Reduz a quantidade de software a ser desenvolvido e, consequentemente, diminui custos e riscos;

Acelera a entrega do software;

Os requisitos podem se perder durante a adaptação;

Pode se perder o controle da versão dos componentes utilizados no sistema.

Modelo Baseado em Componentes

Utiliza componentes de software comercial ou COTS (Commercial Off-The-Shelf)

Incorpora muitas das características do modelo espiral

É evolucionário em sua natureza, demandando uma abordagem iterativa para a criação de software

As aplicações são desenvolvidas a partir de componentes de software pré-empacotados

- Desenvolvimento por montagem X contrução
- Ex: componentes de interface (botões, campo editável, tabela e etc..)

Iteração de Processo

VISÃO GERAL

Iteração de Processo

Modelos de processos apresentados → tem vantagens e desvantagens

Para a maioria dos grandes sistemas é necessário utilizar diferentes abordagens para diferentes partes do sistema → Modelo Híbrido

Processo iterativo → partes do processo são repetidas à medida que os requisitos do sistema evoluem

Modelos Híbridos → apoiam diferentes abordagens de desenvolvimento e foram explicitamente projetados para processo iterativos

- Desenvolvimento Incremental
- Desenvolvimento em Espiral

Iteração de Processo

Importante:

- Essência do processo iterativo → a especificação é desenvolvida em conjunto com o software
- Isso gera um conflito com o modelo de fornecimento de muitas empresas!!!
  - A especifica completa faz parte do contrato → eventos de pagamento ($$$)
- Na abordagem gradativa não há uma especificação completa do sistema até que o estágio final seja alcançado.
  - Isso requer um novo tipo de contrato (isso é complicado com órgãos do governo)

Desenvolvimento Incremental

Cascata → cliente precisa se comprometer com um conjunto de requisitos específicos antes do projeto

- Mudanças geram retrabalho (requisitos, projeto e implementação)
- Modelo fácil de ser gerenciado
- Separação projeto X implementação → levam a sistemas robustos, que são sensíveis a mudanças

Desenvolvimento Evolucionário → requisitos e decisões do projeto podem ser postergadas

- Leva a um software mal estruturado, difícil de manter e compreender

Desenvolvimento Incremental

Desenvolvimento incremental (Mills et al., 1980) → combina as vantagens desses dois modelos (cascata e evolucionário)

Principais objetivos

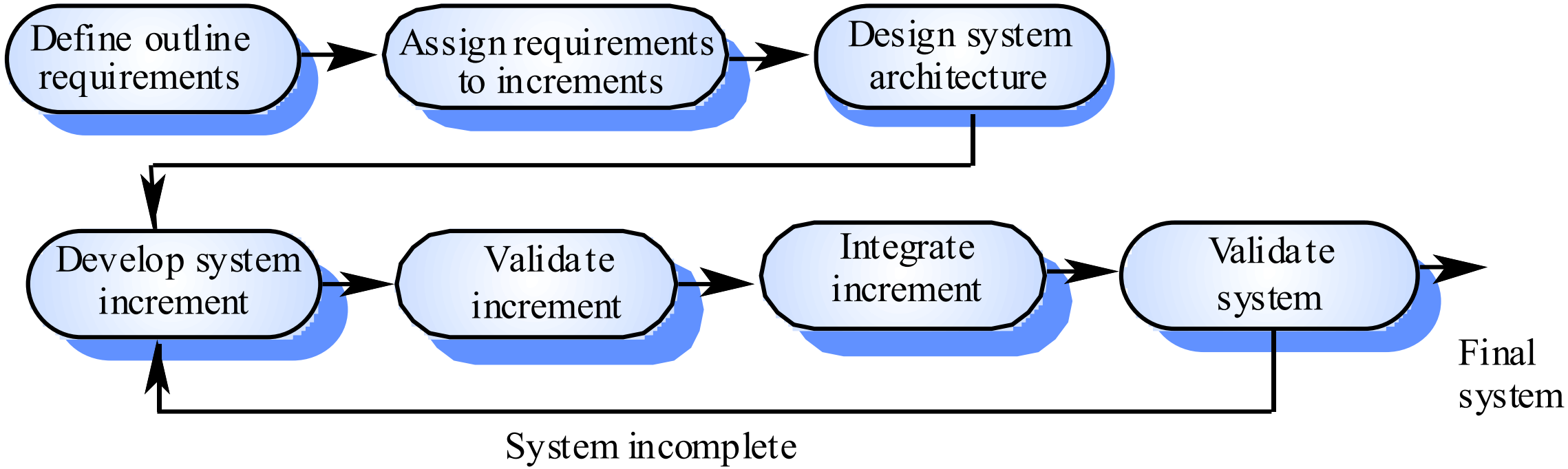
- Reduzir retrabalho (cascata) no processo de desenvolvimento
- Proporcionar aos clientes alguma oportunidade de adiar decisões sobre seus requisitos detalhados, até que eles tenham alguma experiência com o sistema

Desenvolvimento Incremental

Dinâmica do desenvolvimento incremental:

- Clientes identificam (esboço) os requisitos a serem fornecidos pelo sistema
- Clientes priorizam os requisitos (mais e menos importantes)
- Em seguida é definida uma série de estágios de entrega, com cada estágio fornecendo um subconjunto de funcionalidades do sistema.

Desenvolvimento Incremental



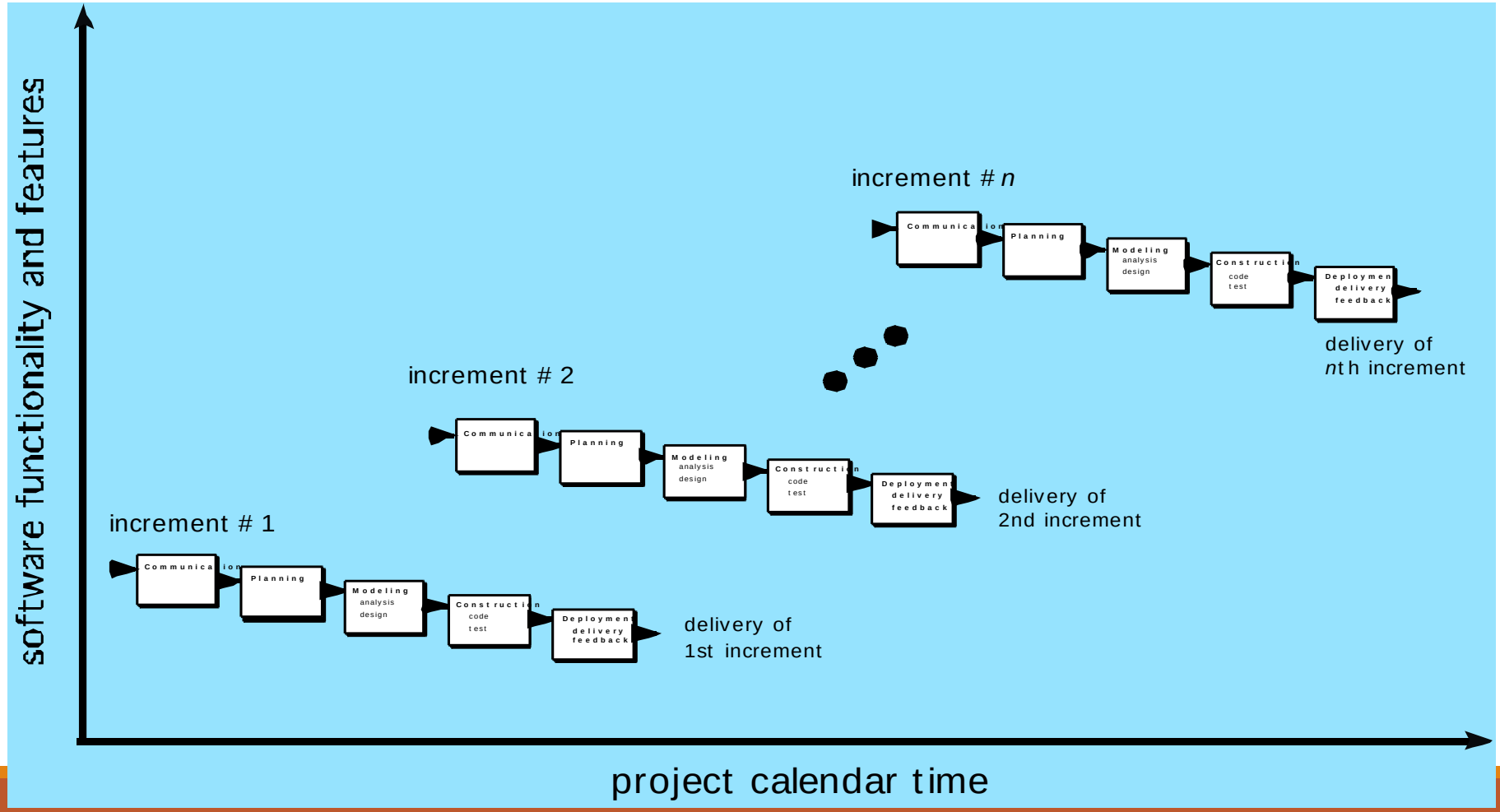
Desenvolvimento Incremental

Identificado o incremento --> Os requisitos são detalhados e desenvolvidos usando o processo mais adequado

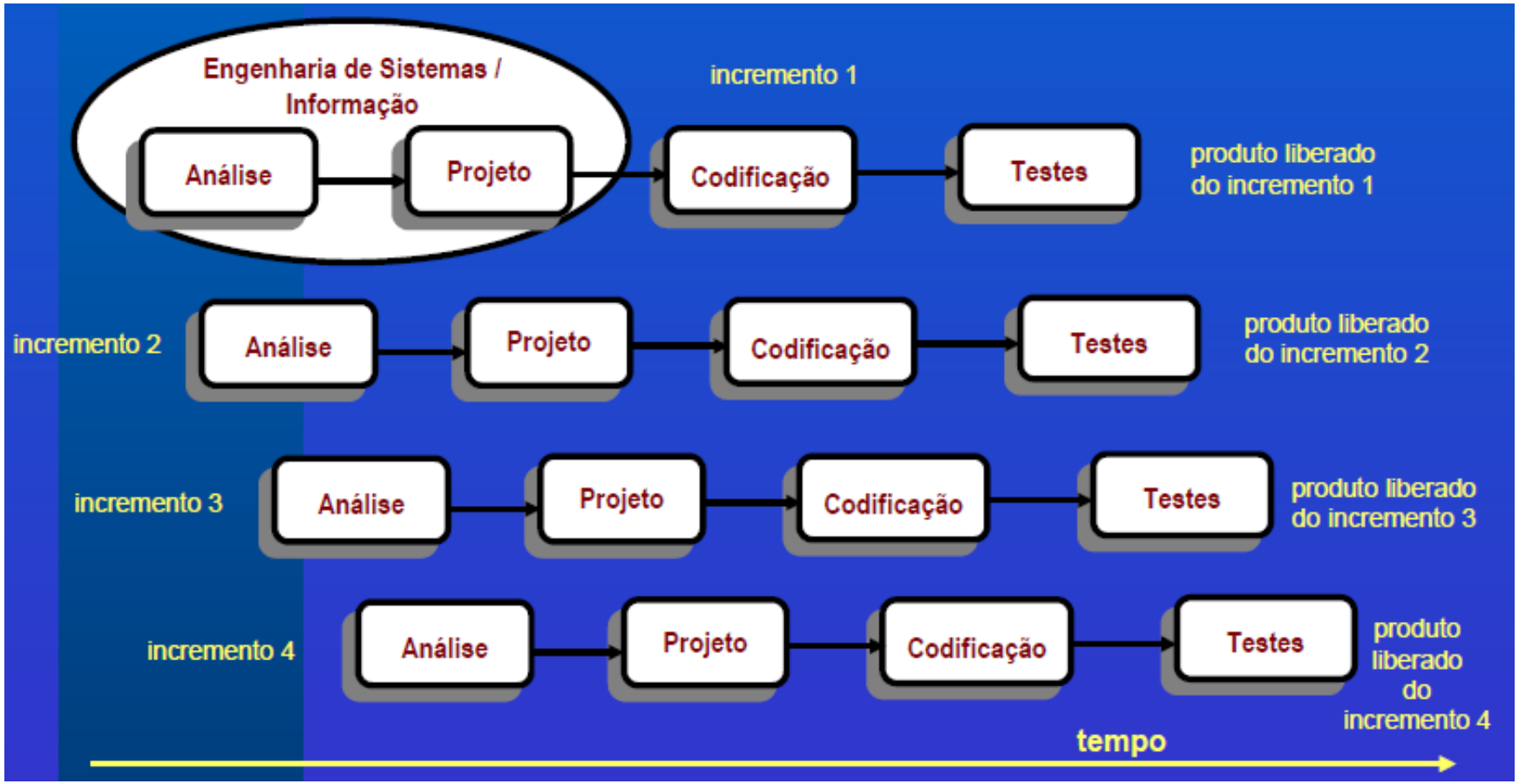
Uma vez que o incremento é concluído e entregue, os clientes já podem colocá-lo em operação.

Para cada incremento, pode-se utilizar um processo diferente.

Desenvolvimento Incremental



Desenvolvimento Incremental



Benefícios do desenvolvimento incremental

O custo para acomodar mudanças nos requisitos do cliente é reduzido.

- A quantidade de análise e documentação que precisa ser feita é bem menor do que o necessária no modelo cascata.

É mais fácil obter feedback do cliente sobre o trabalho de desenvolvimento que tem sido feito.

- Os clientes podem comentar demonstrações do software e ver quanto foi implementado.

Possibilidade de mais rapidez na entrega e implantação de software útil para o cliente.

- Os clientes podem usar e obter ganhos do software mais cedo do que é possível no processo cascata.

Problemas do desenvolvimento incremental

O processo não é visível.

- Gerentes precisam de entregas regulares para medir o progresso. Se os sistemas são desenvolvidos de forma rápida, não é viável do ponto de vista do custo produzir documentação para refletir todas as versões do sistema.

A estrutura do sistema tende a degradar conforme novos incrementos são adicionados.

- A menos que tempo e dinheiro sejam gastos na reconstrução para melhorar o software, as mudanças regulares tendem a corromper a estrutura do sistema. A incorporação posterior de mudanças no software se torna progressivamente mais difícil e cara.

Desenvolvimento em Espiral

Este modelo foi definido por Barry Boehm em 1996 (artigo: "***A Spiral Model of Software Development and Enhancement***"

Representa o processo de software como um espiral

Cada loop na espiral representa uma fase do processo do software

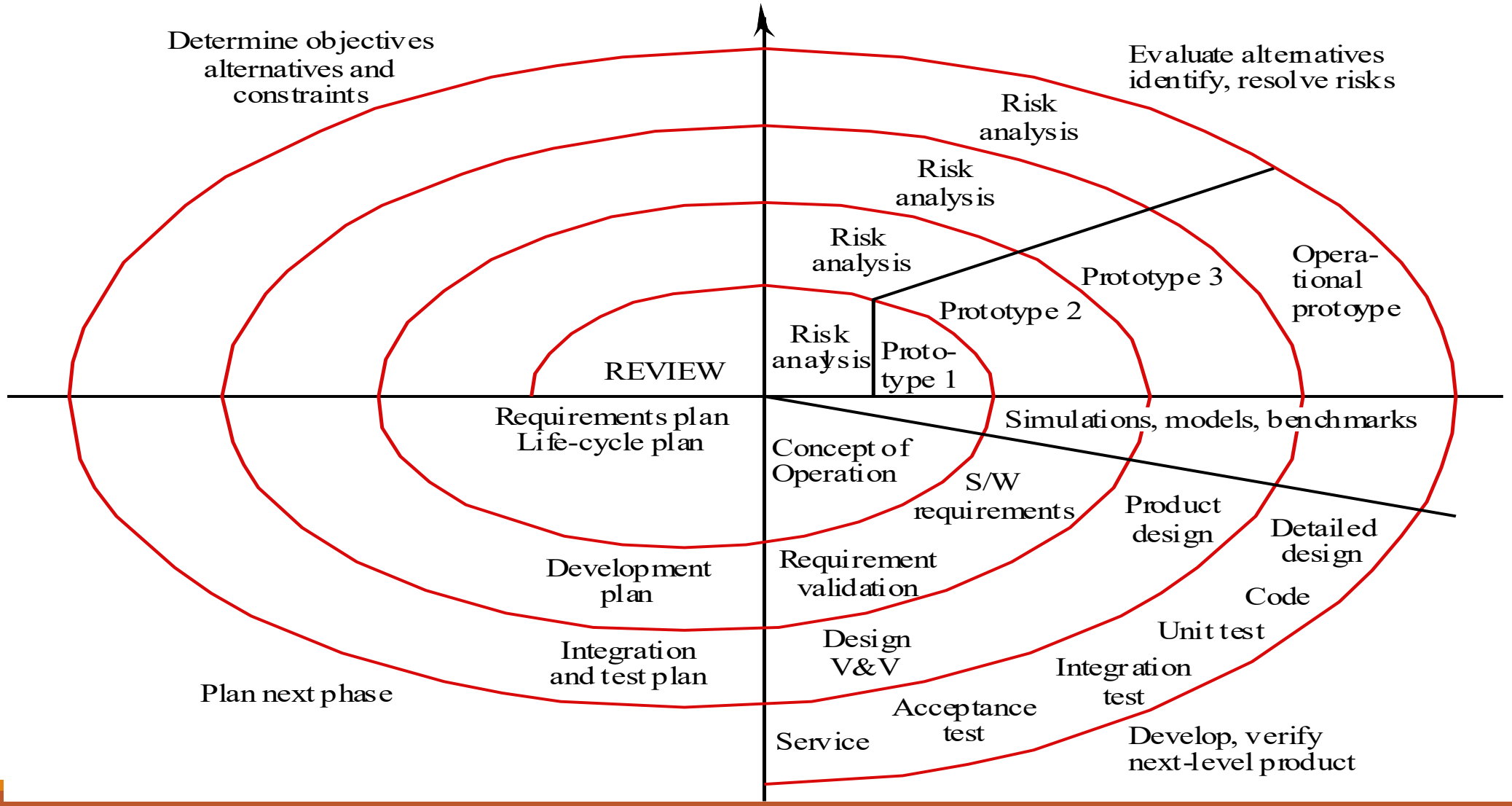
- Loop interno pode estar relacionado à viabilidade do sistema
- Loops seguintes → definição dos requisitos do sistema e os próximos ao projeto
- Assim vai até a fase de operação

Desenvolvimento em Espiral

Cada loop da espiral é dividido em quatro setores:

- Definição dos objetivos
- Avaliação e redução de riscos
- Desenvolvimento e validação
- Planejamento

Desenvolvimento em Espiral



Desenvolvimento em Espiral

Principal característica → Explícita consideração dos riscos

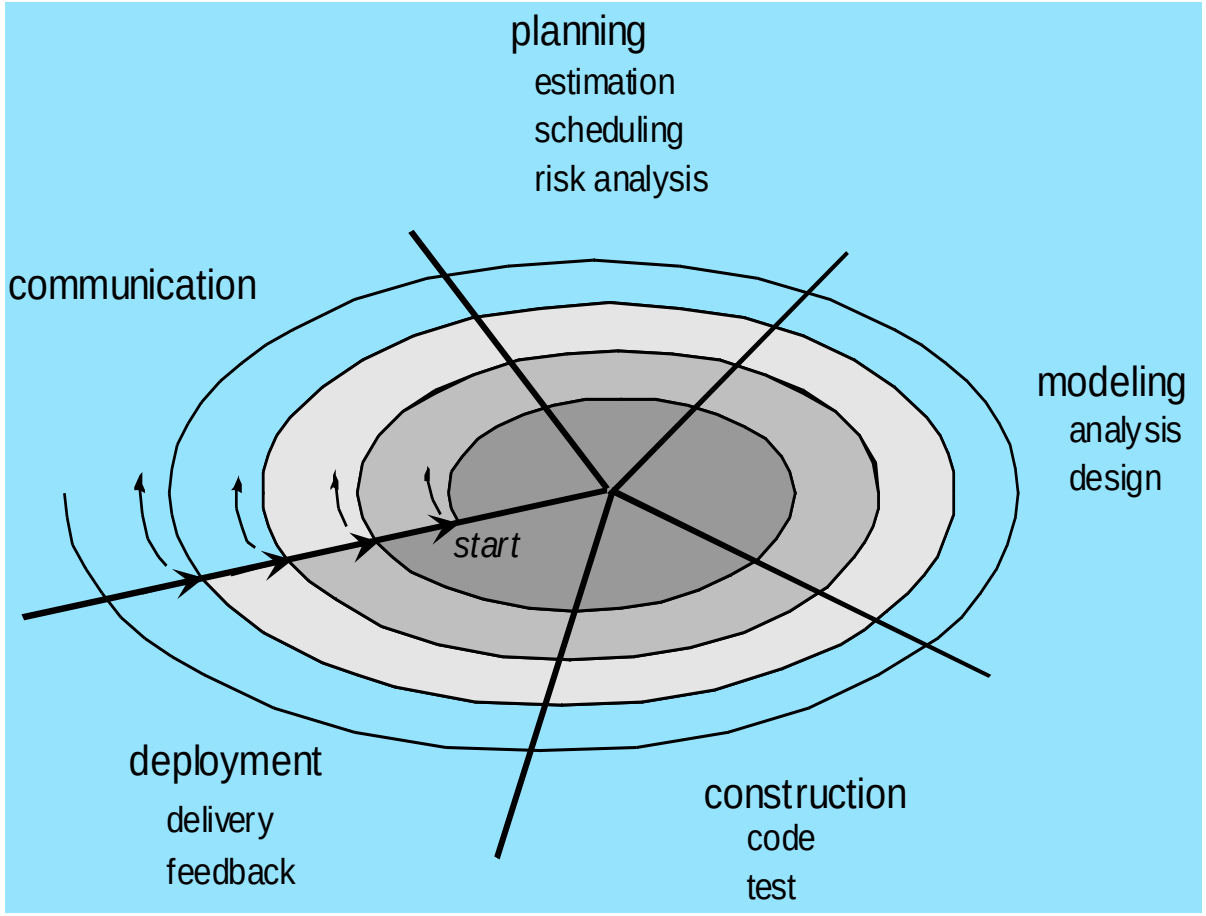
Engloba a natureza iterativa da Prototipação com os aspectos sistemáticos e controlados do Modelo em Cascata

Fornece o potencial para o desenvolvimento rápido de versões incrementais do software

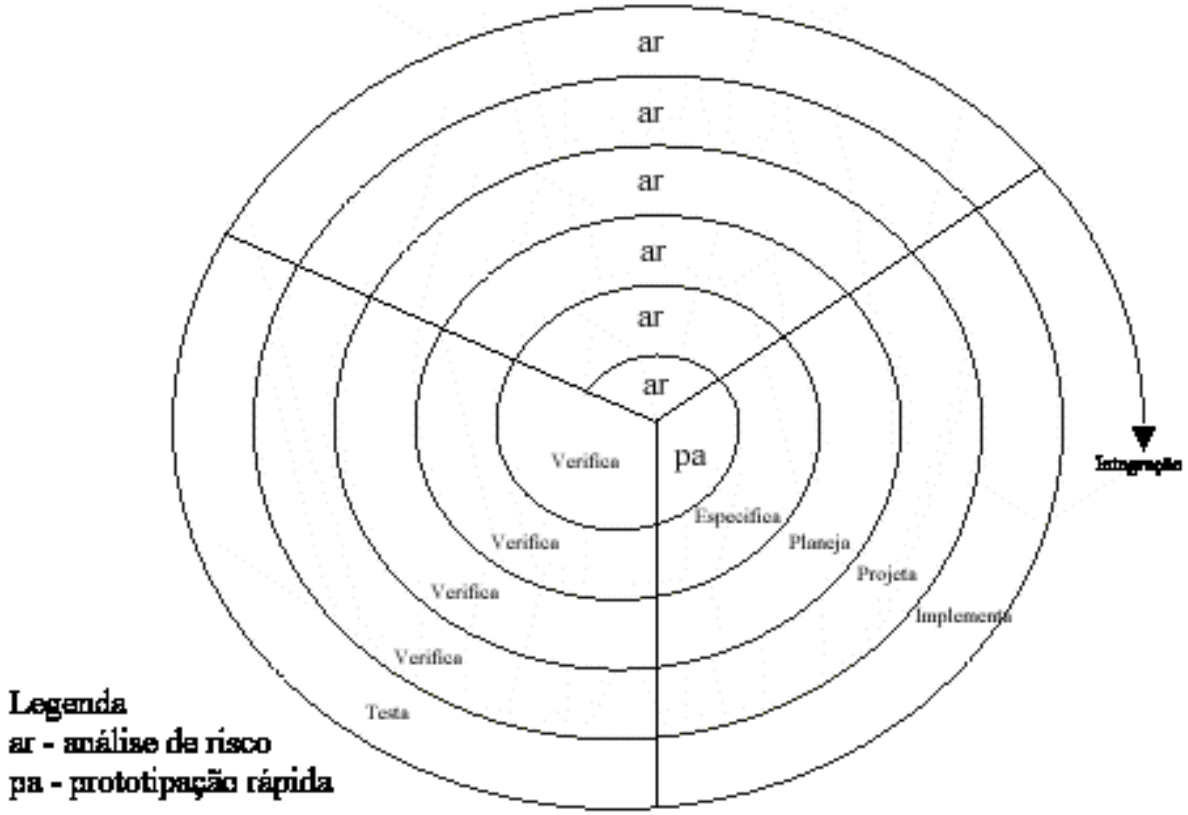
As primeiras iterações a versão incremental pode ser um modelo em papel ou um protótipo

Nas iterações mais adiantadas são produzidas versões incrementais mais completas e melhoradas

Desenvolvimento em Espiral



Desenvolvimento em Espiral



Desenvolvimento em Espiral

É, atualmente, a abordagem mais realística para o desenvolvimento de software em grande escala

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

Exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso (Gestão de Riscos)

Desenvolvimento em Espiral

O modelo é relativamente novo e não tem sido amplamente usado

Serão necessários alguns anos até que a eficácia desse modelo possa ser determinada com certeza absoluta.