UFSC
Araranguá

Modelos de Processos Prescritivos

Engenharia de Software II

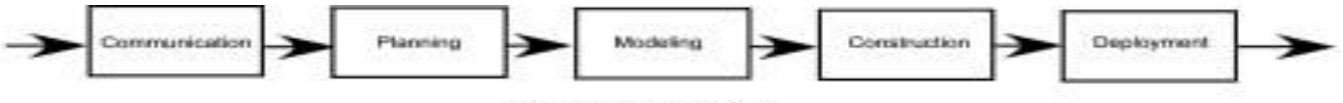
Profa. Andréa Sabedra Bordin

Processo de Software

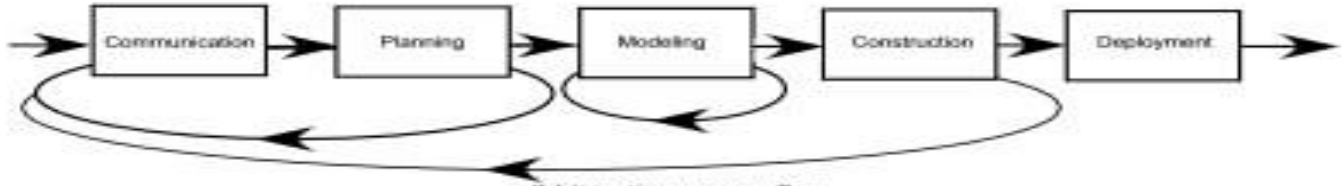
- Um conjunto estruturado de atividades requeridas para desenvolver um sistema de software.
  - Especificação
  - Projeto
  - Validação
  - Evolução

Para que o desenvolvimento de sistemas deixe de ser artesanal e aconteça de forma mais previsível, organizada e com mais qualidade, é necessário que se estabeleça e se compreenda um processo de produção.

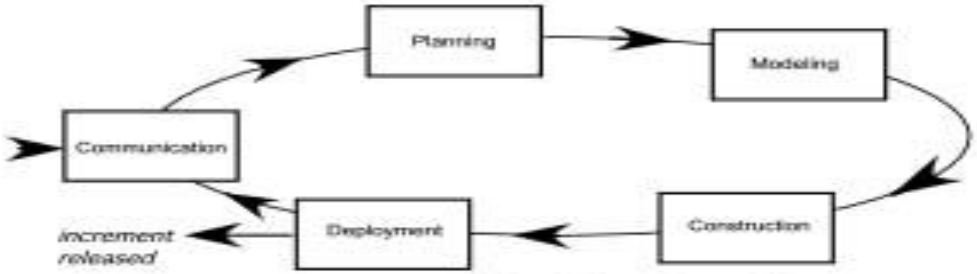
Fluxos de processos



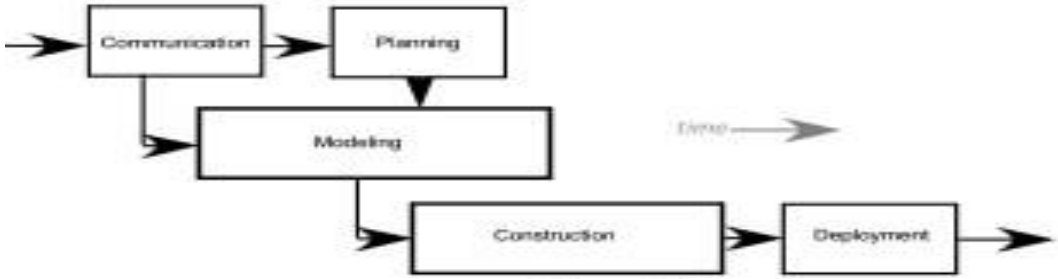
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

Descreve como a atividades e tarefas são organizadas de acordo com a sequência e o tempo

(PRESSMAN, 2015)

Fluxos de processos

- Um **fluxo de processo linear** executa cada uma das cinco atividades em sequência, começando com a comunicação e culminando com a implantação (Figura a).
- Um **fluxo de processo iterativo** repete uma ou mais atividades antes de passar para a próxima (Figura b).
- Um **fluxo de processo evolutivo (incremental)** executa as atividades de forma “circular”. Cada circuito de atividades conduz para uma versão mais completa do software (Figura c).
- Um **fluxo de processo paralelo** (Figura d) executa uma ou mais atividades em paralelo com outras atividades (por exemplo, modelagem para um aspecto do software pode ser executada em paralelo com a construção de outro aspecto do software).

Modelo de processo de software

- Processos de software são construídos de acordo com **modelos** (estilos)
- Um **modelo de processo de software** é uma representação abstrata de um processo.
  - Apresenta uma descrição de um processo de alguma perspectiva particular.
- Famílias de modelos de processo (WASLAWICK, 2013):
  - **Prescritivos**
  - Ágeis

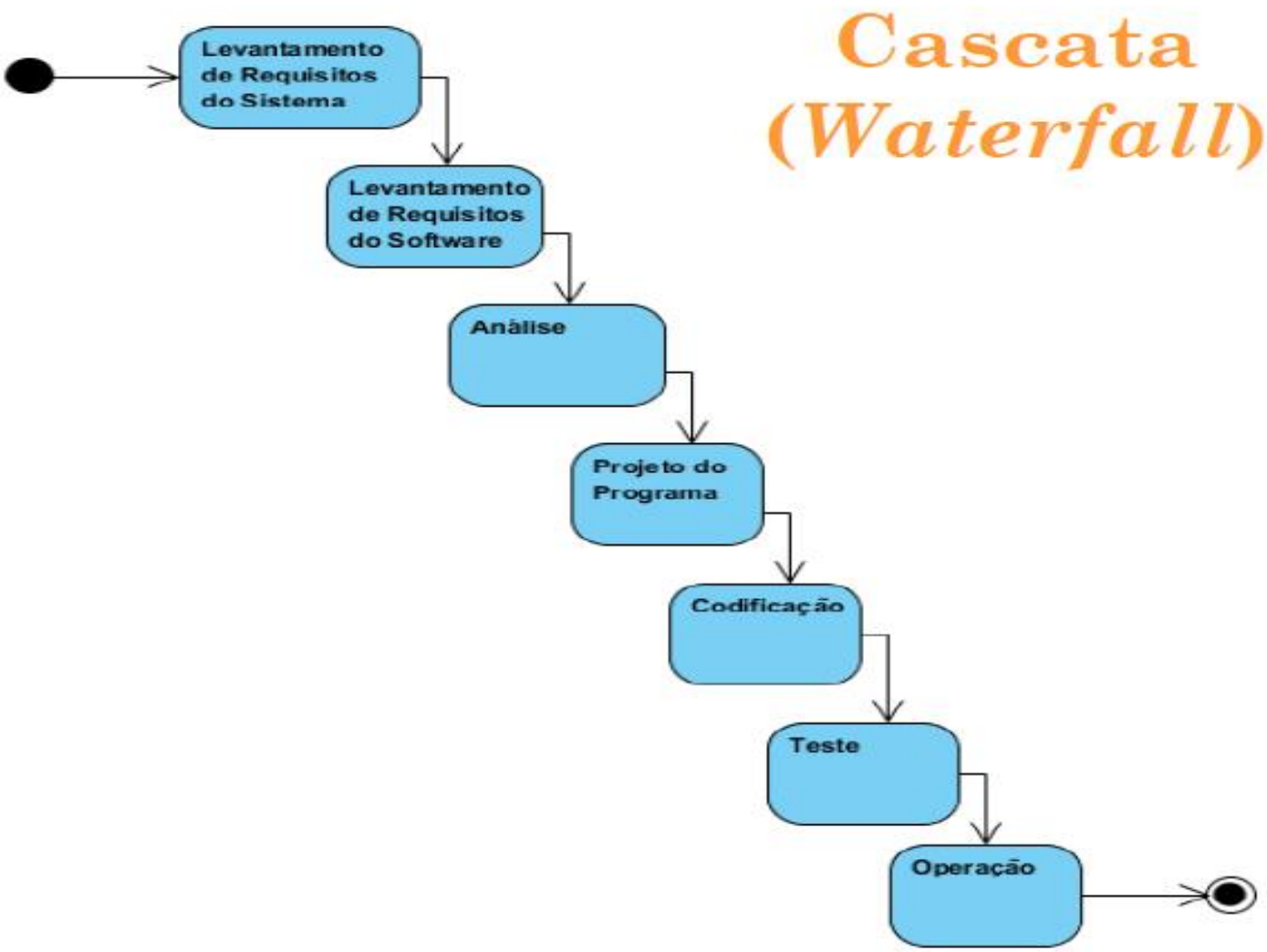
Existem outras classificações na literatura!

Modelos de Processo Prescritivos

- Prescrevem uma abordagem **ordenada** para a engenharia de software (PRESSMAN, 2015).
- Descrevem **como** as atividades são feitas (WASLAWICK, 2013).
- Cada modelo de processo também prescreve um **fluxo de processo** - isto é, a maneira pela qual os elementos do processo estão inter-relacionados entre si.

We call them "prescriptive" because they prescribe a set of process elements—framework activities, software engineering actions, tasks, work products, quality assurance, and change control mechanisms for each project **(PRESSMAN, 2015).**

Modelo Cascata



Modelo Cascata

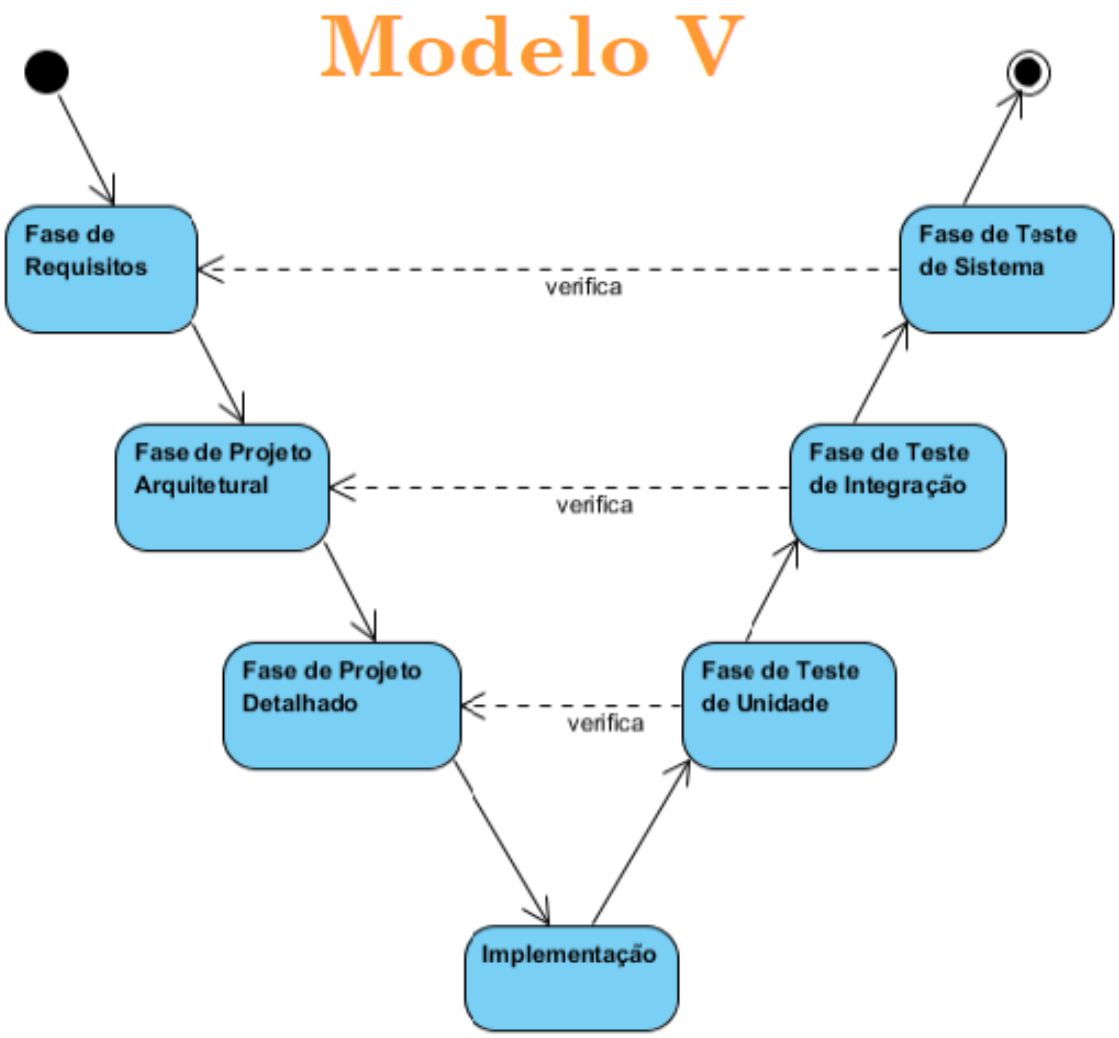
- Fluxo de processo linear.
  - As fases são executadas sequencialmente.
    - O sistema só pode ser implantado quando todo o ciclo estiver concluído.
- Introduz a noção que o desenvolvimento de software ocorre em fases bem definidas.
  - Primeiro modelo de processo - surgiu na década de 1970.
  - Detalhamento das fases de análise e projeto.
    - Orientado à documentação *(plan-oriented).*
- Desvantagens:
  - Dificuldade de estabelecer requisitos completos antes de começar a codificar. Cliente geralmente não consegue especificar todos os requisitos no inicio do processo.
  - Não é flexível
    - Uma fase só pode começar quando a anterior for completada.
    - Dificuldade de acomodar mudanças depois que o processo está em andamento.
- Apropriado quando os requisitos são bem entendidos.

Modelo Cascata

- Atualmente o desenvolvimento de software é acelerado e sujeito a um fluxo interminável de mudanças (funções, recursos, etc.).
- Assim o modelo em cascata é frequentemente inapropriado para esse tipo de trabalho.

- No entanto, é melhor que uma abordagem casual de desenvolvimento de software.

Modelo V


Modelo V
Fase de Requisitos
verifica
Fase de Teste de Sistema
Fase de Projeto Arquitetural
verifica
Fase de Teste de Integração
Fase de Projeto Detalhado
verifica
Fase de Teste de Unidade
Implementação

Modelo V

- Variação do Cascata, também com fluxo de processo linear.
- Prevê uma fase de **validação** e **verificação** para cada fase do processo.
- Assim como o Cascata, é o **orientado à documentação**.
- Enfatiza a importância dos testes no processo de desenvolvimento, desde o princípio e não apenas no final.
- No lado direito do “V”, diferentes tipos de teste **verificam** se os sistema satisfaz os requisitos especificados.
  - Fase de Teste de unidade: verifica se todas as unidades se comportam de acordo com a especificação detalhada.
  - Fase de Teste de integração: verifica se o sistema de comporta conforme a especificação do projeto arquitetural.
  - Fase de Teste de sistema: verifica se o sistema satisfaz os requisitos especificados.
  - Garantem a qualidade do software.

Modelo V

- A ligação entre os lados direito e esquerdo do modelo V implica que, caso sejam encontrados problemas em uma atividade de teste, a correspondente fase do lado esquerdo e suas fases subsequentes podem ser executadas novamente para corrigir ou atenuar esses problemas.
- Desvantagens:
  - As mesmas do cascata.
  - Requisitos imprecisos dificultam os testes do lado direito.
- Indicado para projetos com requisitos estáveis.

Modelos com fluxos lineares (waterfall e variações) não funcionaram com software!

Software é diferente

- Engenharia de Software ≠ Engenharia Tradicional
- Software ≠ (carro, ponte, casa, avião, celular, etc)
- Software ≠ (produtos físicos)
- Software é abstrato e "adaptável"

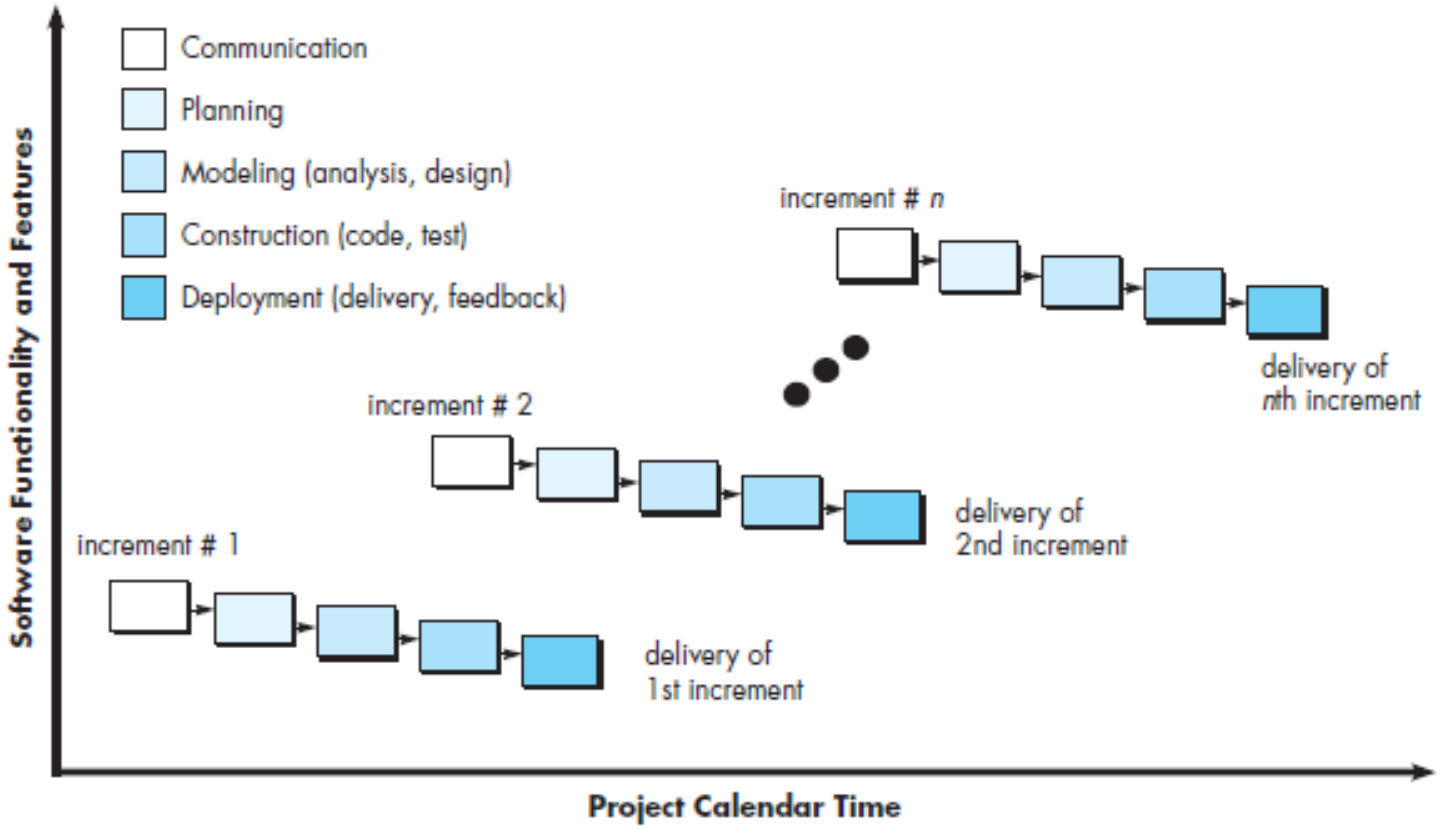
Dificuldade 1: Requisitos

- Clientes não sabem o que querem (em um software)
  - Funcionalidades são "infinitas" (difícil prever)
  - Mundo muda!
- Não dá mais para ficar 1 ano levantando requisitos, 1 ano projetando, 1 ano implementando, etc.
- Quando o software ficar pronto, ele estará obsoleto!

Dificuldade 2: Documentações Detalhadas

- Verbosas e pouco úteis.
- Na prática, muitas vezes desconsideradas durante implementação.
- *Plan-and-document* não funcionou com software.

Modelo incremental



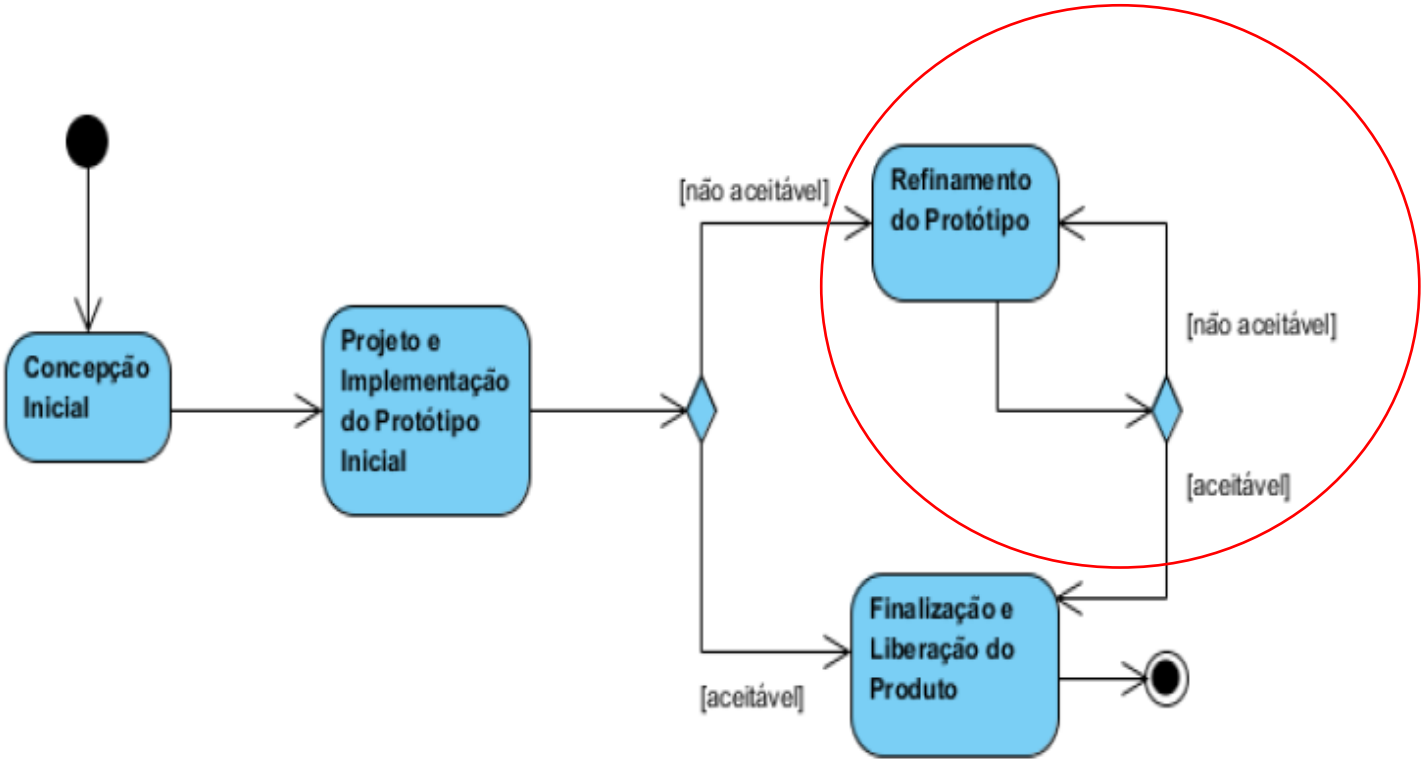
Modelo incremental

- Combina fluxos de processo linear e paralelo.
- Pode haver uma necessidade de fornecer rapidamente um conjunto de funcionalidades de software para os usuários e, em seguida, refinar e expandir funcionalidade em versões posteriores do software.
- **Cada sequência produz incrementos entregáveis do software**
  - Frequentemente o primeiro incremento contem as funcionalidades básicas ou mais importantes (*core product*).
  - O cliente avalia o incremento.
  - Planeja-se o próximo incremento.
  - Por exemplo, um software editor de texto poderá entregar primeiramente os recursos mais básicos de edição e salvamento. Posteriormente seriam entregues as funções mais avançadas.

Modelos evolucionários

- O software evolui ao longo de um período de tempo.
- Requisitos do negócio e do produto muitas vezes mudam conforme o desenvolvimento prossegue.
- Prazos apertados de mercado tornam a conclusão de um produto de software abrangente impossível, mas uma **versão limitada deve ser introduzida** para atender à pressão competitiva.
- Um conjunto de requisitos do produto principal são bem compreendidos, mas os detalhes do produto ainda não foram definidos.
- Nessas situações, é necessário um modelo de processo que foi explicitamente projetado para acomodar um produto que cresce e muda.
- **Modelos evolutivos** permitem que se desenvolva versões cada vez mais completas do software.

Prototipação Evolucionária



Prototipação Evolucionária

- Protótipo é uma versão **incompleta** do software que está sendo desenvolvido.
- Existem duas abordagens de prototipação:
  - *Throw-away* (Descartável)
    - Criados unicamente para estudar aspectos do sistema, entender melhor os requisitos e avaliar riscos
    - Depois de cumprir sua finalidade, é descartado
  - *Cornerstone* (pedra fundamental)
    - Também são usados para estudar aspectos do sistema, entender melhor os requisitos e avaliar riscos
    - O protótipo fará parte do sistema, ou seja, vai evoluindo até se tornar um produto que pode ser entregue.

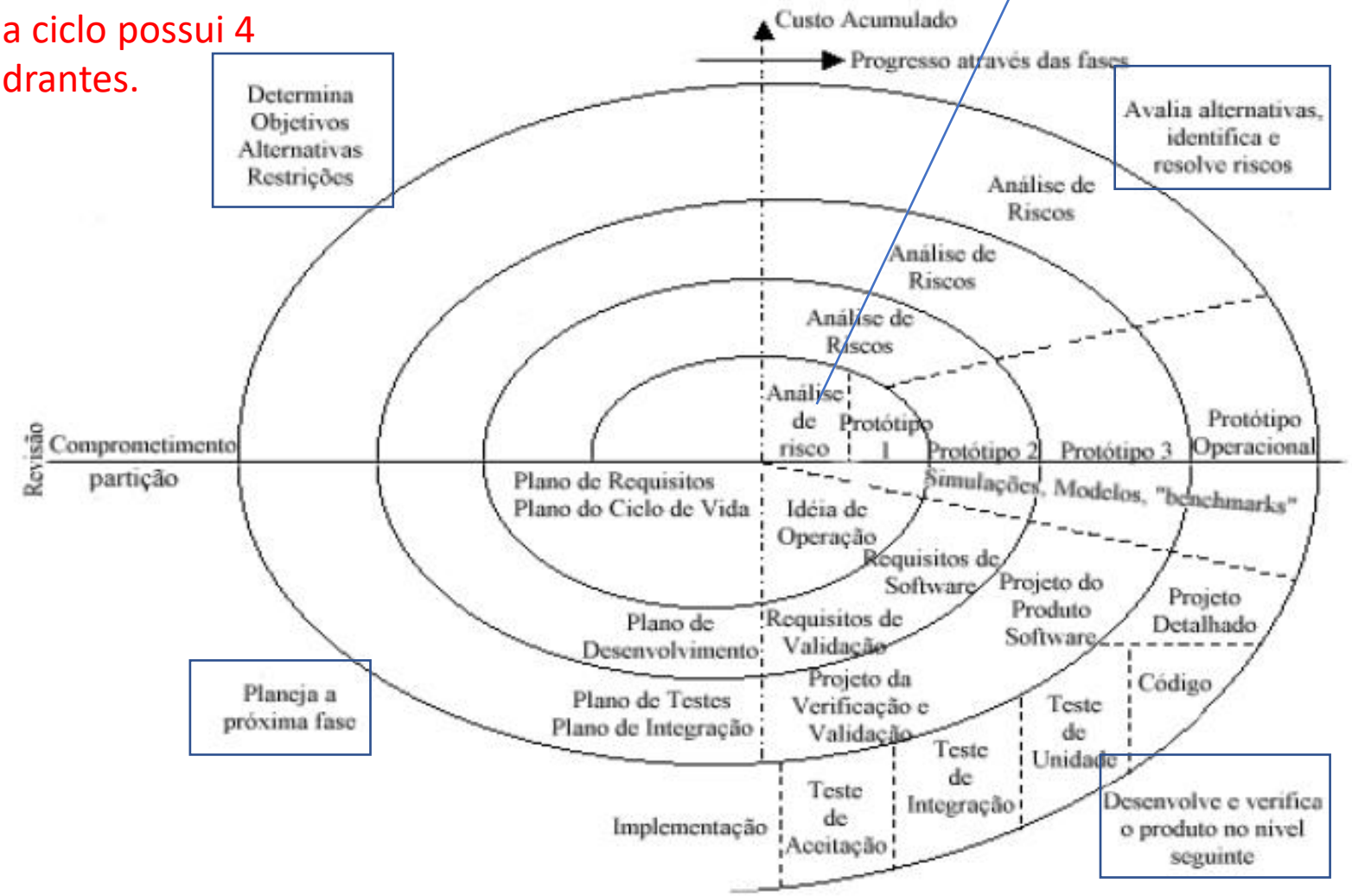
Prototipação Evolucionária

- Este modelo sugere que a equipe trabalhe junto com o cliente os aspectos mais visíveis (interface).
- Útil quando for difícil ao cliente comunicar seus requisitos.
- Útil quando nem o cliente e nem a equipe conhecem bem os requisitos.
  - Cliente só sabe especificar os requisitos em alto nível.
  - Equipe não tem certeza em relação ao uso de determinado algoritmo.
- Não é útil quando existe a necessidade de previsão de tempo de entrega.
- Embora a prototipagem possa ser usada como um modelo de processo autônomo, é mais comumente usado como uma técnica que pode ser implementada dentro do contexto de qualquer um dos modelos de processo.

Modelo Espiral

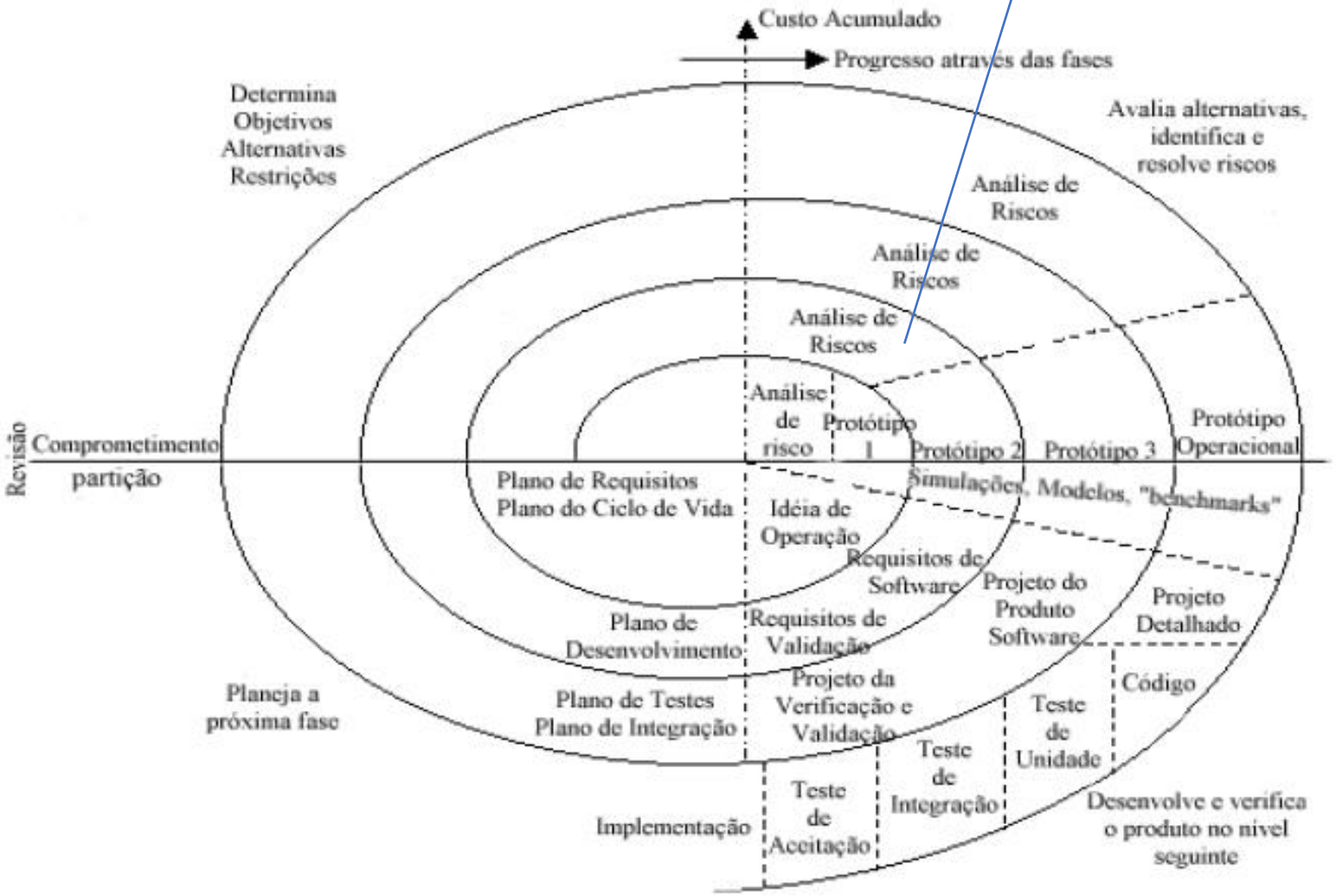
Ciclo mais interno é orientado às possibilidades do sistema

Cada ciclo possui 4 quadrantes.



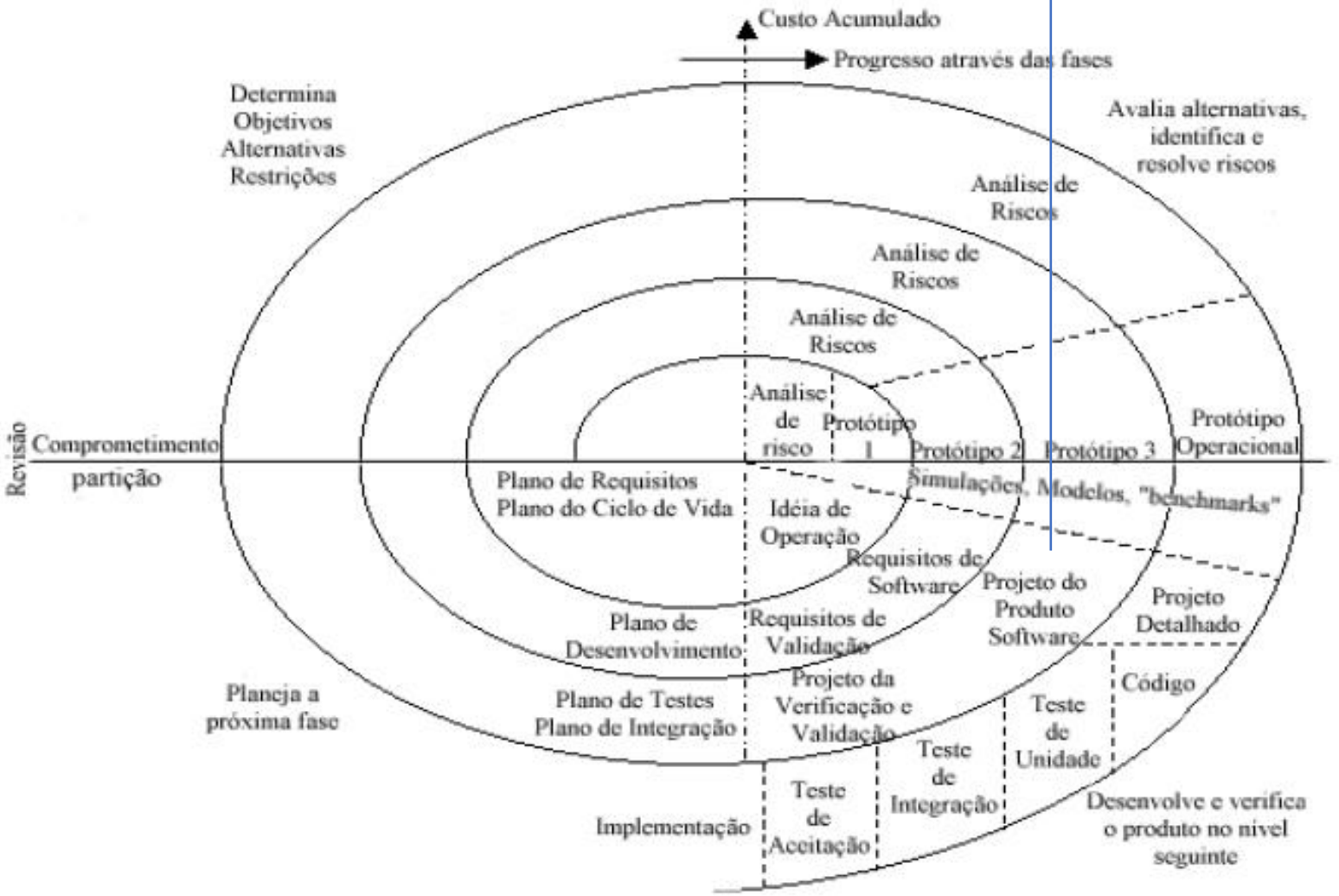
Modelo Espiral

o próximo ciclo está concentrado na definição dos requisitos do sistema



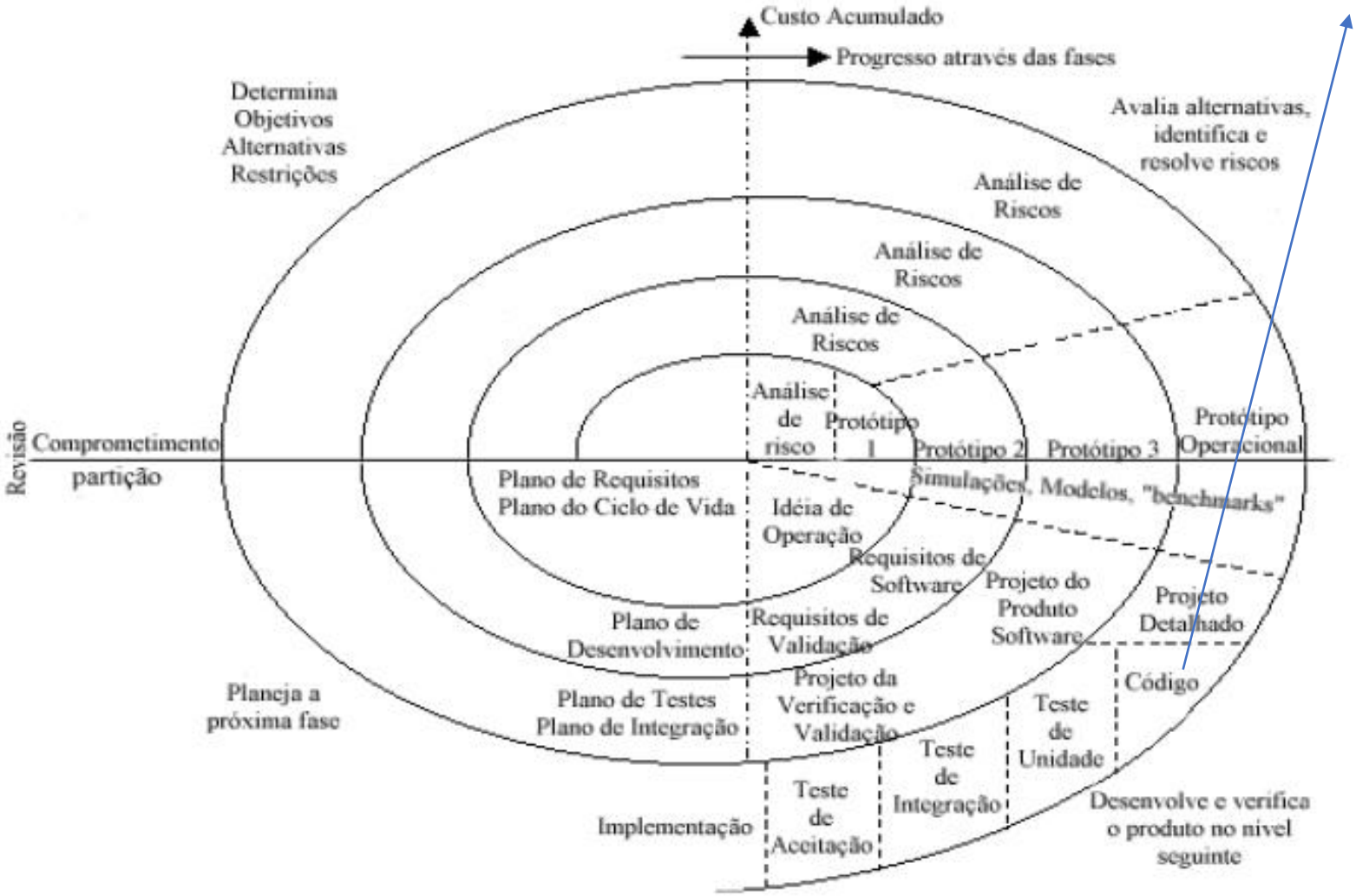
Modelo Espiral

orientado ao projeto



Modelo Espiral

Concentra-se na construção do sistema



Modelo Espiral

- Cada volta no ciclo (iteração) faz o projeto avançar um pouco mais.
- Inicia com pequenos **protótipos** e avança para um projeto maior.
- O cliente revisa a iteração atual e fornece feedback, que é analisado e usado para planejar a próxima iteração.
- Cada iteração envolve os seguintes passos:
  - Determinar os objetivos, alternativas e restrições relacionadas à iteração que vai começar;
  - Identificar e resolver riscos relacionados à iteração;
  - Avaliar as alternativas disponíveis. Aqui podem ser utilizados protótipos para verificar a viabilidade das alternativas;
  - Desenvolver artefatos relacionados à iteração e certificar que eles estão corretos (validação);
  - Planejar a próxima iteração.

Modelo Espiral

- É orientado a riscos
  - Risco pode significar requisitos mal compreendidos, problemas tecnológicos (arquitetura de hardware e software, etc.).
  - Em todas as etapas da iteração os possíveis riscos são analisados.
- Depois que os principais riscos são mitigados, o processo prossegue de forma semelhante ao Cascata.
- Indicado para:
  - projetos de larga escala.
  - quando os requisitos não estão claros no inicio do projeto.

Qual o modelo de processo mais adequado? (Waslawick,2013)

- Aquele que for mais adequado ao projeto e a equipe de desenvolvimento.
  - Projetos de software tem características diferentes.
  - Não há um modelo que seja melhor que o outro.

- Para ajudar na escolha a equipe deve fazer alguns questionamentos:

Qual o modelo mais adequado?

1. Quão bem os analistas e o cliente conhecem os requisitos do sistema?
   - Com requisitos estáveis pode-se trabalhar com modelos mais previsíveis como o **Cascada e o V.**
   - Com requisitos instáveis ou mal compreendidos pode-se trabalhar com **ciclo de redução de risco (espiral), prototipação e métodos ágeis.**
- Quanto planejamento é necessário ?
  - Modelos prescritivos costumam privilegiar o planejamento.
  - Modelos ágeis permitem um planejamento menos detalhado e uma maior adaptação às mudanças do projeto.

Quando usar os processos tradicionais?

Quando não usar os processos tradicionais?

Referências