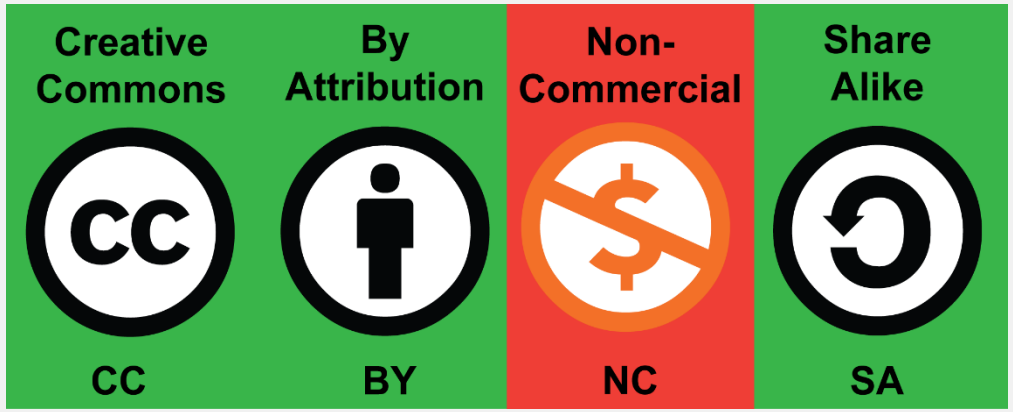
Ciclo de Vida de Sistemas (Software) / Processo de Software

Prof. MSc. Jackson Antonio do Prado Lima
jacksonpradolima at gmail.com / jackson.lima at udesc.br

Departamento de Sistemas de Informação – DSI

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Licença



UDESC

Histórico de Modificação

- Esta apresentação possui contribuição dos seguintes professores:
  - Silvia Regina Vergilio (UFPR)
  - Fabio Fernando Kobs (UDESC)
  - Jackson Antonio do Prado Lima (UDESC)

UDESC

Agenda

- Introdução
  - Processo de Software
- Ciclos de Vida / Modelos de Processo de Software
  - Clássico (Modelo Cascata)
  - Semi-Estruturado
  - Estruturado
  - Modelo Evolutivo
  - Modelo Incremental
  - Modelo Espiral
  - Prototipação
  - Linguagens de Quarta Geração
  - Desenvolvimento de Sistemas Formais
  - Desenvolvimento Baseado em Componentes

Introdução

- O que é processo de software?
  - Um conjunto estruturado de atividades requeridas para desenvolver um produto de software;
  - Envolve: fases, atividades, pessoas (responsáveis, participantes, papéis), entradas e saídas (artefatos), recursos, passos para execução, regras, procedimentos.

Introdução

- **Fases**: grandes divisões (marcos) de processos, períodos de tempo no qual algumas atividades são executadas;
- **Atividades**: possui entradas e saídas bem definidas, pessoas (responsáveis, participantes, papéis), entradas e saídas;
- **Artefatos**: qualquer documento que possa ser produzido ao longo do desenvolvimento de software;
- **Recursos**: software, hardware, material consumível ou não;
- **Passos para execução**: o que precisa ser feito em cada atividade.

Introdução

- Os desenvolvimentos de software são iniciados mediante o entendimento verbal entre usuário e gerente do projeto (papel);
- Envolve diversas fases/atividades:
  - Levantamento de requisitos
  - Detalhamento do projeto
  - Implementação
  - Testes
  - Instalação
  - Operação
  - Manutenção

Introdução

- Ao encadeamento dessas fases para a construção do sistema dá-se o nome de Modelo de Ciclo de Vida.
- Há diversos modelos, a diferença entre um e outro está na maneira como as fases são encadeadas;

Introdução

- Modelos de processo de software: representação abstrata de um processo de software;
- Projeto: execução de um processo.
- Padrão de processo: resolvem problemas recorrentes relacionados ao processo de software;
- Normas de avaliação de processos: NBR ISSO/IEC 12207

Introdução
Modelos de processo de software

- Objetivos:
  - Definir as atividades a serem executadas em um projeto de desenvolvimento de sistemas;
  - Introduzir consistência entre os projetos de desenvolvimento de sistemas da mesma organização;
  - Introduzir pontos de verificação para o controle gerencial de decisões: para determinar se está fora das previsões e se será preciso obter recursos adicionais.

Introdução

Modelos de processo de software

- Apresentam uma descrição de um processo de alguma perspectiva particular
- **Modelos prescritivos**: definem uma ordem para realizar as atividades e um fluxo de trabalho previsível
- **Modelos ágeis**: planejamento é gradativo, mais fácil adaptação

Ciclo de Vida do Projeto Clássico - Modelo Cascata

CICLO DE VIDA CLÁSSICO

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Ciclo de Vida Clássico

- Modelo mais antigo e o mais amplamente conhecido da Engenharia de Software
- Modelado em função do ciclo da engenharia convencional
- Requer uma abordagem sistemática, sequencial ao desenvolvimento de software
- O resultado de uma fase é entrada para outra fase
- Existem variações/adaptações

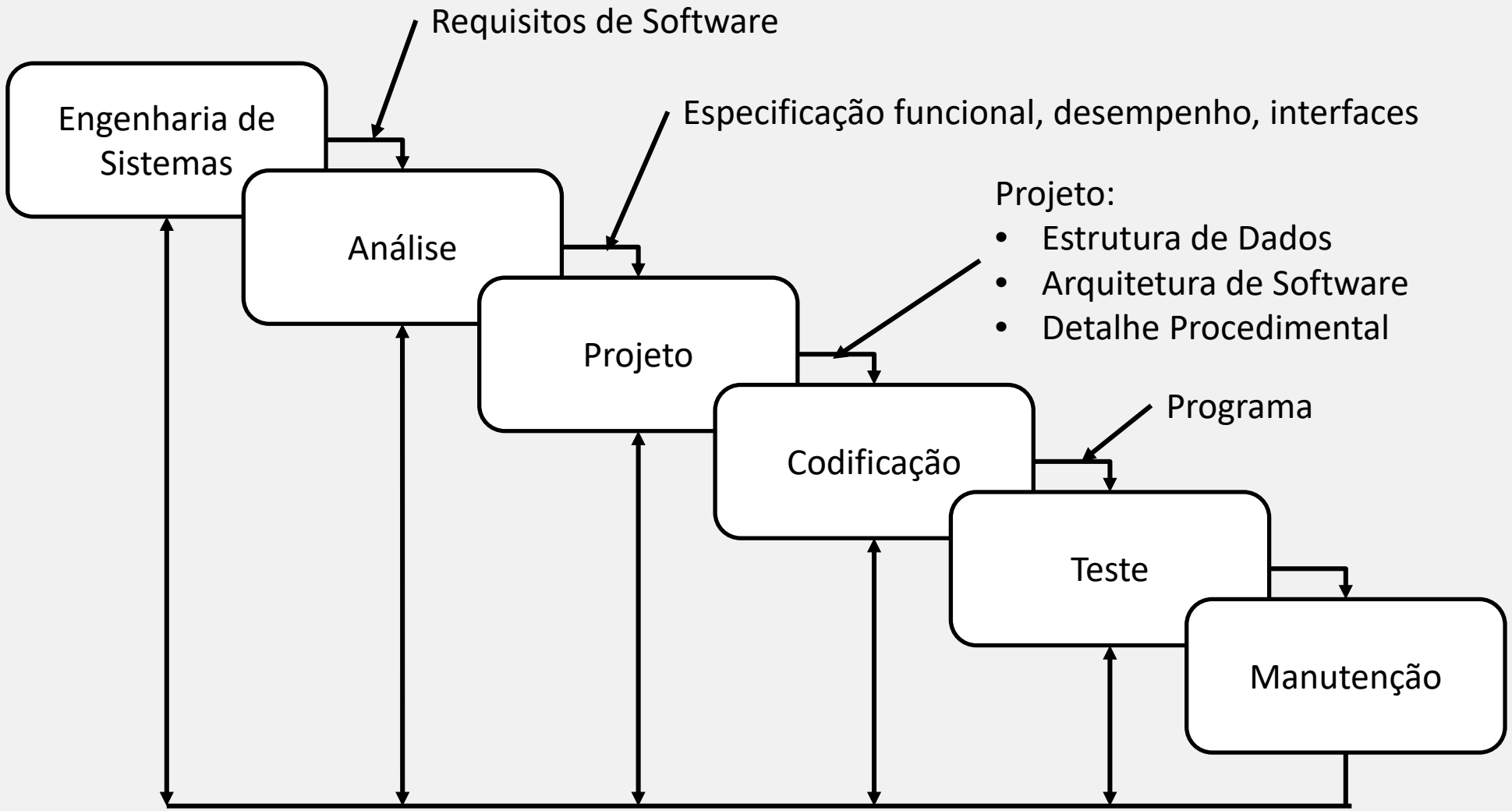
Ciclo de Vida Clássico

- O modelo 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

Ciclo de Vida Clássico

- O que caracteriza o ciclo de vida de um projeto como clássico é a forte tendência à implementação *bottom-up* do sistema e a insistência na progressão linear e sequencial entre uma fase e a seguinte.
  - Espera-se que os programadores executem todos os testes de seus módulos primeiros, em seguida o teste de subsistemas e, por fim, o teste do sistema

Ciclo de Vida Clássico



Ciclo de Vida Clássico

- **Engenharia de Sistemas**
  - Envolve a coleta de requisitos em nível do sistema, com uma pequena quantidade de projeto e análise de alto nível
  - Esta visão é essencial quando o software deve fazer interface com outros elementos(hardware, pessoas e banco de dados)
- **Análise de Requisitos de Software**
  - O processo de coleta dos requisitos é intensificado e concentrado especificamente no software
  - Deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos

Ciclo de Vida Clássico

- **Projeto**
  - Tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie
- **Codificação**
  - Tradução das representações do projeto para uma linguagem “artificial” resultando em instruções executáveis pelo computador

Ciclo de Vida Clássico

- **Teste** 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.
- **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*

Ciclo de Vida Clássico

Dificuldades

a) **Nada está terminado até que esteja totalmente pronto**. Exemplo: se o projeto estiver atrasado e o prazo final coincidir com o meio do teste de sistema, não haverá nada para mostrar ao usuário;

b) **Os erros mais triviais são encontrados no início do período de teste e os erros mais sérios são encontrados no final.** Os mais importantes erros não são os que o programador deseja encontrar no final de um projeto em desenvolvimento. Estes erros podem levar a modificação de muitos módulos;

Ciclo de Vida Clássico

Dificuldades

c) **A depuração de erros é muito difícil durante os estágios finais dos testes do sistema**. Pela depuração descobre-se onde está o erro (e determina-se como corrigi-lo) depois do processo de teste ter revelado que existe um erro. Quando um erro é descoberto durante a fase de teste de sistema é muito difícil definir qual o módulo que contém o erro;

d) **As necessidades de tempo de computador para testes aumentam durante os estágios finais dos testes**. Assim, o projeto poderá sofrer um atraso.

CICLO DE VIDA DO PROJETO SEMI-ESTRUTURADO

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

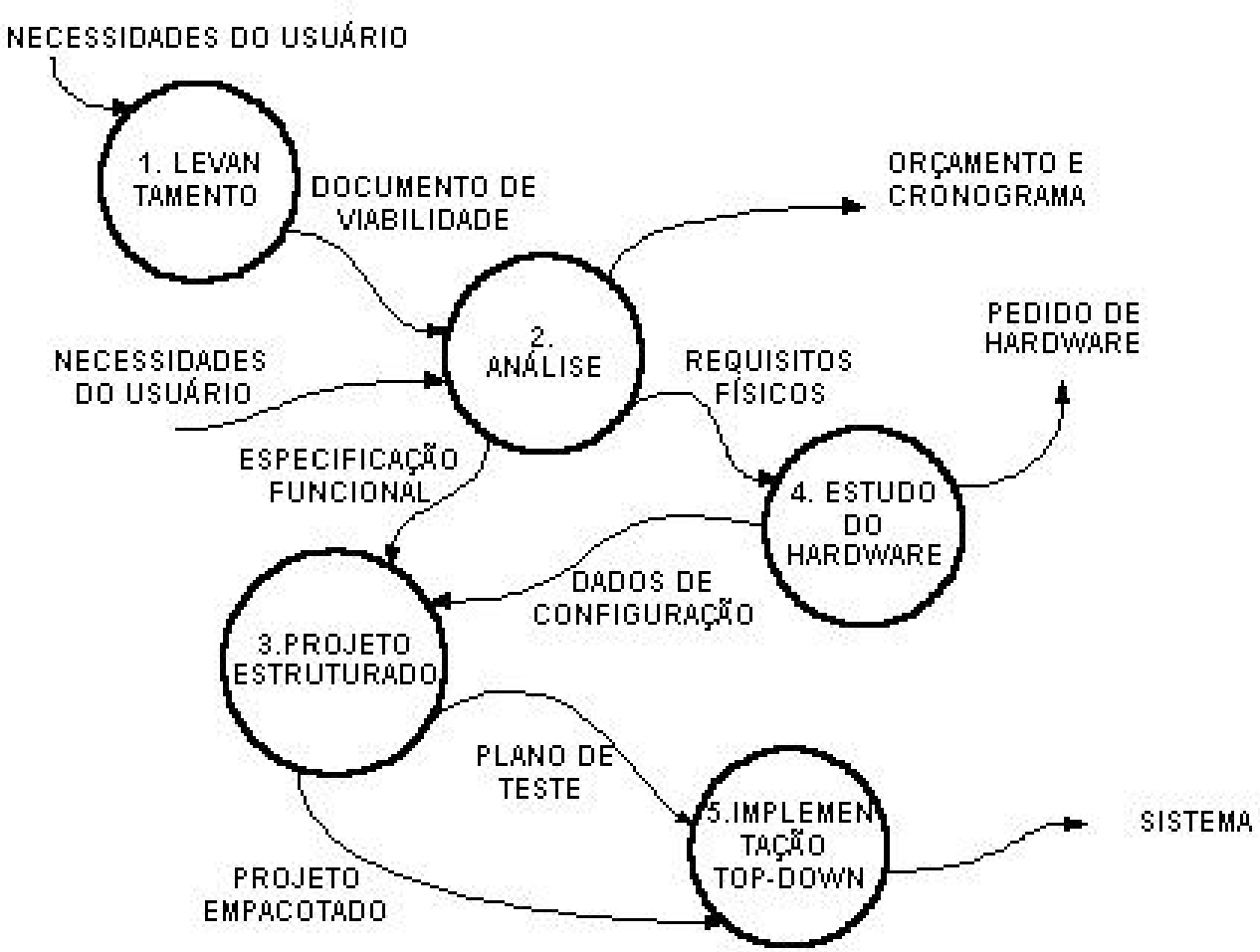
Ciclo de Vida do Projeto Semi-Estruturado

- Começando no final da década de 70 e início da década de 80, as técnicas como projeto estruturado, programação estruturada e implementação *top-down* foram reconhecidas.

Ciclo de Vida do Projeto Semi-Estruturado

Características

- Uma abordagem onde os módulos de alto nível são codificados e testados em primeiro lugar, seguidos pelos módulos detalhados de baixo nível. A programação estruturada será usada como método de codificação;
- O projeto clássico é substituído pelo projeto semi-estruturado. A implementação *top-down* significa que alguma codificação e alguns testes ocorrem paralelamente;
- Há um afastamento das fases sequenciais (ciclo de vida clássica) e uma realimentação entre a atividade de codificação e de testes, e a de depuração de erros.

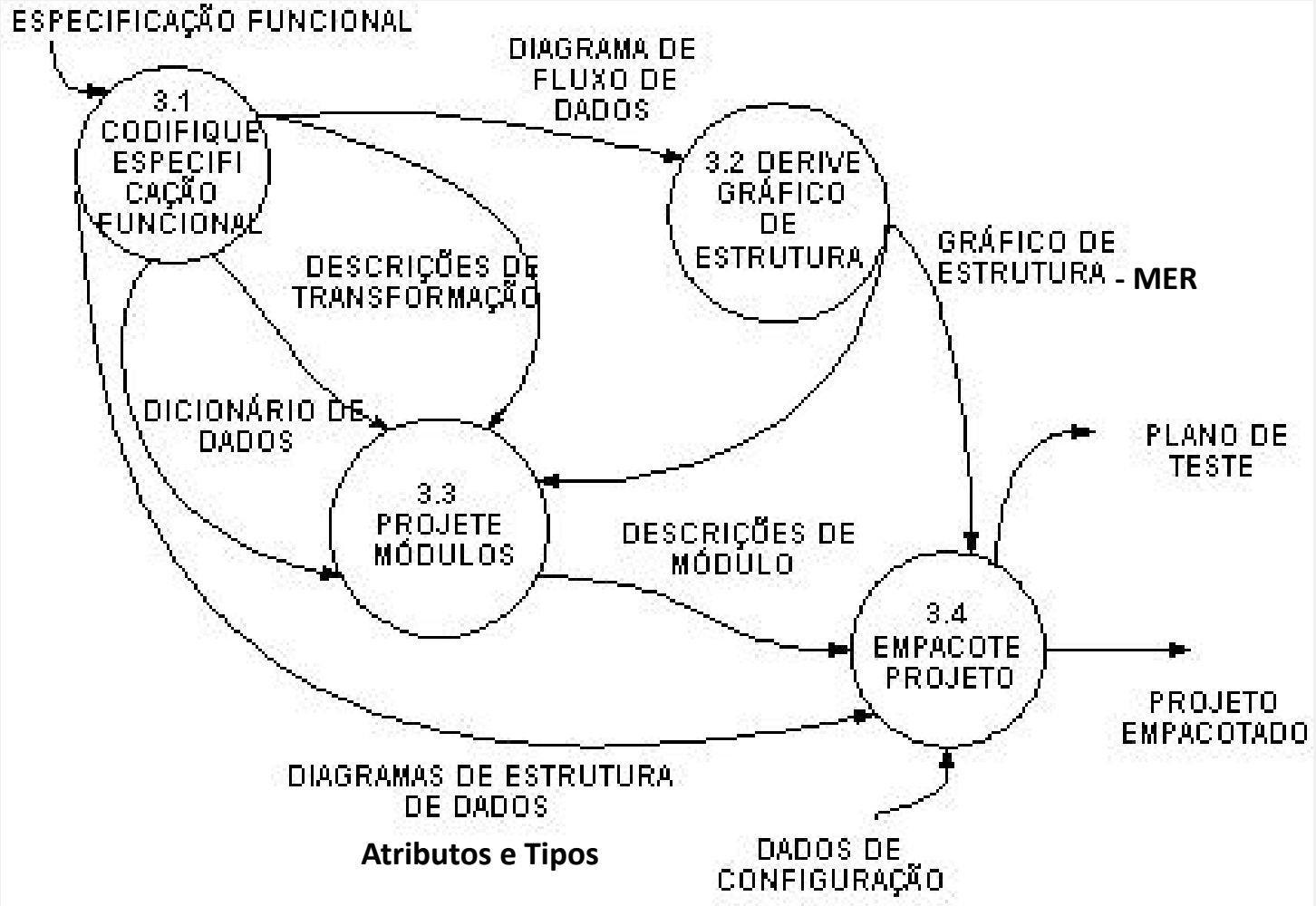


Ciclo de Vida do Projeto Semi-Estruturado

Fases

Ciclo de Vida do Projeto Semi-Estruturado

Uma parte significativa do trabalho ocorre sob o título da fase 3 “projeto estruturado”, que tem como objetivo corrigir especificações narrativas mal feitas.



CICLO DE VIDA DO PROJETO ESTRUTURADO

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

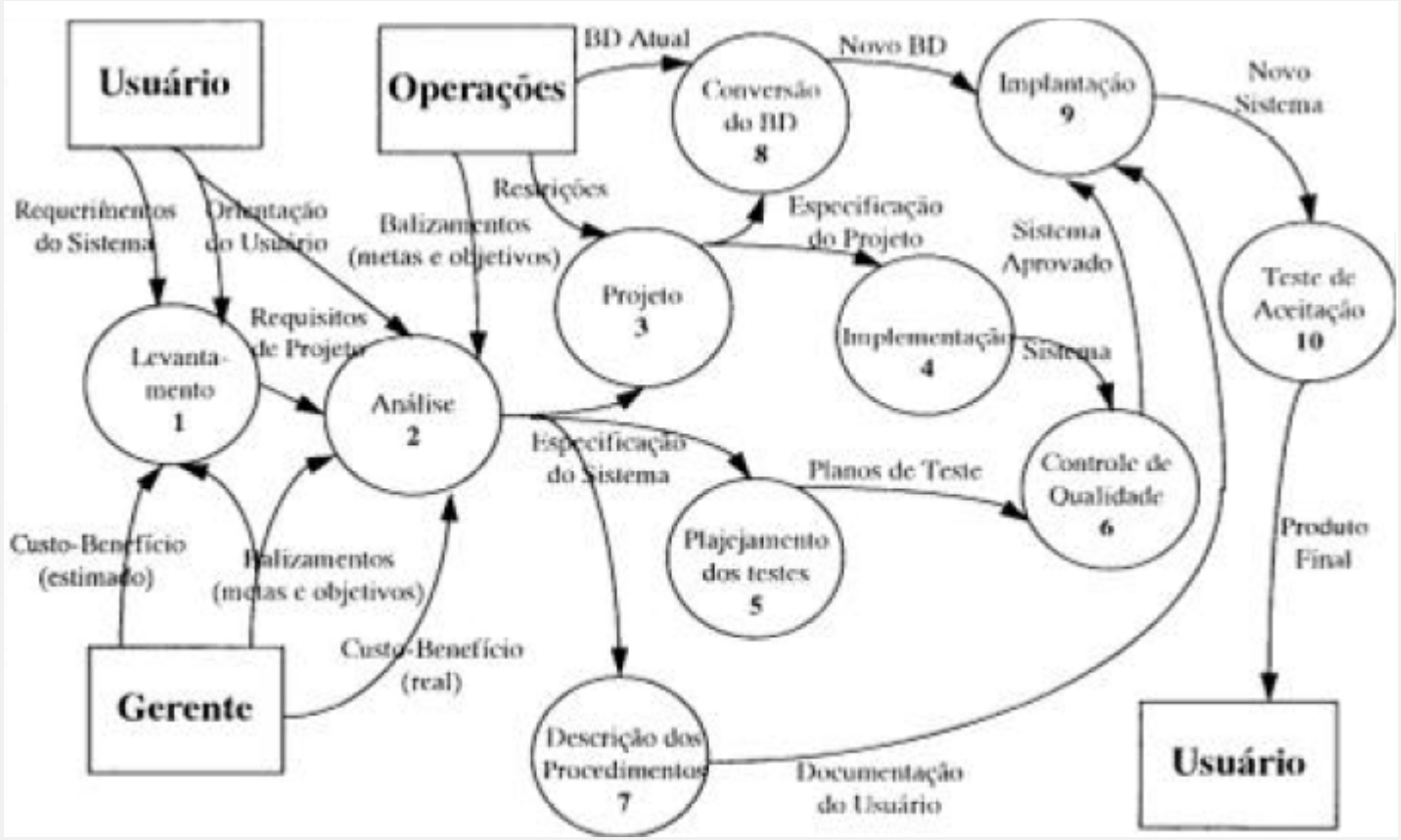
Ciclo de Vida do Projeto Estruturado

- São nove atividades e três terminais do ciclo de vida do projeto.
- Os terminais consistem em usuários, gerentes e pessoal da operação. São indivíduos ou grupos de indivíduos que fornecem entradas à equipe do projeto e que são os últimos receptores do sistema. Eles interagem com as nove atividades.

Ciclo de Vida do Projeto Estruturado

Atividades:

1. Levantamento
2. Análise
3. Projeto
4. Implementação
5. Testes
6. Controle de Qualidade
7. Descrição dos Procedimentos
8. Conversão do Banco de Dados
9. Instalação



Ciclo de Vida do Projeto Estruturado

1. Levantamento

- Também conhecido como estudo de viabilidade de estudo inicial das atividades. Começa quando um usuário solicita que uma ou mais partes de sua atividade sejam automatizadas.

Principais objetivos:

a) **Identificar os usuários responsáveis e desenvolver um "escopo" inicial do sistema**. Pode envolver o desenvolvimento de um simples diagrama de contexto inicial ou um simples diagrama de fluxo de dados no qual o sistema inteiro está representado por um único processo;

Ciclo de Vida do Projeto Estruturado

1. Levantamento

b) **Identificar as atuais deficiências no ambiente do usuário**: lista das funções que estejam faltando ou que estejam atuando de modo insatisfatório;

c) **Estabelecer metas e objetivos para um novo sistema**: lista de funções que necessitam ser reimplementadas, ou novas funções a serem acrescentadas, e critérios de desempenho para o novo sistema;

d) **Determinar se é possível automatizar o sistema e sugerir alguns esquemas aceitáveis**. Algumas estimativas do cronograma e do custo de construção de um novo sistema e dos benefícios podem ser elencados;

e) **Preparar uma previsão do projeto que será usada para conduzir o restante do projeto**. Incluirá a informação listada acima, identificará o gerente responsável do projeto e pode descrever os detalhes do ciclo de vida que o resto do projeto seguirá (cronograma).

Ciclo de Vida do Projeto Estruturado

2. Análise

- Principal objetivo é transformar as suas duas principais entradas, critério do usuário e previsão do projeto, em uma especificação estruturada;
- Envolve a modelagem do ambiente do usuário com diagramas de fluxo de dados, diagramas de entidades-relacionamentos, diagramas de transições de estado e outras ferramentas;
- Envolve o desenvolvimento de um modelo ambiental e o desenvolvimento de um modelo comportamental;

Ciclo de Vida do Projeto Estruturado

2. Análise

- Esses dois modelos se combinam para formar o modelo essencial que representa uma descrição do que o novo sistema deve fazer, independente da natureza da tecnologia a ser usada;
- Um detalhado conjunto de orçamentos e cálculos de custo/benefício é preparado, geralmente ao final da atividade de análise;
- Nessa atividade ocorrerá maior gasto de tempo.

Ciclo de Vida do Projeto Estruturado

3. Projeto

- Alocação de partes da especificação com o desenvolvimento de uma hierarquia apropriada de módulos de programa e interfaces entre os módulos, para implementar a especificação criada na atividade com a transformação de modelos de dados de ER em um projeto de banco de dados.

Ciclo de Vida do Projeto Estruturado

4. Implementação

- Codificação e integração de módulos;
- Inclui a programação estruturada e a implementação *top-down*;
- A programação e os testes normalmente começam quando termina a atividade de projeto;
- A programação envolve a escrita de instruções em alguma linguagem de programação para implementar o que o analista especificou e o que o projetista organizou em módulos;

Ciclo de Vida do Projeto Estruturado

4. Implementação

- Os testes envolvem a experimentação do sistema para ver se ele produz as saídas corretas e apresenta o comportamento correto para um grande número de entradas;
- Alguns projetos (em algumas organizações) a análise de sistemas, o projeto e a implementação são feitos pela mesma pessoa (analista de sistemas).

Ciclo de Vida do Projeto Estruturado

5. Testes

- A especificação estruturada deve conter as informações necessárias para definir um sistema aceitável do ponto de vista do usuário;
- O processo de desenvolvimento de casos de testes de aceitação pode ser executado em paralelo com as atividades de implementação de projeto e programação, de modo que, quando os programadores tiverem terminado seus programas e executado seus próprios testes locais, a equipe usuário/analista estará pronta para seus próprios casos de testes.

Ciclo de Vida do Projeto Estruturado

5. Testes

- Existem diferentes estratégias de testes: as duas mais conhecidas são os testes *bottom-up* e os *top-down*.
- A *bottom-up* começa por testar os módulos pequenos de forma individual, também chamado teste de unidade.
- Já a *top-down* presume que os módulos de execução de alto nível do sistema foram desenvolvidos. A abordagem *top-down* de testes é geralmente considerada melhor para a maioria dos sistemas.

Ciclo de Vida do Projeto Estruturado

5. Testes

Além dos testes básicos, existem os tipos de testes:

- **Testes funcionais**: Tem como objetivo, verificar se o sistema executa corretamente suas funções normais.
- **Testes de recuperação**: Verifica se o sistema pode recuperar-se adequadamente de vários tipos de falhas. Essencial para sistemas que controlam dispositivos físicos e/ou processos de fabricação.
- **Testes de desempenho**: O objetivo deste tipo de teste é verificar se o sistema pode manipular o volume de dados e transações recebidas específicas no modelo de implementação do usuário e verificar se apresenta o tempo de resposta necessário.
- **Testes completos**: No projeto ideal, geram-se casos de testes que cobrem todas as entradas possíveis e toda possível combinação de situações com que o sistema poderá se defrontar; então testa-se exaustivamente o sistema para assegurar que seu comportamento será perfeito.

Ciclo de Vida do Projeto Estruturado

6. Controle de Qualidade

- Conhecido também como teste final ou teste de aceitação;
- O analista pode estar envolvido na atividade de controle de qualidade, mas isso normalmente não acontece;
- Um ou mais membros da organização usuária pode assumir a responsabilidade, ou ela pode ser executada por um grupo independente de testes ou pelo departamento de controle de qualidade;
- É uma atividade com o propósito de garantir que o sistema apresentará o nível apropriado de qualidade.

Ciclo de Vida do Projeto Estruturado

7. Descrição dos Procedimentos

- É a geração de uma descrição formal das partes do novo sistema que serão manuais, e de uma descrição de como os usuários vão interagir com a parte automatizada do novo sistema;
- A saída desta atividade é o manual do usuário.

Ciclo de Vida do Projeto Estruturado

8. Conversão do Banco de Dados

- Em alguns projetos, a conversão de banco de dados envolvia mais trabalho do que o desenvolvimento de programas para o novo sistema; em outros casos, poderia não haver qualquer banco de dados a ser convertido.

Ciclo de Vida do Projeto Estruturado

9. Instalação

- Suas entradas são o manual do usuário produzido pela atividade 7, o banco de dados convertido produzido pela atividade 8, e o sistema aprovado, produzido pela atividade 6.
- A instalação poderá ser um processo gradual, com um grupo de usuários após o outro recebendo os manuais do usuário, e sendo treinado no uso do novo sistema para finalmente começar a usá-lo.

Ciclo de Vida do Projeto Estruturado

Manutenção

- Alguns programadores permanecem para executar atividades de manutenção, mas, quando termina a fase de desenvolvimento, a maioria dos analistas, projetistas e programadores são deslocados para novos projetos;
- Algumas modificações que não causam impacto na especificação do sistema: a correção de um erro de um programa, a modificação do código para melhorar a legibilidade ou a eficiência do sistema ou uma mudança do *hardware* existente ou do *software* de sistema (o compilador, o sistema operacional, o sistema de gerenciamento de banco de dados etc.);

Ciclo de Vida do Projeto Estruturado

Manutenção

- Qualquer modificação feita em um sistema normalmente resulta em uma modificação no *software* e/ou no *hardware* do sistema; porém pode também causar a alteração dos manuais, dos procedimentos de operação e de vários outros componentes do sistema;
- O documento mais importante a ser mantido atualizado é a declaração de requisitos do usuário.

MODELO EVOLUTIVO

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

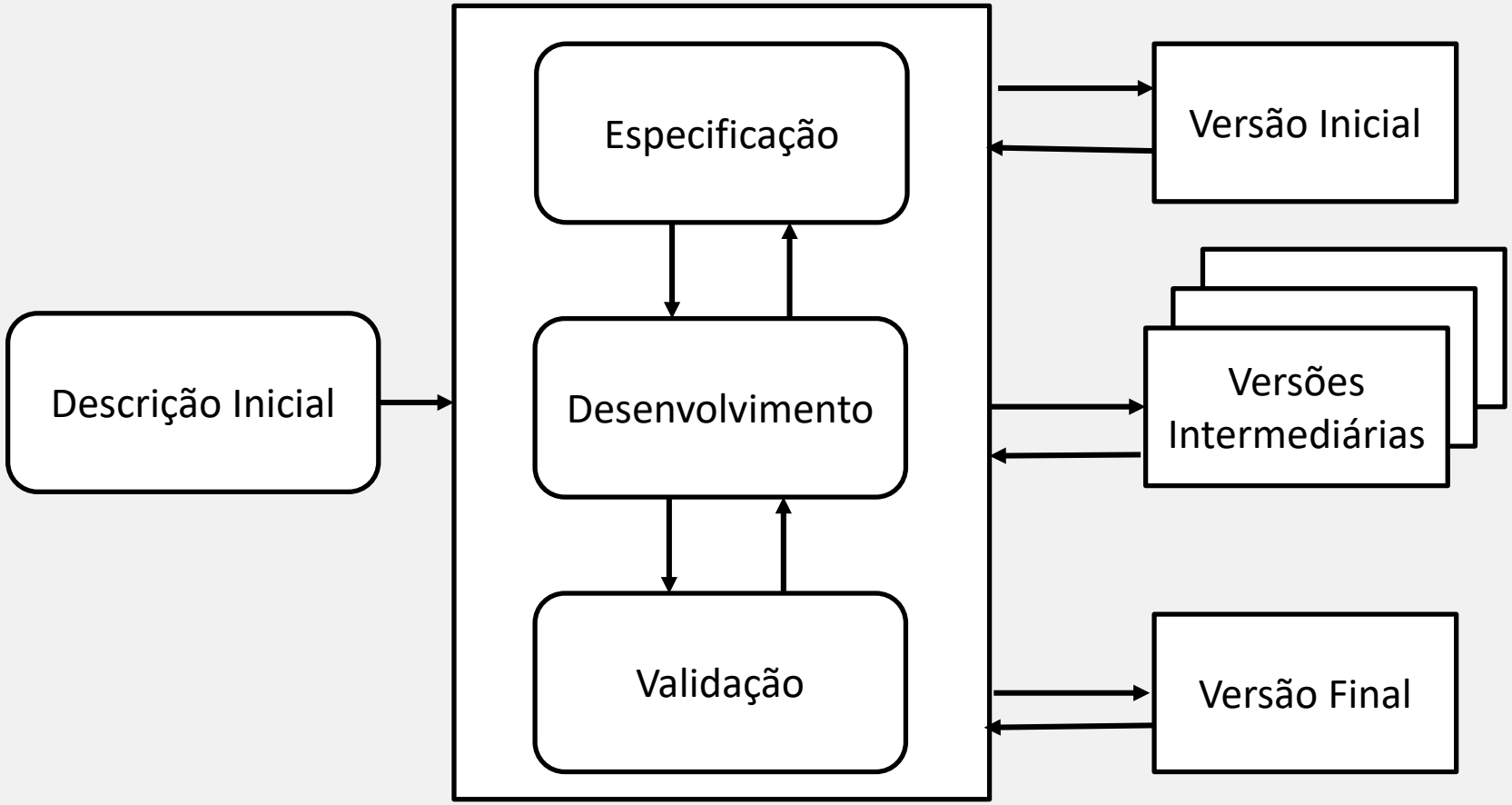
Modelo Evolutivo

- Diferentemente do modelo em cascata, o evolutivo considera que nenhum requisito é conhecido no momento do planejamento.
- Projeto faz entregas com base nas requisições conhecidas naquele momento, pede para o cliente usar o software em suas operações diárias e solicita feedbacks para criar novas versões.

Modelo Evolutivo

- Tudo merece uma nova chance – situações para acomodar um processo que evolui com o tempo
- Incorporação de diferente partes e criação de diferentes versões
- São iterativos – repetem atividades
- Inclui prototipação
- Permite o desenvolvimento exploratório

Modelo Evolutivo



MODELO INCREMENTAL

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Modelo Incremental

- Abordagem intermediária
- Combina vantagens dos paradigmas ciclo de vida clássico e evolutivo
- Identificação das funções do sistema, estabelecimento de incrementos e prioridades
- Cada incremento pode utilizar um paradigma de desenvolvimento diferente
- Dificuldade para dividir e gerenciar versões

Modelo Incremental

- O valor agregado ao Cliente está na entrega em cada incremento de modo que a funcionalidade do sistema estará disponível mais cedo
- Incrementos iniciais funcionam como protótipos para ajudar a evocar requisitos para incrementos posteriores
- Menores riscos de falha no projeto em geral
- Os serviços do sistema de alta prioridade tendem a receber a maioria dos testes

Modelo Incremental

- Nos primeiros anos de desenvolvimento de software, alguns projetos demoravam vários anos para serem entregues. Evidentemente que, atualmente, isso não é mais aceito devido ao mercado concorrido e a economia super aquecida do mundo globalizado.
- Segundo Pfleeger (2004), uma solução para reduzir o tempo de entrega do software foi passar a projetar o mesmo em pequenas partes e entregar para o cliente ao término de cada fase do projeto. Dessa forma, o usuário pode experimentar, mesmo que aos poucos, os recursos e funções que o software terá ao final do desenvolvimento.

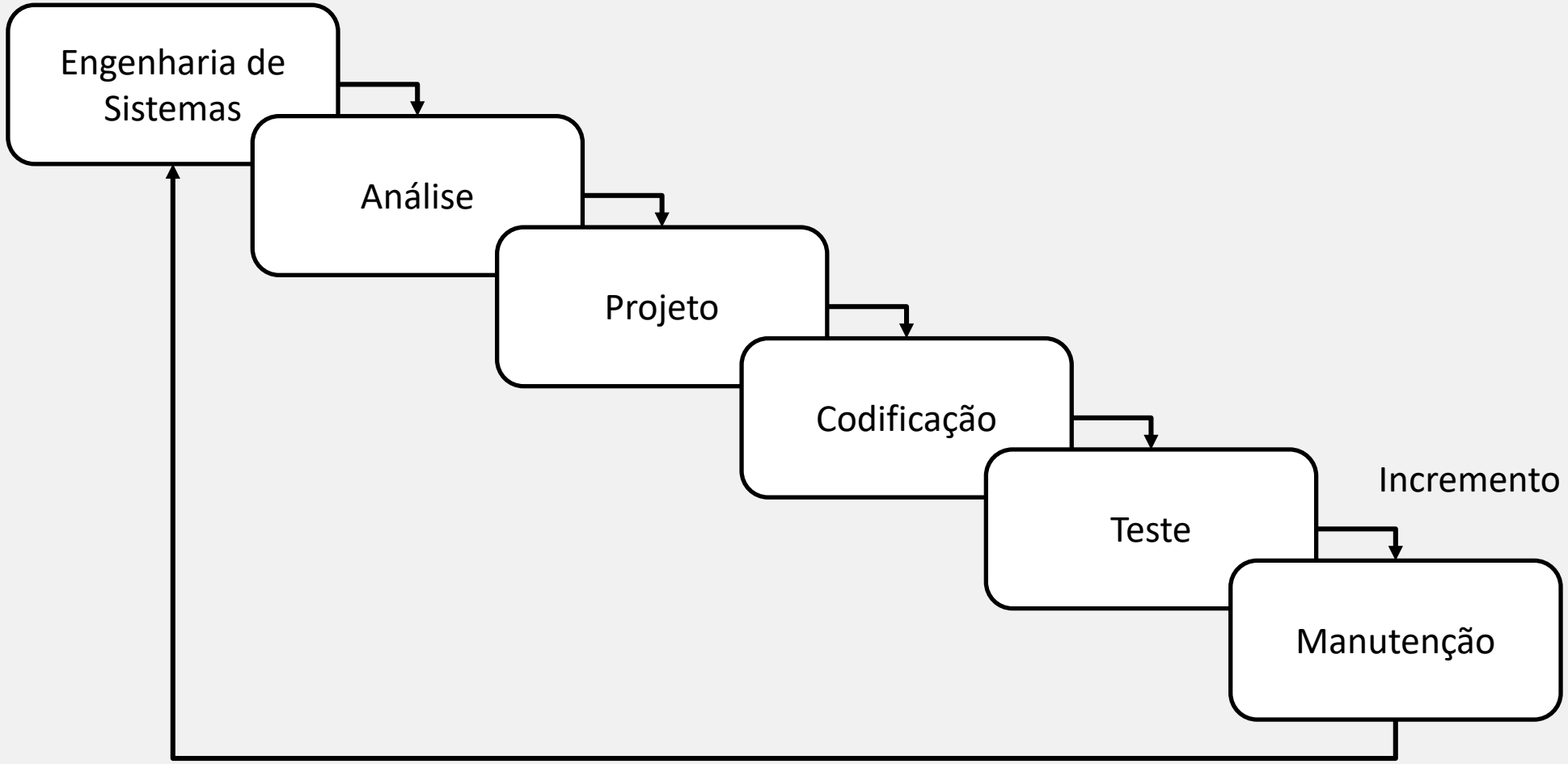
Modelo Incremental

- Na verdade, **o modelo incremental combina os elementos do modelo em cascata de forma iterativa**. Cada iteração é uma sequência linear que produz um subproduto de um sistema.
- O primeiro incremento é frequentemente chamado de núcleo de produto, que possui os requisitos básicos do sistema para que o cliente possa avaliar e elaborar os próximos incrementos que melhor satisfaçam as suas necessidades (PRESSMAN, 2006).

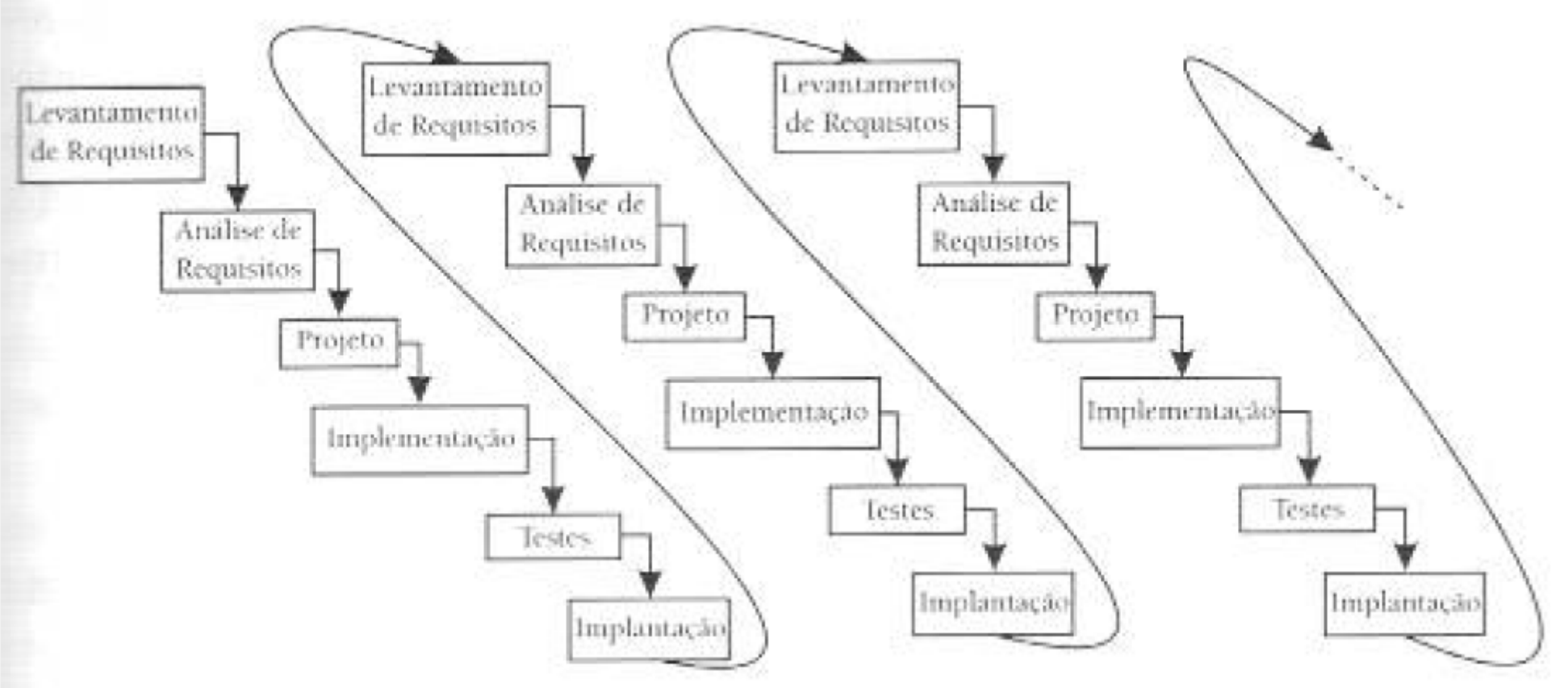
Modelo Incremental

- De acordo com Bezerra (2002) um processo de desenvolvimento segundo essa abordagem divide o desenvolvimento de um sistema em ciclos.
- Em cada ciclo – considera um subconjunto de requisitos – podem ser identificadas as fases de análise, projeto, implementação e testes.
- Assim o desenvolvimento evolui em versões, mediante a construção incremental e iterativa de novas funcionalidades até que o sistema completo esteja construído.
- Essa abordagem somente é possível se existir um mecanismo para dividir os requisitos do sistema em partes.

Modelo Incremental



Modelo Incremental



UDESC

Modelo Incremental

- O modelo incremental pode possui variações:
  - Orientado a funcionalidades: a cada versão novas funcionalidades são entregues
  - Orientado a cronograma: entrega em tempos determinados

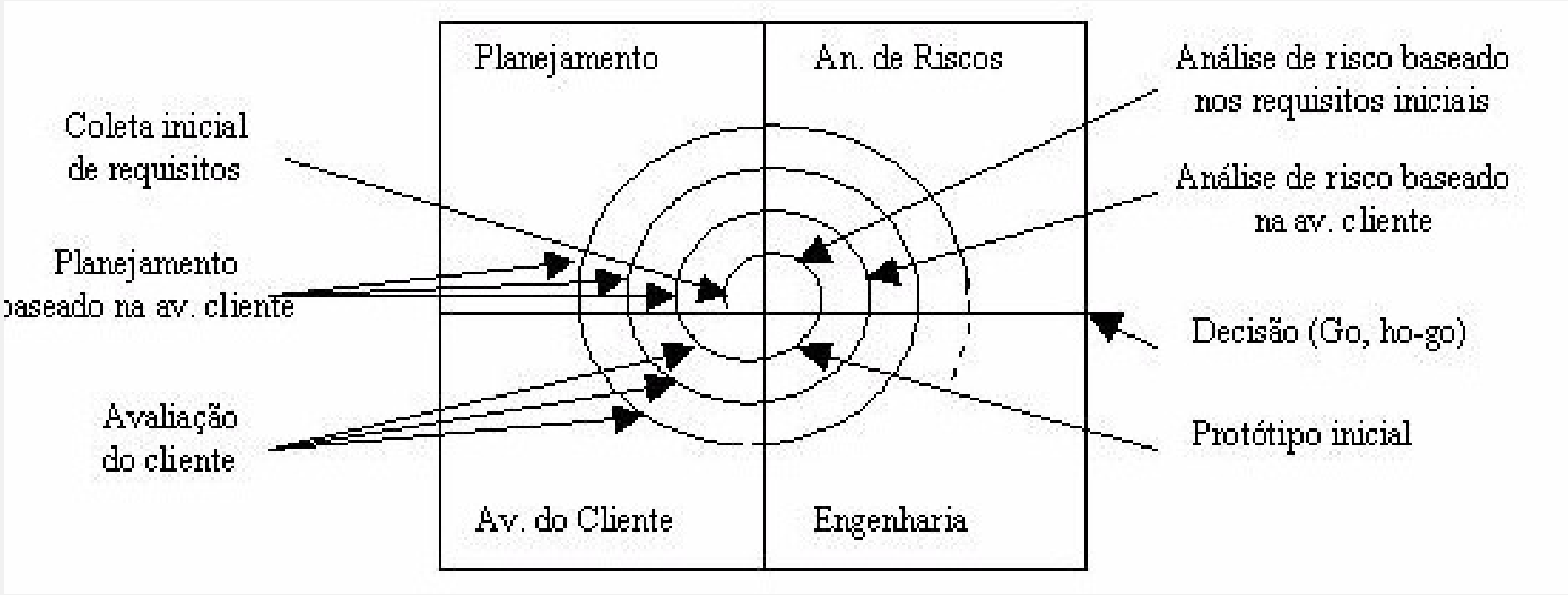
MODELO ESPIRAL

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Modelo Espiral

- Modelo que acopla a natureza iterativa da prototipação com os aspectos controlados e sistemáticos do modelo cascata
- Dividido em regiões ou atividades de trabalho.
- Cada volta do espiral representa um desenvolvimento completo.
- O loop mais interno se concentra mais na definição dos requisitos
- A gerência pode adaptar o modelo e suas fases

Modelo Espiral



Modelo Espiral

- Paradigma mais realístico - sistemas grandes
- É um metamodelo
- Incorpora análise de riscos.
- Permite prototipação em mais de um estágio
- Problemas: O modelo é relativamente novo.
- Requer esperteza. Pode nunca terminar, precisa ser evolutiva e controlável.

CONSIDERAÇÕES

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Considerações sobre processos

- Que processo usar?
  - Depende da natureza da aplicação.
  - Métodos e ferramentas disponíveis, etc.

Outros processos – objetivos específicos, mais especializados

- RAD (*Rapid Application Development*)- enfatiza rapidez e um ciclo mais curto
- Prototipação
- Processo formal
- Modelo de componentes
- Processo unificado – UP
- Orientados a reuso
- Linguagem de quarta geração
- Engenharia de Linha de Produto de Software (LPS)

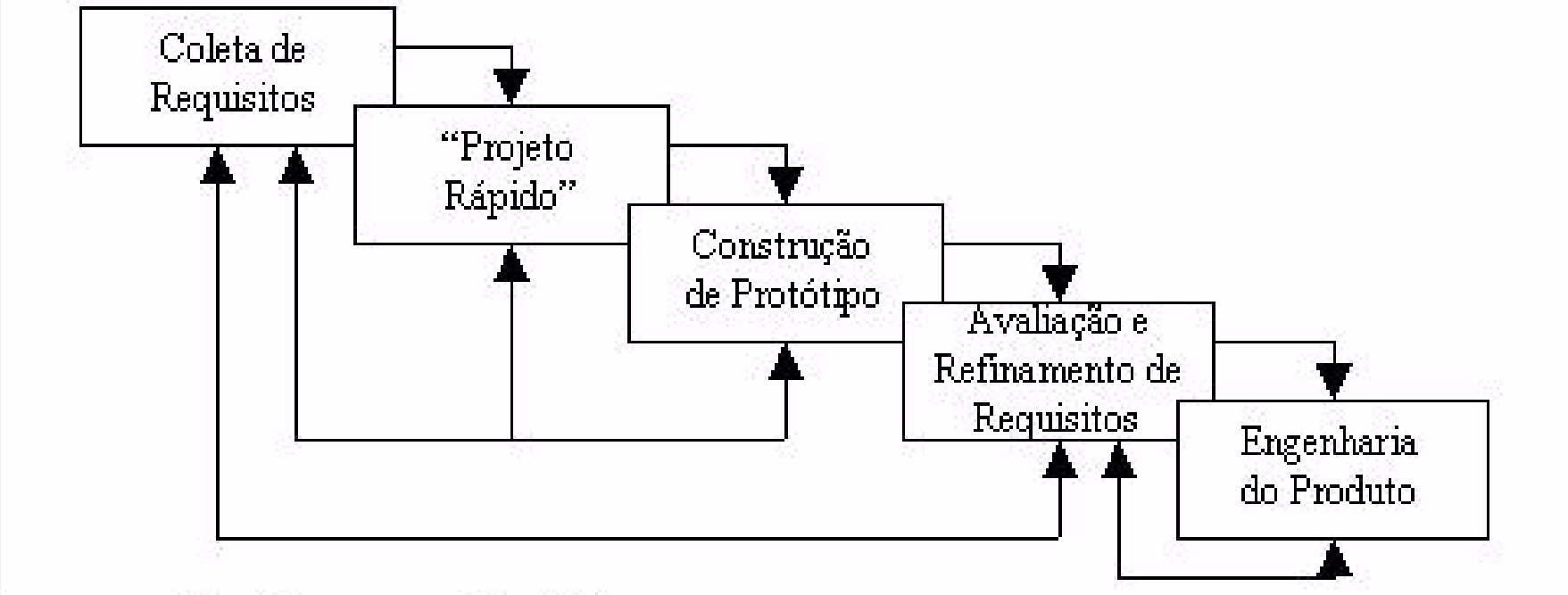
PROTOTIPAÇÃO

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

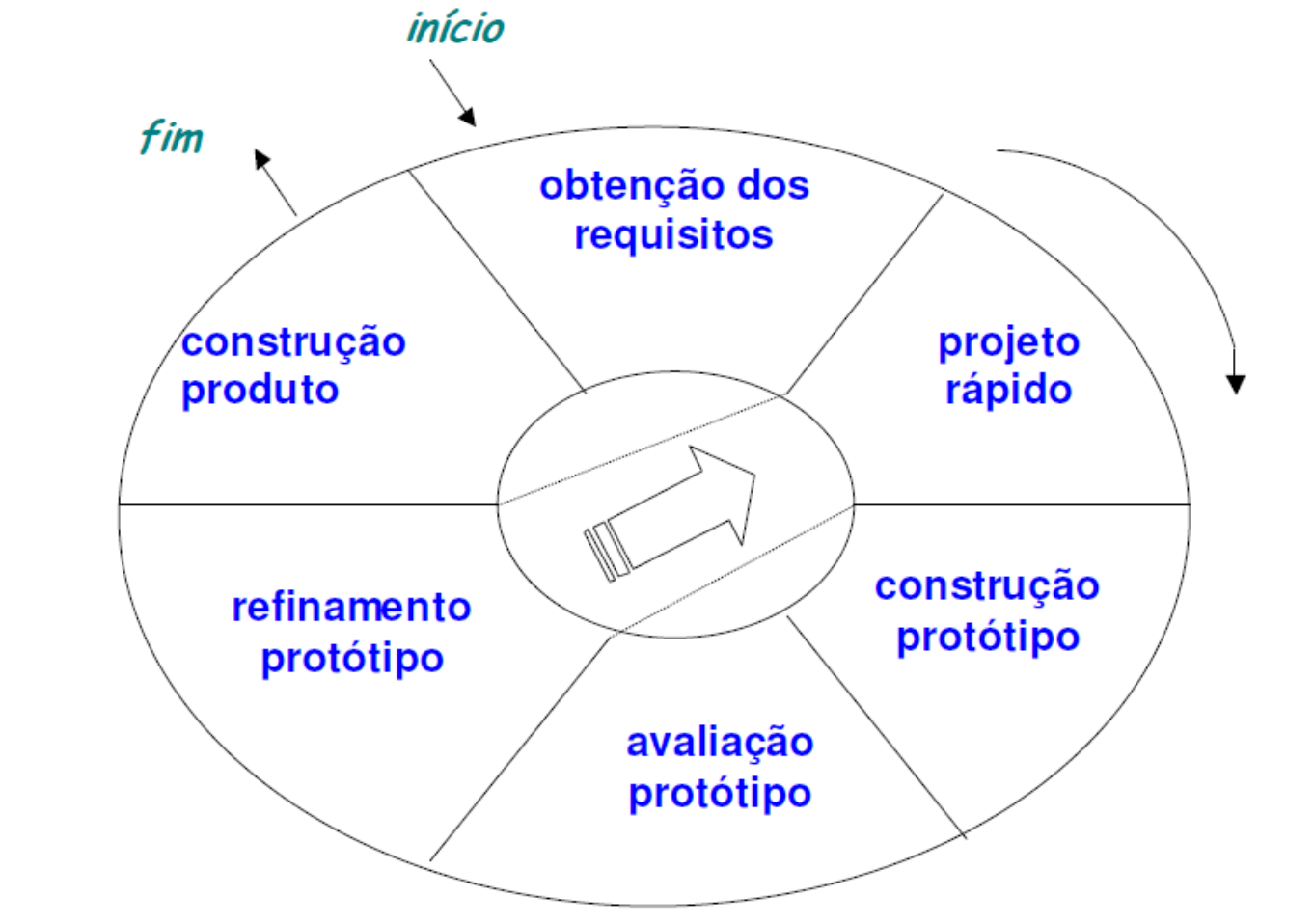
Prototipação

- Possibilita que o desenvolvedor crie um modelo (protótipo) do software a ser construído para auxiliar a entender os requisitos do usuário
- Apropriado para quando os requisitos não estão claramente definidos
- Derivado da técnica de extração de requisitos

Prototipação



Prototipação



UDESC

Prototipação
Abordagem

- Validar a precisão dos requisitos ou aceitabilidade das decisões
  - O usuário irá colaborar?
- Validar a viabilidade de uma estratégia proposta.
  - O software é particionável?
- Observações:
  - protótipos só são válidos se construídos rapidamente

Prototipação

- O protótipo pode ser descartado ou fazer parte do produto final.
  - C*onerstone* (pedra fundamental): Prototipação evolutiva
  - *Throw away*: descartável

Prototipação

- Localiza “aspectos visíveis” para o usuário (E/S).
- A iteração pode adequar o protótipo às necessidades do usuário.
- Problemas:
  - Cliente insiste que o protótipo seja com ligeiras modificações, a versão final do produto.
  - Decisões e soluções improvisados tornam-se parte do produto final.

LINGUAGENS DE QUARTA GERAÇÃO

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

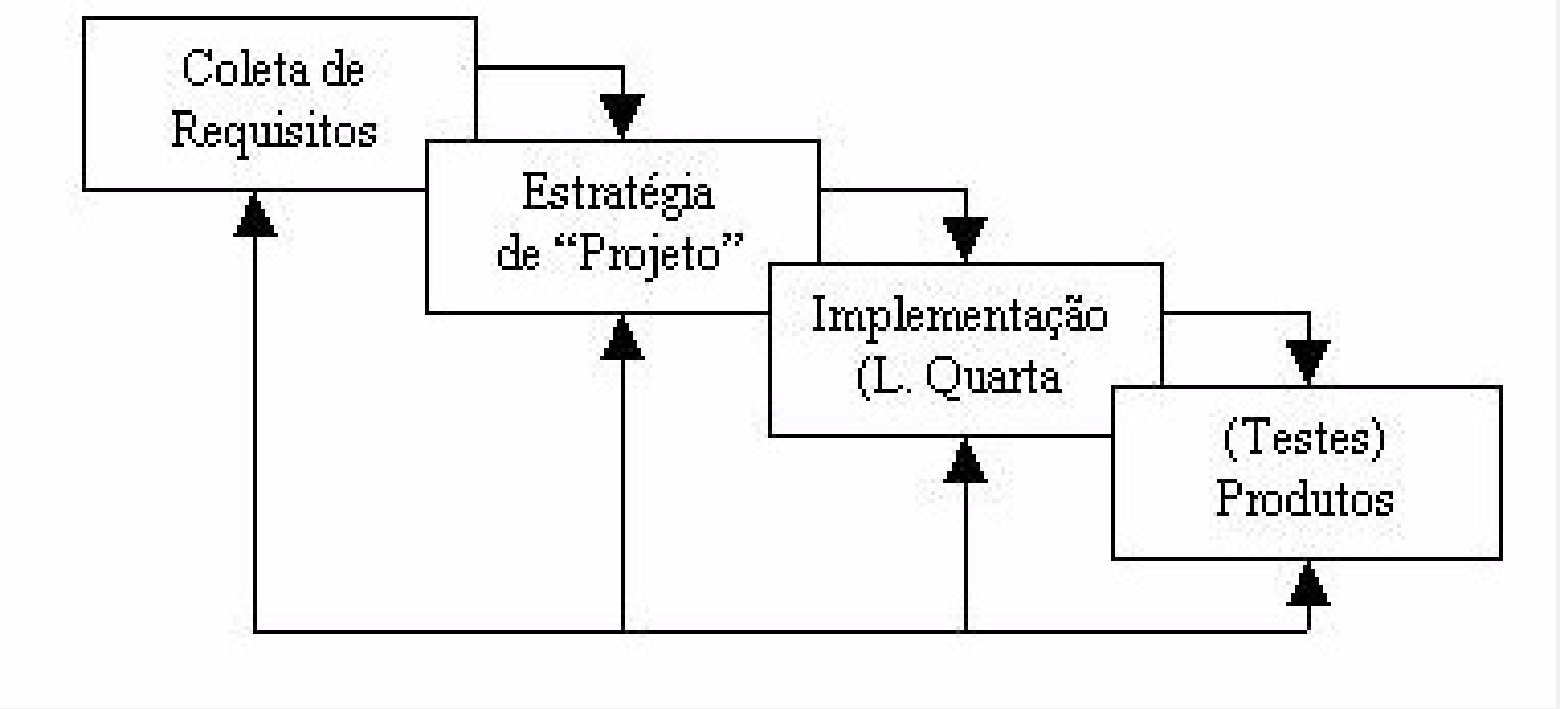
Linguagens de Quarta Geração

- Capacidade de se especificar software a uma máquina em um nível que esteja próximo à linguagem natural ou de se usar uma notação que comunique uma função significativa
- Permite ao programador especificar o que deve ser feito visando um resultado imediato
- Linguagens que geram programas em outras linguagens, como Java e C++.

Linguagens de Quarta Geração

- Ferramentas para especificação de alto nível (L4G):
  - Consulta a base de dados.
  - Geração de relatórios.
  - Manipulação de dados.
  - Definição e interação com Telas.
  - Geração de código.
- Exemplos de linguagens de 4ª geração:
  - SQL
  - Oracle Reports
  - MATLAB
  - PowerBuilder
  - Scilab
  - Strata

Linguagens de Quarta Geração



Linguagens de Quarta Geração

- Domínio predominante : Sistemas comerciais de informação.
- Boa produtividade para sistemas pequenos e médios e aplicação específicas.
- Problemas:
  - Para sistemas grandes, demanda muito tempo e;
  - Ainda permanece a necessidade de projeto

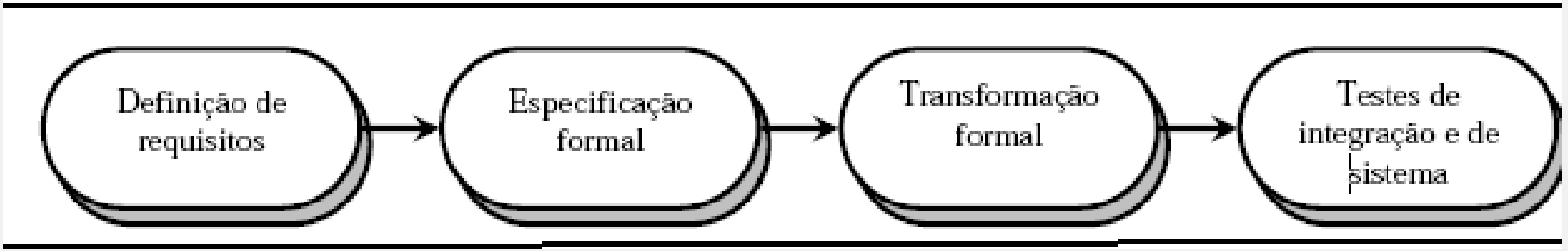
DESENVOLVIMENTO DE SISTEMAS FORMAIS

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Desenvolvimento de Sistemas Formais

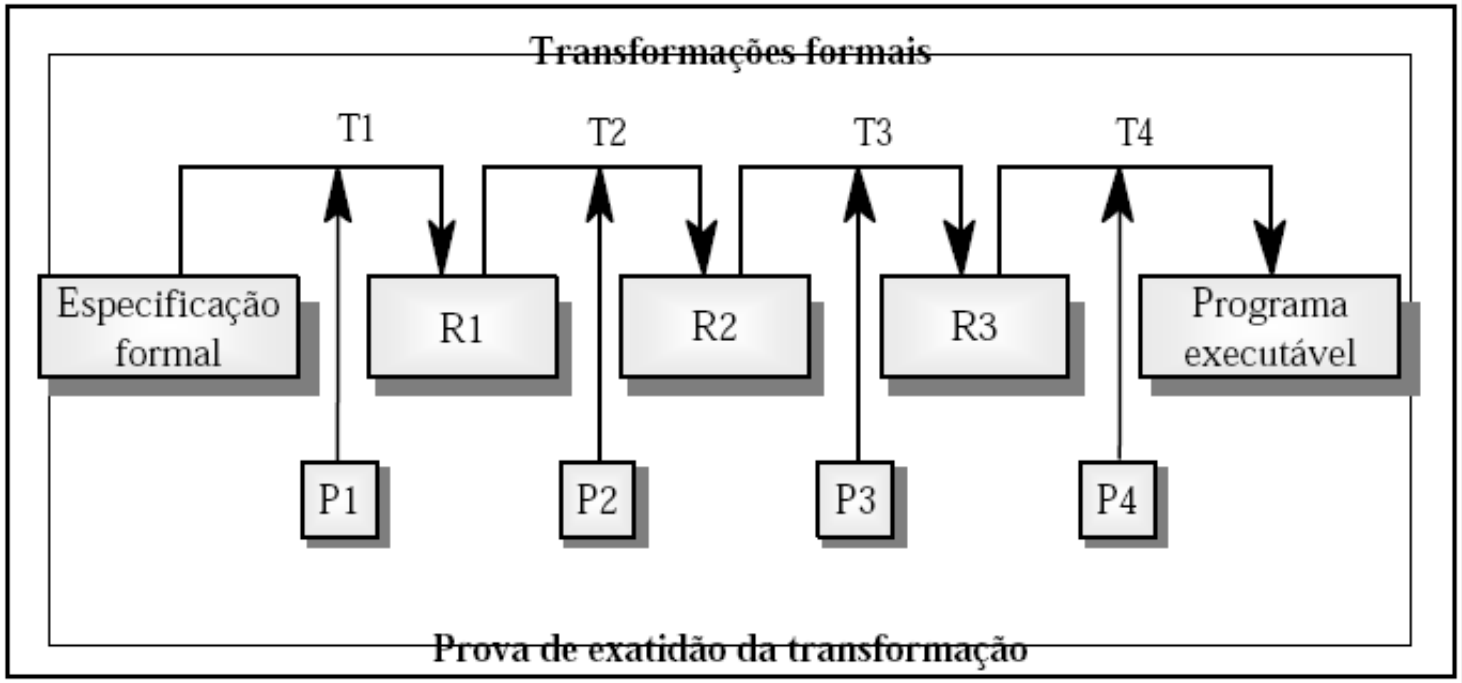
- Baseado na transformação de uma especificação matemática através de diferentes representações para um programa executável
- Transformações são ‘preservadoras de exatidão’, portanto, são diretas para mostrar que o programa está de acordo com sua especificação
- Contido na abordagem ‘Cleanroom’ para desenvolvimento de software

Desenvolvimento de Sistemas Formais



Desenvolvimento de Sistemas Formais

Transformações Formais



Desenvolvimento de Sistemas Formais

- Problemas
  - Necessidade de habilidades especializadas e treinamento para aplicar a técnica
  - Difícil de especificar formalmente alguns aspectos do sistema como a interface de usuário
- Aplicabilidade
  - Sistemas críticos, especialmente aqueles no qual um case de segurança deve ser feito antes do sistema ser posto em operação

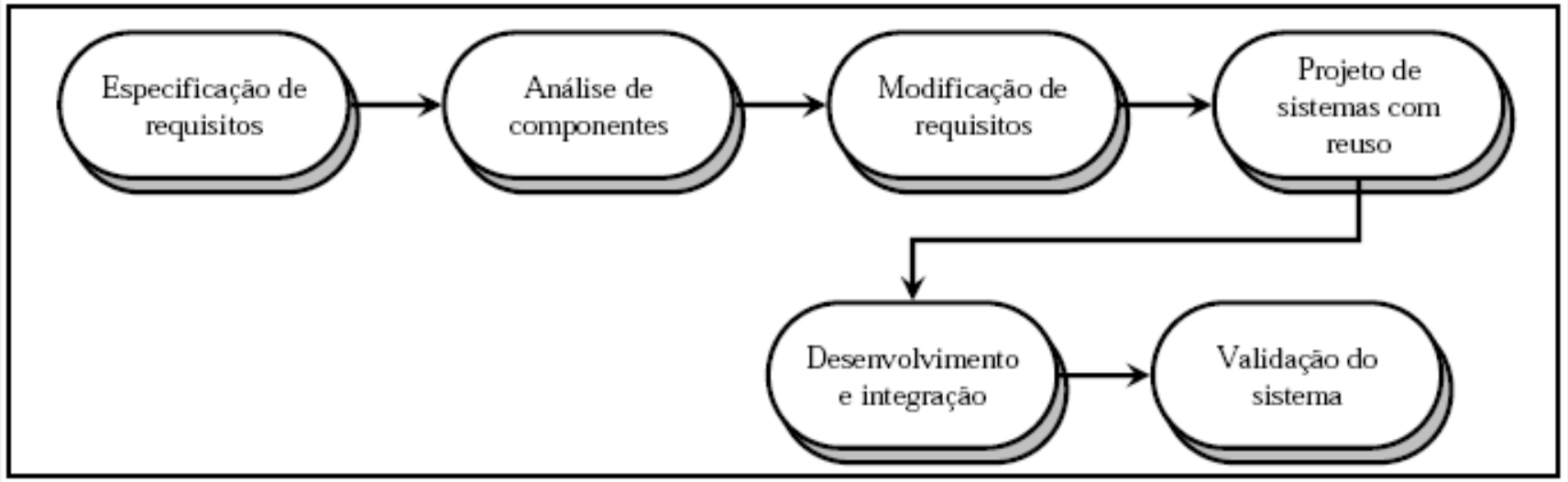
DESENVOLVIMENTO BASEADO EM COMPONENTES

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Desenvolvimento Baseado em Componentes

- Baseado no reuso sistemático, onde os sistemas são integrados de componentes existentes ou sistemas padronizados
- Estágios do Processo
  - Análise do componente
  - Modificação dos requisitos
  - Projeto do sistema com reuso
  - Desenvolvimento e integração
- Esta abordagem está se tornando mais importante, mas a experiência ainda é limitada com ela

Desenvolvimento Baseado em Componentes



UDESC

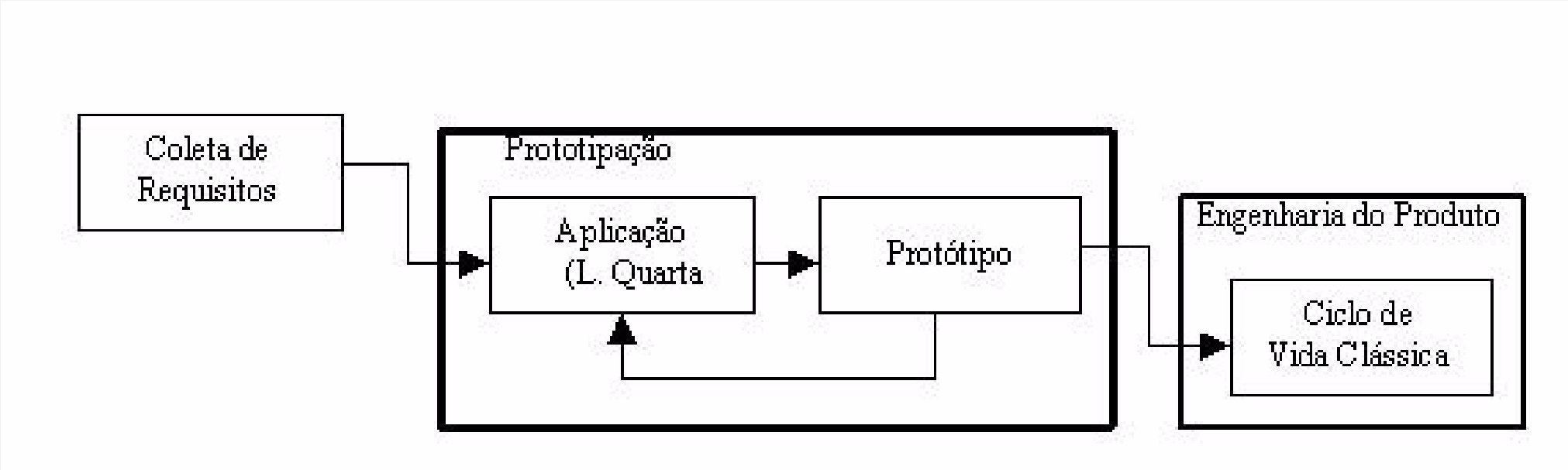
CONSIDERAÇÕES

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Que processo usar?

- Sempre deve existir um processo de software definido - padrões de qualidade.
- Criar um processo baseado em fases específico para cada projeto.
- O profissional deve estar apto a avaliar a aplicação a ser desenvolvida e a situação do ambiente de desenvolvimento para decidir qual o melhor processo de software a ser definido.

Combinação


Coleta de Requisitos
Prototipação
Aplicação (L. Quarta
Protótipo
Engenharia do Produto
Ciclo de Vida Clássica

EXERCÍCIOS

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Exercícios

1. Quais as principais dificuldades encontradas no desenvolvimento de software?
2. O que um modelo de processo de software?
3. Quais as vantagens e desvantagens do modelo Clássico ou Cascata?
4. Caracterize os modelos Incremental e Espiral. Quais as vantagens e desvtangens destes modelos? Eles são mais adequados a que tipos de projetos e por quê?

Exercícios

5. O que significa para um modelo de processo de software ser iterativo e incremental? Dê um exemplo de modelo de processo iterativo.
6. Como decidir pela aplicação de um modelo? Que informações reunir?
7. Quais as etapas que são inerentes à maioria dos modelos de processo de software?

Exercícios

9. Reflita e anote seu parecer sobre a seguinte afirmação: *“o processo de desenvolvimento de um sistema de software pode ser visto como um processo de resolução de um problema”*.
10. Reflita e anote seu parecer sobre a frase do consultor Tom Gilb (1988): *“se você não atacar os riscos [do projeto] ativamente, então estes irão ativamente atacar você”*.

Exercícios

Atividades em grupo: preparar e entregar em formato pdf.

- Avaliação:
  - Capacidade de síntese;
  - Organização.

11. Pesquise e faça uma sinopse das atividades típicas de um processo de desenvolvimento de sistemas do ciclo de vida da análise orientada por objetos.
12. Faça uma sinopse sobre a etapa de levantamento de requisitos, num processo de desenvolvimento de sistemas do ciclo de vida de sistemas da análise orientada por objetos.
13. Analise os pontos divergentes das atividades do ciclo de vida de sistemas da análise estruturada, quando comparado com as atividades do ciclo de vida de sistemas da análise orientada por objetos.

Exercícios Bônus

1. Descreva o modelo baseado em componentes com suas principais atividades.
2. O que é o Processo Unificado (*Unified Process*) e quais as suas principais características?
3. Quais são os fases do *Rational Unified Process* (RUP)? Descreva cada uma delas.
4. O que é *Rational Application Development* (RAD)? Quais as vantagens e desvantagens?

Referências

- BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. 2 ed. Rio de Janeiro: Campus, 2007.
- YOURDON, Edward. **Análise estruturada moderna**. Trad. Dalton Conde de Alencar. 3 ed. Rio de Janeiro: Campus, 1990.
- SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed. São Paulo: Pearson, 2011.
- PRESSMAN, Roger. **Engenharia de Software: uma abordagem profission**. 7 ed. São Paulo: Mc Graw-Hill do Brasil, 2011.

Obrigado

jacksonpradolima.github.io
github.com/ceplan

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA