

Engenharia de Software I

Prof. Rodrigo Martins

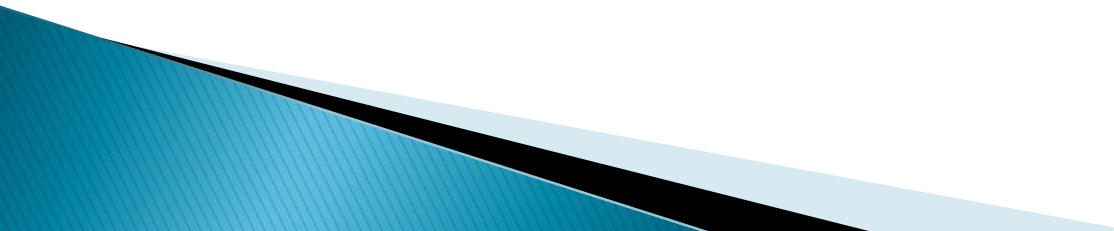
rodrigo.martins@francomontoro.com.br



Cronograma da Aula

- ▶ Apresentação da disciplina
- ▶ Introdução a Engenharia de Software
- ▶ Paradigmas de Engenharia de Software
- ▶ Exercícios

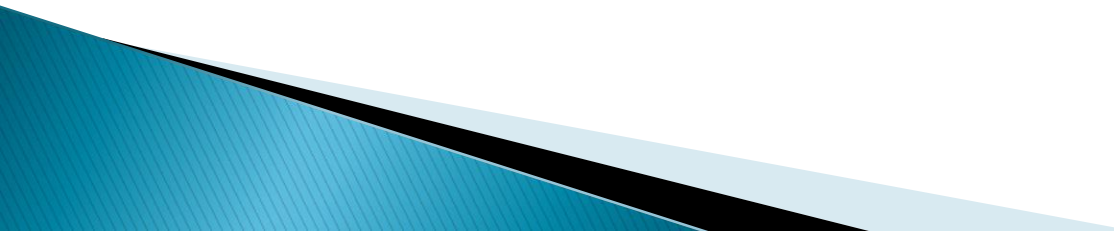
Ementa

- ▶ Introdução e aplicação de conceitos de software e engenharia de software;
 - ▶ Gerência de projetos;
 - ▶ Administração de projetos;
 - ▶ Análise de requisitos de software e de sistemas;
 - ▶ Princípios fundamentais da análise de requisitos;
 - ▶ Análise e design orientado a objetos.
- 

Objetivos

- ▶ Essa disciplina tem como objetivo geral apresentar conceitos de Engenharia de Software e como aplicar esses conceitos no desenvolvimento e projeto de sistemas.
- ▶ Trabalhar com gerência, administração e planejamento de projetos e análise de requisitos.

Ementa

- ▶ Introdução e aplicação de conceitos de software e engenharia de software;
 - ▶ Gerência de projetos;
 - ▶ Administração de projetos;
 - ▶ Análise de requisitos de software e de sistemas;
 - ▶ Princípios fundamentais da análise de requisitos;
 - ▶ Análise e design orientado a objetos.
- 

Critérios de Avaliação

- ▶ T1 – Lista de Exercícios 1 (30%)
- ▶ P1 – Avaliação Bimestral 1 (70%)

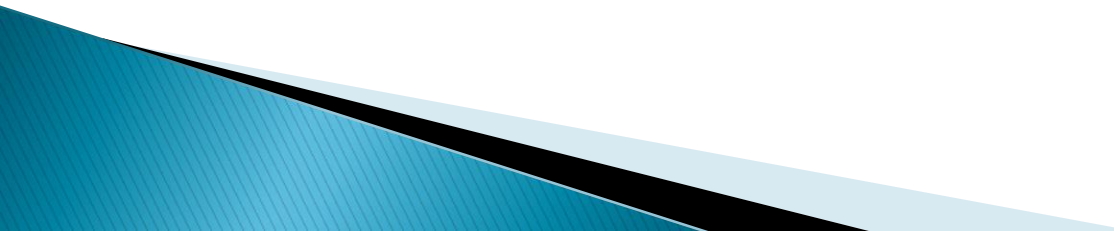
Media Bimestral: $T1 * 0.30 + P1 * 0.70$

- ▶ T2 – Lista de Exercícios 2 (30%)
- ▶ P2 – Avaliação Bimestral 2 (70%)

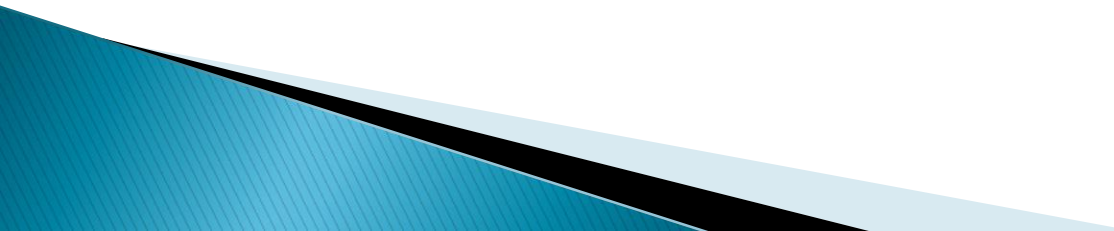
- ▶ Média Final – $(MB1 + MB2) / 2$

- ▶ Média final maior ou igual a 7,0 (sete) implicará em aprovação sem exame final;
- ▶ Média final igual ou superior a 4,0 (quatro) e inferior a 7,0 (sete) dependerá de aprovação em exame final;
- ▶ Média final de aproveitamento inferior a 4,0 (quatro) implicará em reprovação;
- ▶ A aprovação em exame final será obtida se a média aritmética da média final de aproveitamento com a nota do exame final for igual ou superior a 5,0 (cinco).

Metodologias de Trabalho

- ▶ Material exposto em sala de aula (Apresentações);
 - ▶ Indicação de Sites sobre o conteúdo (Artigos);
 - ▶ Pesquisas;
 - ▶ Uso de metodologias ativas no desenvolvimento de projetos.
- 

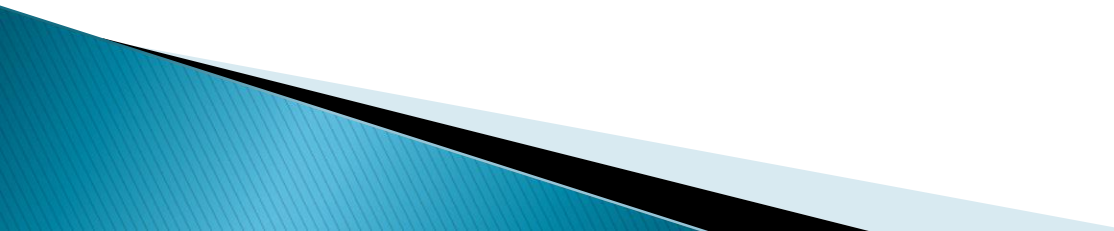
Bibliografia Básica

- PRESSMAN, R. S. Engenharia de software. 7.ed. McGraw-Hill, 2011.
 - SOMMERVILLE, I. Engenharia de software. 9.ed. Addison Wesley, 2011.
- 

Bibliografia Complementar

- WEBER, K. C.; ROCHA, A. R. C.; NASCIMENTO, C. J. Qualidade e Produtividade em Software; 4a. edição renovada. São Paulo: Makron Books, 2001.
- WILSON FILHO, P. P. Engenharia de Software: Fundamentos, Métodos e Padrões. LTC – Livros Técnicos e Científicos Editora S.A., 2001.

Características do Software

1. Desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico;
 2. Não se desgasta mas se deteriora;
 3. A maioria é feita sob medida em vez de ser montada a partir de componentes existentes.
- 

Aplicações do Software

BÁSICO	programas de apoio a outros programas
DE TEMPO REAL	monitora, analisa e controla eventos do mundo real
COMERCIAL	operações comerciais e tomadas de decisões administrativas
CIENTÍFICO E DE ENGENHARIA	algoritmos de processamento de números
EMBUTIDO	controla produtos e sistemas de mercados industriais e de consumo
DE COMPUTADOR PESSOAL	processamento de textos, planilhas eletrônicas, diversões, etc.
DE INTELIGÊNCIA ARTIFICIAL	algoritmos não numéricos para resolver problemas que não sejam favoráveis à computação ou à análise direta

Evolução do Software

➤ (1950 – 1965)

- O hardware sofreu contínuas mudanças;
- O software era uma arte "secundária" para a qual havia poucos métodos sistemáticos;
- O hardware era de propósito geral;
- O software era específico para cada aplicação;
- Não havia documentação.

Evolução do Software

➤ (1965 – 1975)

- Multiprogramação e sistemas multiusuários;
- Técnicas interativas;
- Sistemas de tempo real;
- 1ª geração de SGBD's;
- Produto de software – *software houses*;
- Bibliotecas de Software;
- Cresce nº de sistemas baseado em computador;
- Manutenção quase impossível.

..... **CRISE DE SOFTWARE**

Evolução do Software

➤ (1975 – *hoje*)

- Sistemas distribuídos;
- Redes locais e globais;
- Uso generalizado de microprocessadores – produtos inteligentes;
- Hardware de baixo custo.

⇒ Impacto de consumo

..... CRISE DE SOFTWARE

Evolução do Software

- (Quarta era do software: atualidade)
- Tecnologias orientadas o objetos;
- Computação Paralela;
- Internet;
- Sistemas especialistas e software de inteligência artificial usados na prática;
- Software de rede neural artificial.

..... **CRISE DE SOFTWARE**

Crise do Software: Problemas encontrados no Desenvolvimento de Software

▶ Principais problemas:

- Estimativas de prazos e de custo frequentemente são imprecisas.
- Produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços.
- A qualidade do software não é a esperada.

Crise do Software ..Outros problemas: (1)

- ▶ Pouco tempo dedicado para coleta de dados.
- ▶ Normalmente apenas parte das necessidades do usuário são levadas em conta.
- ▶ Os profissionais estão sempre com muita pressa para começar a programar.

Crise do Software ..Outros problemas: (2)

- ▶ Fraca comunicação entre o cliente e a equipe de desenvolvimento.
- ▶ Baixa qualidade do software.
- ▶ Não há importância com testes.
- ▶ A concorrência de software de baixa qualidade feito por pessoas sem qualificação adequada compromete a credibilidade.

Crise do Software ..Outros problemas: (3)

- ▶ Manutenção do software pode ser muito difícil.
- ▶ Consome a maioria dos recursos destinados ao software.
- ▶ Preocupação em construir softwares mais fáceis de se manter.

Mitos do Software

- ▶ Muitos dos problemas do software são consequências de uma mitologia que surgiu nos primórdios do seu desenvolvimento.
- ▶ Propagam desinformação e confusão.
- ▶ Atitudes e hábitos difíceis de modificar.

Mitos Administrativos (1)

▶ Mito:

- Temos um manual completo de padrões e procedimentos para construção de software.

▶ Realidade:

- O manual pode até existir. ..Será que é usado? ..Sua existência é conhecida? ..Será que é completo e atualizado?

Mitos Administrativos (2)

▶ Mito:

- Temos o estado da arte em ferramentas de desenvolvimento de software, compramos os mais modernos computadores.

▶ Realidade:

- Ferramentas CASE são mais importantes do que o hardware para se conseguir qualidade e produtividade, porém não são usadas pela maioria dos profissionais.

Mitos Administrativos (3)

▶ Mito:

- Estamos atrasados nos prazos, podemos adicionar novos programadores para recuperar o atraso.

▶ Realidade:

- Não funciona: Quando novas pessoas são acrescentadas, as que estavam trabalhando vão desperdiçar tempo treinando os recém-chegados.
- Existem atividades que não podem ser subdivididas.

Mitos do Cliente (1)

▶ Mito:

- Uma declaração geral dos objetivos é suficiente para começar a escrever programas.

▶ Realidade:

- Uma definição inicial ruim é a principal causa de fracasso no desenvolvimento de software.

Mitos do Cliente (2)

▶ Mito:

- Os requisitos de projeto modificam-se continuamente, mas isso não é problema, o software é flexível.

▶ Realidade:

- Os requisitos podem mudar, mas o custo da mudança pode ser muito alto.

Mitos do Profissional (1)

▶ Mito:

- Assim que escrevemos o programa e o colocamos em funcionamento o nosso trabalho está terminado.

▶ Realidade:

- Os dados da indústria indicam que entre 50 e 70% do esforço gasto num programa serão despendidos na manutenção.

Mitos do Profissional (2)

▶ Mito:

- Enquanto não tiver o programa “funcionando”, não é possível avaliar a sua qualidade.

▶ Realidade:

- A revisão técnica formal é um dos mecanismos mais efetivos de qualidade do software e pode ser aplicado desde o começo de um projeto.

Mitos do Profissional (3)

▶ Mito:

- A única coisa a ser entregue em um projeto bem-sucedido é o programa funcionando.

▶ Realidade:

- Um programa funcionando é apenas uma parte de uma configuração de software que inclui: requisitos, projeto, estrutura de dados, etc.
- A documentação é a base do desenvolvimento e guia indispensável para manutenção.

Solução para a Crise do Software

- ▶ Disciplina Engenharia de Software
 - Combina métodos e ferramentas adequadas ao processo de desenvolvimento.
 - Utiliza técnicas para garantia de qualidade.
 - Aplica uma filosofia de coordenação, controle e administração.

O que é Engenharia de Software?

- ▶ É o estabelecimento e uso de sólidos princípios de engenharia visando obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.
- ▶ Elementos fundamentais:
 - **Métodos, ferramentas e procedimentos.**

Métodos

- ▶ Proporcionam os detalhes de “como fazer” para construir o software.
- ▶ Tarefas:
 - Planejamento e estimativa do projeto;
 - Análise dos requisitos do software;
 - Projeto da estrutura dos dados;
 - Arquitetura dos programas;
 - Codificação;
 - Teste e manutenção.

Ferramentas

- ▶ Possibilitam a automatização ou semiautomatização dos métodos.
- ▶ CASE (Engenharia de Software Auxiliada por Computador):
 - Ferramentas de suporte ao desenvolvimento de software.

Procedimentos

- ▶ Conjunto de atividades que visa o desenvolvimento ou evolução racional do software.
- ▶ Definem a sequência em que os métodos são aplicados, os produtos que serão entregues, as atividades de controle de qualidade e os milestones de avaliação de progresso.

Milestone é uma técnica de gerência de projetos que permite o teste da funcionalidade de um novo produto ao longo do projeto.

Para que ES?

- ▶ Para solucionar o aumento da demanda de software.
- ▶ Para solucionar o aumento do custo (maximizar lucros).
- ▶ Para solucionar os problemas do software:
 - estimativas de prazo e de custo imprecisas;
 - qualidade não adequada;
 - manutenção;
 - complexidade (difícil entender um sw grande como um todo).

Processos de Software

- ▶ Um processo de software é um conjunto de atividades e resultados que produzem um produto de software.
- ▶ Processos de software diferentes organizam essas atividades de maneiras diferentes e são descritas em níveis de detalhes diferentes.

Etapas do Processo de Software

- ▶ Apesar de existirem diferentes processos para o desenvolvimento de software, no geral, todos os processos apresentam as seguintes atividades:
 - **Especificação:** A funcionalidade do software e restrições da sua operação são definidas.
 - **Projeto e Implementação:** Converte a especificação do sistema em um sistema executável.
 - **Validação:** O software é validado para assegurar que faz o que o usuário deseja.
 - **Evolução:** O software deve evoluir para incluir as mudanças resultantes das novas necessidades do cliente.

Paradigmas de Engenharia de Software (1)

- ▶ A estratégia usada no desenvolvimento do software deve definir etapas que envolvem métodos, ferramentas e procedimentos.
- ▶ Uma estratégia de desenvolvimento é um modelo de processo ou paradigma de engenharia de software.

Paradigmas de Engenharia de Software (2)

- ▶ A escolha da estratégia deve considerar:
 - Natureza do projeto;
 - Tipo da aplicação;
 - Métodos e ferramentas que serão usados;
 - Métodos de controle;
 - Prazo de entrega;
 - Produtos que serão entregues.

Paradigmas de Engenharia de Software (3)

▶ Principais paradigmas:

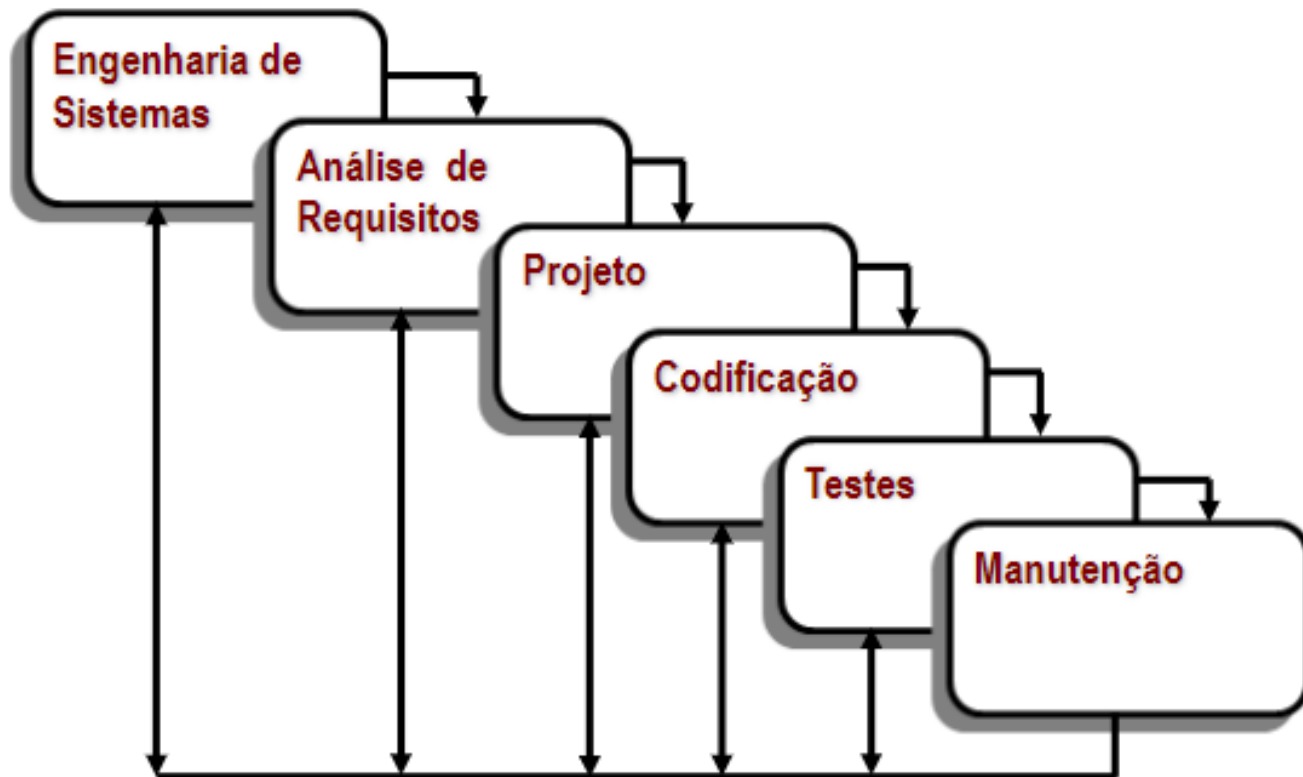
- O Ciclo de Vida Clássico (Modelo Cascata);
- Prototipação;
- Modelo Espiral;
- Fases Genéricas.

▶ Ágeis

- XP (eXtreme Programming);
- OpenUP;
- SCRUM.

O Ciclo de Vida Clássico (Modelo Cascata)

- ▶ O paradigma requer uma abordagem sequencial ao desenvolvimento de software que se inicia em um nível básico e avança ao longo da Análise, Projeto, Codificação, Teste e Manutenção.



Etapas do Modelo Cascata (1)

- ▶ **Análise e Engenharia de Sistemas:**
 - Uma vez que o software sempre faz parte de um sistema mais amplo, o trabalho inicia-se com o estabelecimento dos requisitos para todos os elementos do sistema.
 - Essa visão é essencial quando o software deve fazer interface com outros elementos, tais como hardware, pessoas e banco de dados.

Etapas do Modelo Cascata (2)

► **Análise de Requisitos de Software:**

- O processo de coleta dos requisitos é intensificado e concentrado especificamente no software.
- O Engenheiro (“Analista”) de software deve compreender o domínio (escopo) da informação.
- Os requisitos são documentados e revistos com o cliente.

Etapas do Modelo Cascata (3)

▶ Projeto:

- O projeto de software é, de fato, um processo de múltiplos passos que se concentra em quatro atributos distintos:
 - Estrutura de dados;
 - Arquitetura de Software;
 - Detalhes Procedimentais e Caracterização de Interface;
 - Como os requisitos, o projeto é documentado e torna-se parte da configuração do software.

Etapas do Modelo Cascata (4)

▶ Codificação:

- A etapa de codificação executa a tarefa de traduzir o projeto em uma forma legível por máquina.
- Se o projeto estiver bem detalhado a codificação pode ser executada mecanicamente.

Etapas do Modelo Cascata (5)

▶ Testes:

- Tão logo finalizada a fase de codificação inicia-se os testes. O processo de testes concentra-se nos aspectos lógicos internos do software, garantindo que todas as rotinas tenham sido testadas.
- Concentra-se também nos aspectos funcionais externos, verifica-se as entradas externas produzem resultados reais que reflitam o exigido.

Etapas do Modelo Cascata (6)

► Manutenção:

- Indubitavelmente, o software sofrerá mudanças depois que for entregue ao cliente. Mudanças essas por erros que foram encontrados ou porque o cliente exigiu acréscimos funcionais ou de desempenho.
- A manutenção reaplica cada uma das etapas precedentes do ciclo de vida clássico, e não a um novo.

Vantagens do Modelo Cascata

- ▶ Padroniza os métodos para análise, projeto, codificação, testes e manutenção.
- ▶ Etapas semelhantes às etapas genéricas aplicáveis a todos os paradigmas.

Desvantagens do Modelo Cascata

- ▶ Os projetos reais raramente seguem o fluxo sequencial que o modelo propõe.
- ▶ Muitas vezes é difícil para o cliente declarar todas as exigências explicitamente logo no início.
- ▶ Exige paciência do cliente. Uma versão do software não estará disponível até um ponto tardio do cronograma do projeto. Um erro grosseiro se não detectado pode ser desastroso.

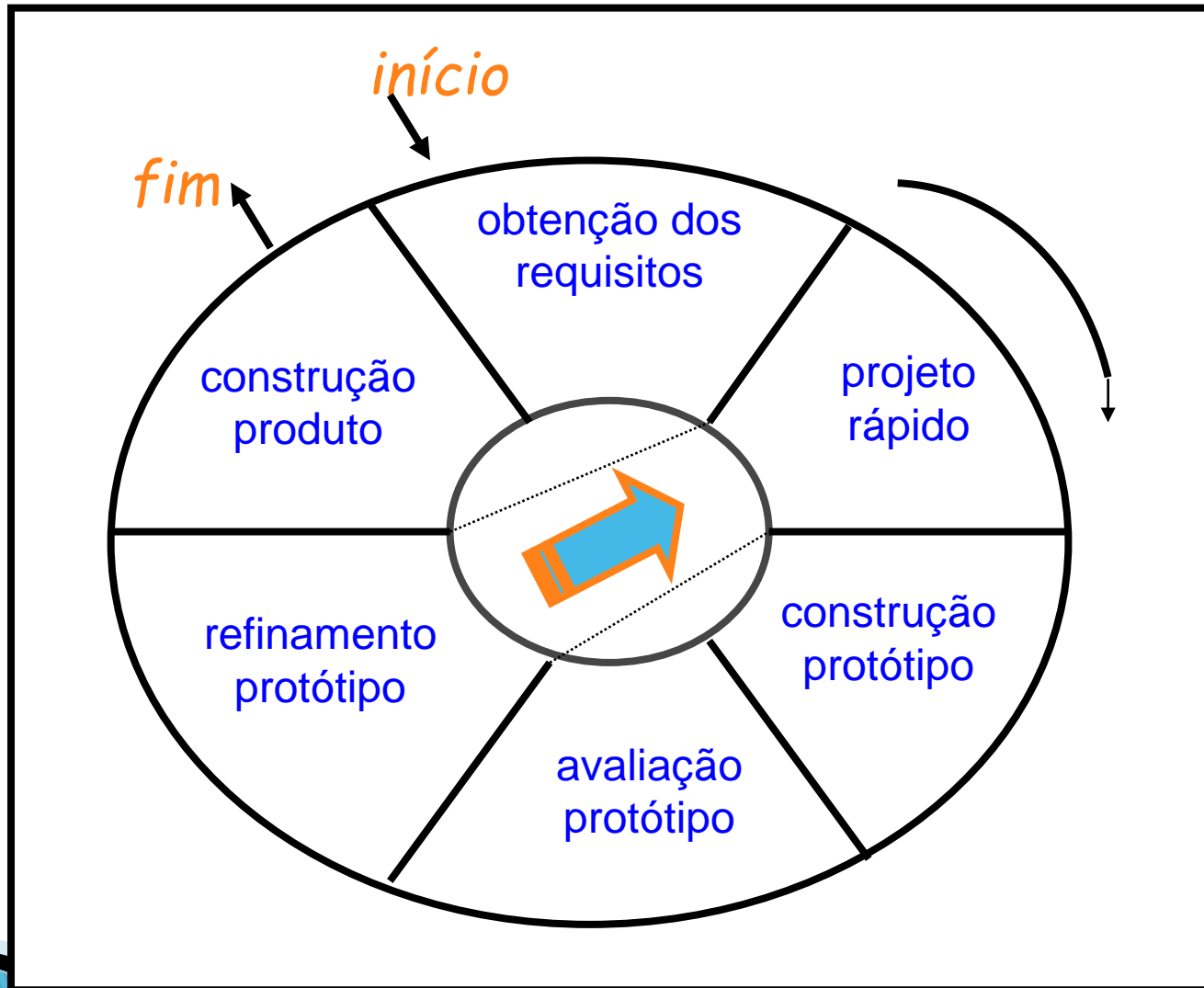
Desvantagens do Modelo Cascata

- ▶ Apesar desses problemas serem reais o paradigma do Ciclo de Vida Clássico tem um lugar definido e importante na Engenharia de Software.
- ▶ Embora tenha fragilidade, ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de software.

Prototipação

- ▶ A Prototipação é um processo no qual o desenvolvedor tem a capacidade de criar um modelo do software que será implementado. O modelo pode assumir uma das três formas:
 1. Um modelo em papel que o usuário tenha a visão da interação homem-máquina e quantas interações ocorrerá.
 2. Um protótipo de trabalho que implementa algumas funcionalidades exigidas pelo software.
 3. Um programa existente que executa parte ou toda a função desejada, mas que tem outras características que serão melhoradas posteriormente.

Prototipação



Etapas da Prototipação (1)

- ▶ Coleta e refinamento dos requisitos.
 - Definição dos objetivos globais.
- ▶ Projeto rápido.
 - Identificação interações entre usuário e software.
 - Concentra-se nas entradas e saídas do software.
 - Construção do protótipo.

Etapas da Prototipação (2)

- ▶ Avaliação do protótipo pelo cliente.
 - Refinamento dos requisitos do software.
- ▶ Refinamento do protótipo.
 - Reflete o refinamento dos requisitos.
 - Capacita o desenvolvedor a compreender melhor os objetivos do software.
- ▶ Engenharia do produto.
 - O produto final é construído baseando-se no protótipo, que é descartado.

Vantagens da Prototipação

- ▶ Melhora a qualidade da especificação do software a ser desenvolvido, contribuindo para uma queda nos custos de desenvolvimento e manutenção.
- ▶ Antecipa o treinamento dos usuários.
- ▶ Partes do protótipo podem ser aproveitadas no desenvolvimento do sistema.

Desvantagens da Prototipação

- ▶ O custo na maioria dos casos é considerado muito alto.
- ▶ O cliente tende a confundir o protótipo com uma versão do sistema.

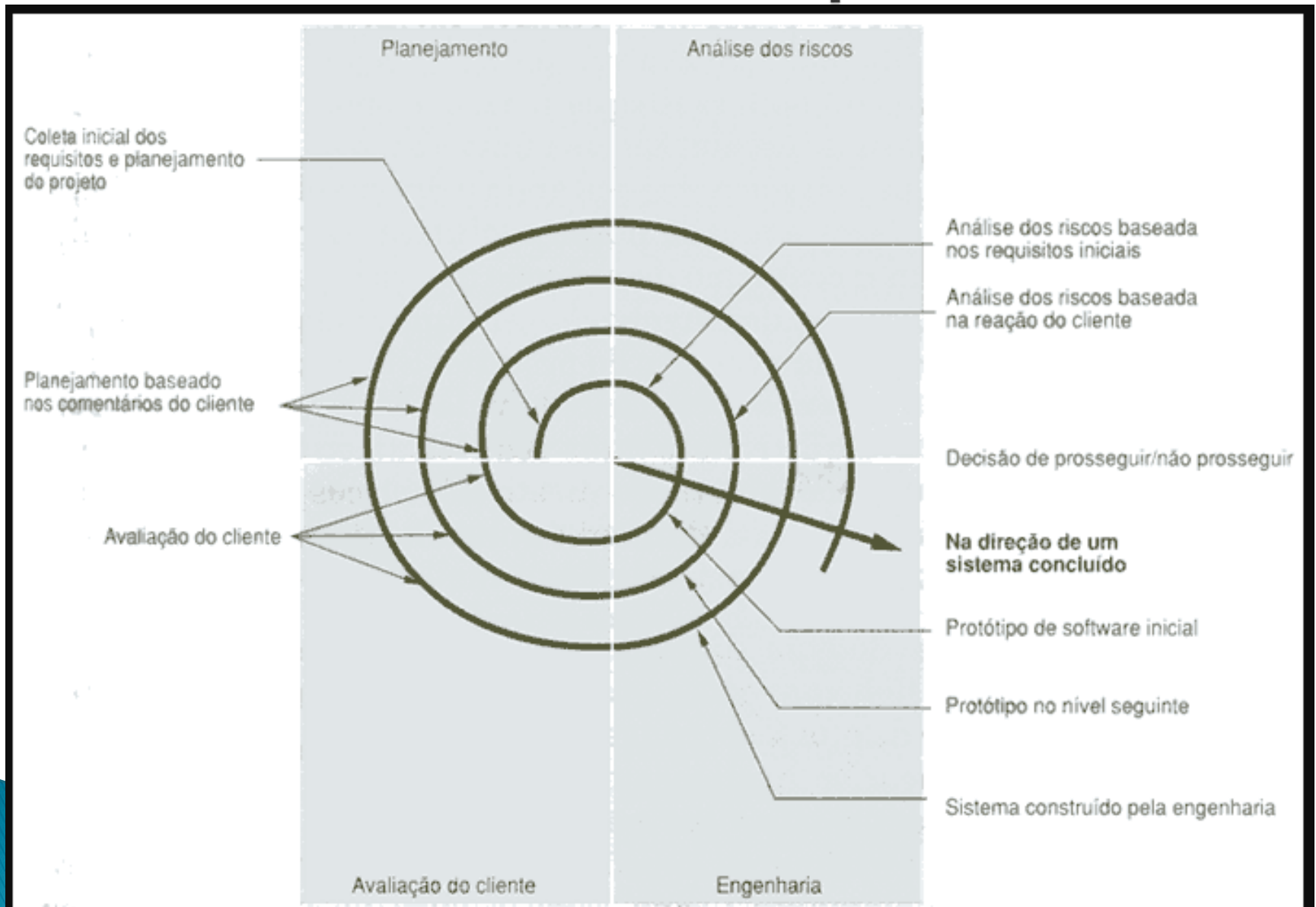
Desvantagens da Prototipação

- ▶ A Prototipação é um paradigma eficiente da Engenharia de Software.
- ▶ A chave é o acordo entre cliente e desenvolver que o protótipo será construído afim de servir como base para definição de requisitos.

O Modelo Espiral

- ▶ Aproveita as melhores características do modelo cascata e da prototipação.
- ▶ Acrescentando um novo elemento: a análise de riscos.

O Modelo Espiral



Etapas do Modelo Espiral

1. **Planejamento:** Determinação dos objetivos, alternativas e restrições.
2. **Análise dos Riscos:** Análise de alternativas e identificação/resolução dos riscos.
3. **Engenharia:** Desenvolvimento do produto no “nível seguinte”.
4. **Avaliação Feita Pelo Cliente:** Avaliação dos resultados da engenharia.

Etapas do Modelo Espiral

- ▶ Caso o cliente tenha alguma sugestão de modificação deverá apresentá-la a equipe de desenvolvimento.
- ▶ A cada iteração na espiral a conclusão da análise dos riscos resulta em uma decisão de “Prosseguir/Não Prosseguir”.
- ▶ Se os riscos forem considerados grandes, o projeto pode ser cancelado.

Vantagens do Modelo Espiral

- ▶ Modelo evolutivo possibilita uma maior integração entre as fases e facilita a depuração e a manutenção do sistema.
- ▶ Permite que o projetista e cliente possa entender e reagir aos riscos em cada etapa evolutiva.

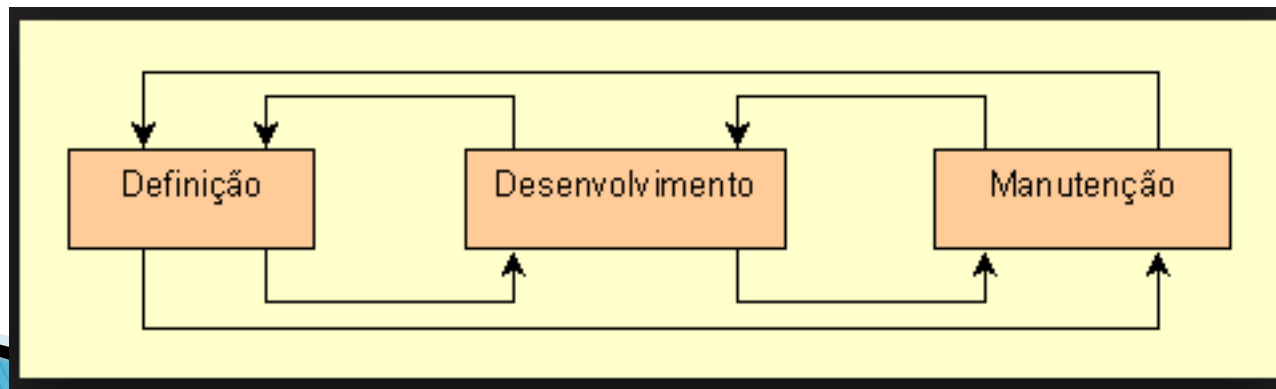
Desvantagens do Modelo Espiral

- ▶ Poderá ser difícil convencer o cliente (Principalmente em situação de contrato) de que a abordagem evolutiva é controlável.
- ▶ Ela exige considerável experiência na avaliação dos riscos.
- ▶ Se um grande risco não for descoberto, sem dúvidas ocorrerão problemas.

Fases Genéricas do Processo de Desenvolvimento

► Fases Genéricas:

- Definição, desenvolvimento e manutenção;
- Independem do paradigma adotado;
- Independem da área de aplicação, tamanho e complexidade do projeto.



A Fase de Definição

- ▶ Concentra-se no “quê”.
 - Que informações devem ser processadas?
 - Que funções devem ser desempenhadas?
 - Quais são as exigências fundamentais do software?
 - Quais são as restrições do projeto?

Etapas da Fase de Definição

- ▶ **Análise do Sistema:**
 - Define o papel que o software deverá desempenhar.
- ▶ **Planejamento do Projeto de Software:**
 - Analisa os riscos, aloca os recursos, estima os custos e define as tarefas que serão realizadas.
- ▶ **Análise de Requisitos:**
 - Define detalhadamente o domínio da informação e a função do software.

A Fase de Desenvolvimento

- ▶ Concentra-se no “como”.
 - Como projetar as estruturas de dados e a arquitetura do software?
 - Como implementar os detalhes procedimentais?
 - Como traduzir o projeto para uma linguagem de programação?
 - Como realizar os testes?

Etapas da Fase de Desenvolvimento

- ▶ Projeto de Software:
 - Traduz os requisitos do software em um conjunto de representações que descrevem o software.

Etapas da Fase de Desenvolvimento

- ▶ Codificação:
 - Converte as representações do projeto em uma linguagem artificial executada pelo computador.
- ▶ Realização de Testes de Software:
 - Procura por defeitos de função, lógica e implementação do software.

A Fase de Manutenção

- ▶ Concentra-se nas mudanças sofridas pelo software.
- ▶ Tipos de Mudanças:
 - **Correção** – Corrige defeitos encontrados pelo cliente (manutenção corretiva).
 - **Adaptação** – Acomoda mudanças ocorridas no ambiente do software (manutenção adaptativa).
 - **Melhoramento Funcional** – Estende o software para além de suas exigências funcionais originais (manutenção perfectiva).

Exercícios de Fixação

- 1) O que é Engenharia de Software? Comente seus elementos fundamentais.
- 2) Cite e explique as etapas do processo de software.
- 3) Comente sobre a crise do software.
- 4) Comente o Ciclo de Vida Clássico (Modelo Cascata), suas vantagens e desvantagens.
- 5) Comente sobre Prototipação, suas vantagens e desvantagens.
- 6) Comente sobre o Modelo Espiral, suas vantagens e desvantagens.

Referências desta Aula

- PRESSMAN, R. S. Engenharia de software. 7.ed. McGraw-Hill, 2011.
- SOMMERVILLE, I. Engenharia de software. 9.ed. Addison Wesley, 2011.

FIM
Obrigado!

Rodrigo

