

Engenharia de Software

Capítulo 1 – Introdução à ES

Histórico e Paradigmas

O que é engenharia?

A engenharia é o processo metodológico, sistemático e disciplinado que envolve a *especificação* (do sistema a ser desenvolvido), a *elaboração* ou estudo do problema, a análise do problema sobre pontos de vistas macros e micros (tanto da estrutura como da dinâmica do sistema). Passa pela *projeto* e definição da arquitetura a ser utilizada (sistemas externos, ferramentas, equipamentos e procedimentos), e pela *implantação* ou execução do sistema (utilizando padrões e mecanismos de controle de qualidade). A seguir deve-se realizar um conjunto de *testes* com o sistema para sua validação, o que inclui a verificação do atendimento dos critérios mínimos de qualidade e usabilidade do sistema final. Não devemos nos esquecer da necessidade de se *documentar* os sistemas e prever mecanismos para sua *manutenção* futura. Isto tudo dentro de prazos e custos compatíveis com o tipo de sistema a ser desenvolvido (análise de custos e *viabilidade* econômica).

O arcabouço de conhecimentos do engenheiro é dado por um conjunto de disciplinas básicas, comuns a engenharia (como matemática, física, química, estatística, lógica, calculo numérico, algoritmos e programação), e por disciplinas aplicadas a sua área de conhecimento.

O engenheiro utiliza procedimentos metodológicos e tecnológicos para produção de produtos e serviços, que melhoram a qualidade de vida das pessoas.

Motivação

- Desde os primórdios da computação
 - Desafio: aumentar o poder de processamento
- Final da década de 70
 - Business Week: “Software: a nova força propulsora”
- Meados/Final de 80, revistas renomadas
 - “A armadilha do sw: automatizar ou não?”
 - Podemos confiar no nosso software?
 - “Criar Software novo, eis uma tarefa agonizante!”

CRISE DO SOFTWARE

- Resultado da introdução de um poderoso hardware (na época)
 - Tornando viáveis aplicações até então inimagináveis!
 - As aplicações poderiam ser bem maiores e mais complexas que antes
- Experiência inicial de construção dos sistemas
 - Projetos grandes sofriam com os atrasos, às vezes de anos
 - Custos altíssimos
 - Não eram confiáveis, de difícil manutenção e de desempenho inferior
- Custos: Hardware caía, software subia

Produção de SW - Início

- Agravantes da crise do software
 - Natureza do software: sucesso medido a partir de uma unidade
 - Resistência às mudanças
 - Mitos do software (confusão e desinformação)
 - cliente, requisitos, documentação, ferramentas, atrasos
- No começo
 - Hardware era aspecto prioritário
 - Programação era considerada uma forma de arte
 - Existiam poucos métodos formais
 - Aprendizado era baseado em tentativa e erro
 - Não havia disciplina ou gerenciamento dos desenvolvedores

Produção de software - depois

■ Atualmente

- Software é prioridade
- Necessidade de técnicas para gerenciar tempo e custo
- Investimento das empresas em qualidade
- Dificuldade para lidar com sistemas legados sem documentação, que precisam de mudanças e que não podem ser substituídos (alta complexidade)
- É preciso evitar que erros se repitam!

Engenharia de Software

■ Papel

- Construção de sistemas complexos que sejam confiáveis e manuteníveis

■ Fornece

- **Métodos** - Conjunto das tarefas
- **ferramentas** - Apoio automatizado ou semi-automatizado
- **Procedimentos** - Seqüência de aplicação dos métodos, produtos entregues

■ A administração é um ponto importante

- projetos incluem aspectos técnicos (ferramentas e competências) e aspectos administrativos (estratégias, controle e liderança)

Objetivos Básicos

Um dos objetivos básicos da engenharia de software é gerar softwares com qualidade. A **Norma Qualidade de Software** (ISO/IEC 9126: NBR 13596) inclui as seguintes definições:

Funcionalidade – satisfaz as necessidades.

Confiabilidade – é imune a falhas.

Usabilidade – é fácil de usar.

Eficiência – é rápido.

Desafios

- Legado
 - Como manter e atualizar essas aplicações?
- Heterogeneidade
 - Diferentes aplicações com diferentes tecnologias
- Fornecimento
 - Uso de padrões de desenvolvimento burocráticos
 - Time-to-market cada vez menor

Perfil do Engenheiro de Software

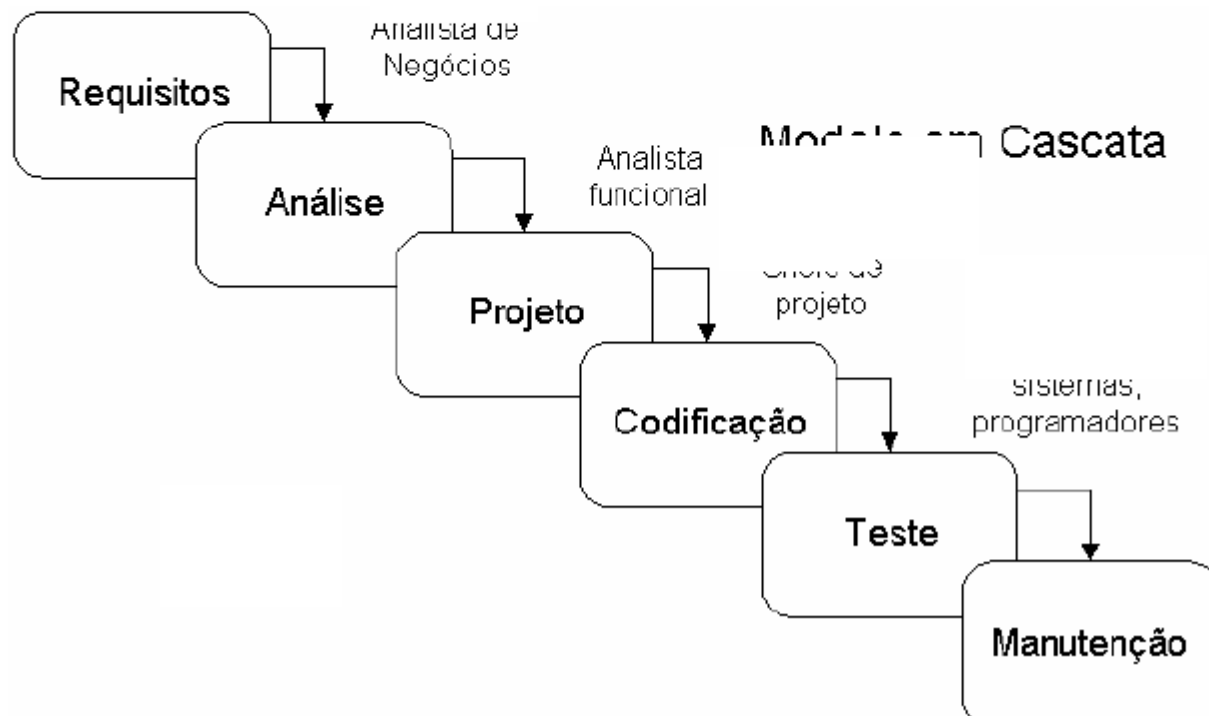
- Conhecimento em várias áreas
 - **Computação**: lógica de programação, arquitetura de computadores, estruturas de dados, redes de computadores, banco de dados
 - **Administração**: estimativa de recursos e cronogramas, atividades de planejamento, definição de formas de controle e liderança, gestão de projetos
 - **Comunicação**
 - **Solução de problemas**

Paradigmas

- Para gerenciar um processo de desenvolvimento deve-se entender o ciclo de vida
 - Diferentes paradigmas da ES
 - Cascata, prototipação, espiral
- Alguns fatores influenciam a escolha
 - Clareza dos requisitos
 - Mudanças
 - Urgência ou não que o cliente tem em ver algo pronto
 - Tamanho e complexidade do problema

Processo Cascata

- Também chamado de modelo Clássico
 - uma fase só começa quando a anterior termina
- Quando optar por esse processo?



Processo Cascata

■ Vantagens

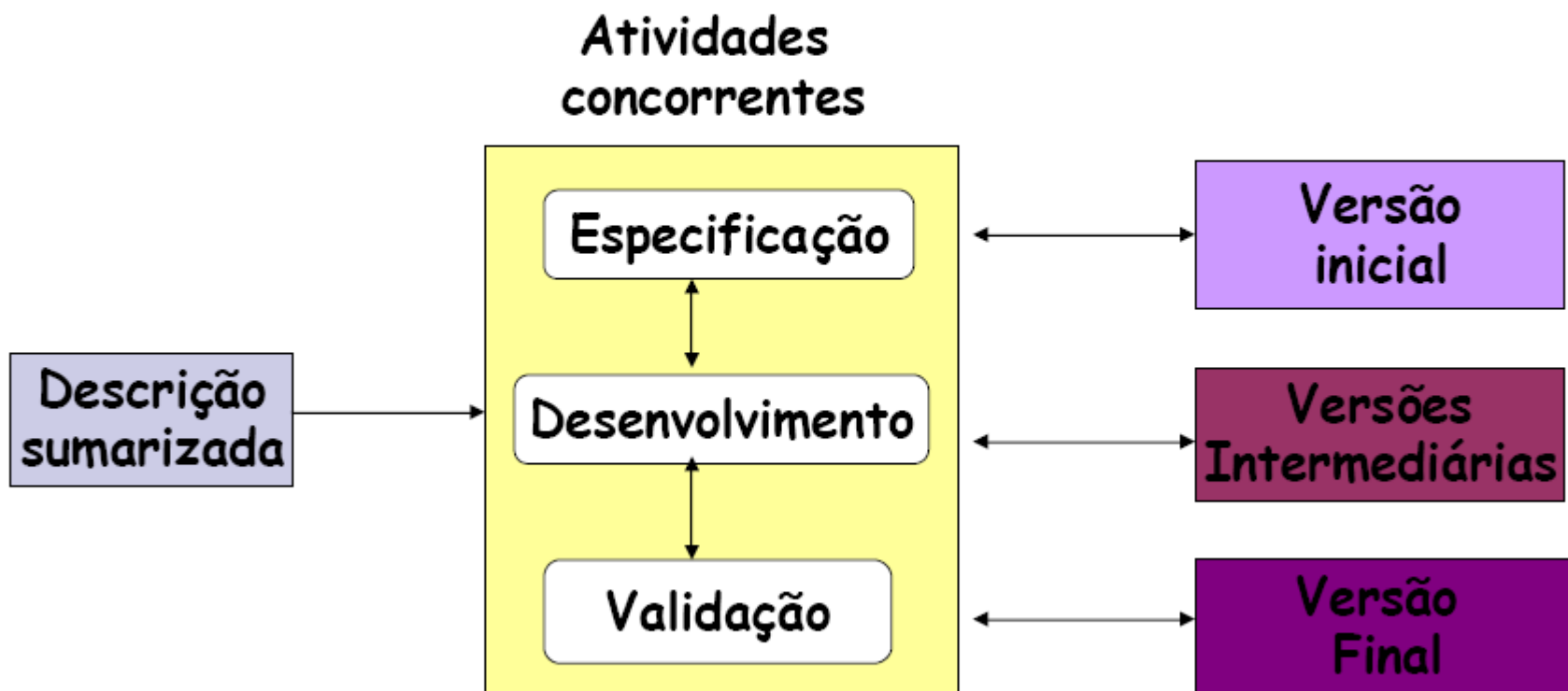
- Modelo simples de aplicar
- É mais fácil estimar tempo para finalizar o projeto

■ Desvantagens

- Projetos reais raramente seguem esse fluxo
- Difícil relacionar todos os requisitos no início
- O software será entregue quando o cronograma estiver bem adiantado

Processo Prototipação

- Também chamado de evolucionário



Processo Prototipação

- Criação de um modelo do software que será implementado
 - Facilita a definição dos objetivos do software
 - Melhora a interação entre o cliente e a equipe
- Quando utilizar?
 - Incerteza dos requisitos
 - Domínio do problema ainda não entendido
 - Sistemas pequenos, ciclo de vida curto
- Abordagens
 - Exploratória: sistema evolui funcionalmente com os protótipos
 - Descartável: protótipos de interface são descartados

Processo Prototipação

■ Vantagens

- Melhora a qualidade da especificação dos requisitos
- Aproxima o cliente da equipe
- Antecipa o treinamento dos usuários
- Partes do protótipo podem ser aproveitadas

■ Desvantagens

- O custo de construir o protótipo é alto
- Usuários confundem protótipo com o sistema final
- A mudança constante pode corromper a qualidade da estrutura do software
- Dificuldade em gerenciar o cronograma devido aos protótipos

Processo Espiral

- Acrescenta aspectos gerenciais ao processo de desenvolvimento de software
 - Ao invés de uma seqüência, uma espiral
 - Cada volta na espiral representa uma fase
 - Riscos são resolvidos ao longo do processo

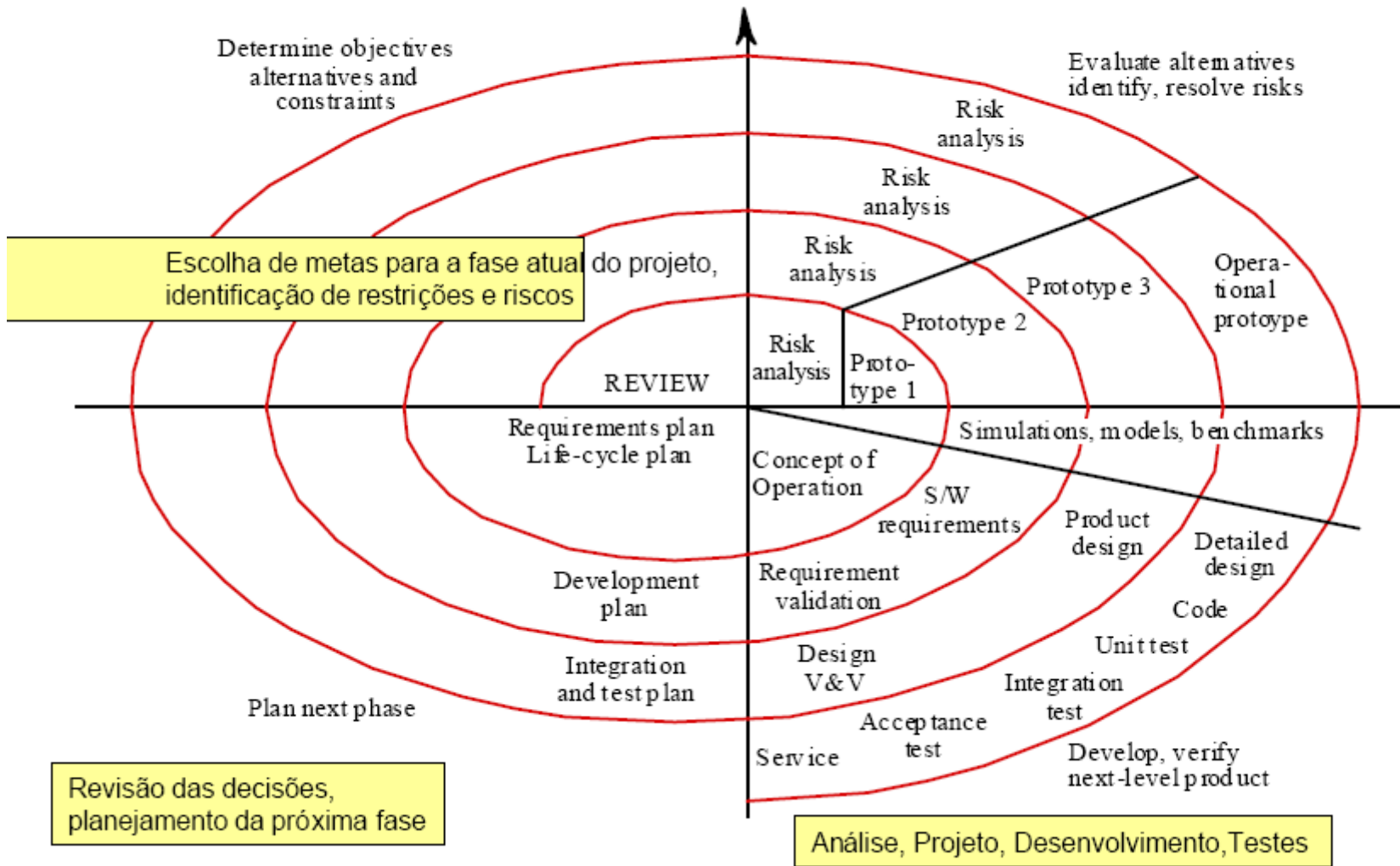
Fases do espiral

- Definição de objetivos
 - Escolha de metas para a fase atual do projeto, identificação de restrições e riscos
- Avaliação e redução de riscos
 - São tomadas providências para reduzir cada um dos riscos identificados (Ex: realizar nova entrevista com cliente para esclarecer requisitos)

Fases do espiral

- Desenvolvimento e validação
 - É escolhido um modelo de desenvolvimento para o sistema de acordo com os riscos (segurança, usabilidade, integração, etc)
- Planejamento
 - O projeto é revisado e há uma decisão sobre a continuidade do projeto; definição de planos para a fase seguinte

providências para reduzir cada um dos riscos identificados



Processo Espiral

■ Vantagens

- produto final é desenvolvido através de iterações
- Permite que o projetista e cliente possam entender e reagir aos riscos em cada etapa evolutiva
- a integração das fases facilita a depuração e manutenção do sistema

■ Desvantagens

- Avaliação dos riscos exige muita experiência
- Não há fases fixas como especificação ou projeto