Engenharia de Software
Conceitos Básicos

Introdução

- Importância do Software
  - Qualidade é fundamental
  - Conseqüências de erros no software podem ser catastróficas
  - Exemplo:

    Queda do foguete Ariane em Junho de 1996....
    US$ 500.000.000,00

Introdução

- Principal causa do fracasso no desenvolvimento de software é a não utilização de metodologias eficientes para a produção

- Solução está na formação de profissionais especializados em metodologias, técnicas e ferramentas da **Engenharia de Software**

Histórico da Engenharia de Software

- Primórdios da Computação (anos 40 e 50)
  - Grande dificuldade era fazer o hardware funcionar...
  - Programação dos computadores era feita através de reconfigurações do hardware

Histórico da Engenharia de Software

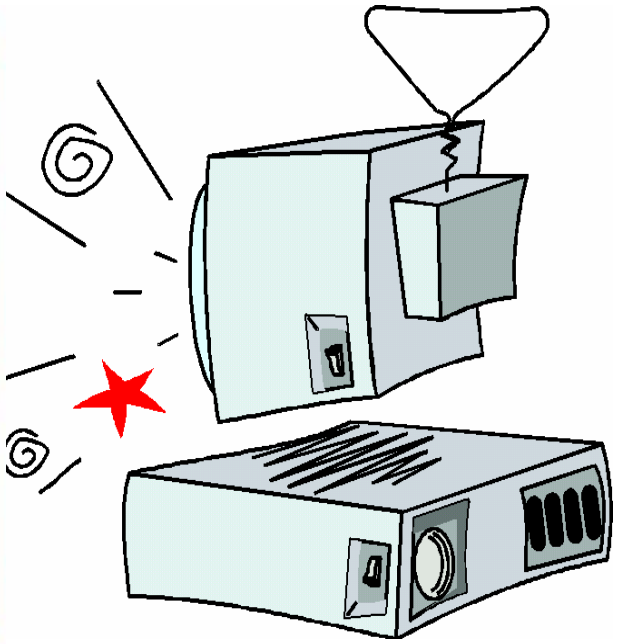
- Evolução da Computação (anos 50 e 60)
  - Aparecimento de sistemas operacionais
  - Conceito de programa armazenado
  - Surgimento das primeiras linguagens de programação:
    - *Fortran (orientação mais científica)*
    - *Cobol (voltada a aplicações comerciais)*

Histórico da Engenharia de Software

- Primórdios da Computação
  - Programação era encarada como uma atividade artística ou de lazer

Histórico da Engenharia de Software

- Crise do Software



Hardware x Software

- Custo dos equipamentos está reduzindo a cada mês!
  - mas custo de desenvolvimento de software não obedece a esta tendência
- Software é cada vez mais caro!
  - corresponde a uma percentagem cada vez maior no custo global de um sistema informatizado
- Porque é tão caro?
  - tecnologia de desenvolvimento de software implica em grande carga de trabalho

Hardware x Software

- **Porque é tão caro?**
  - **Projetos de grandes sistemas de software**
    - envolve grande número de pessoas
    - num prazo relativamente longo de desenvolvimento
  - **Desenvolvimento destes sistemas**
    - é realizado de forma "ad-hoc“
    - conduzindo a freqüentes desrespeitos de cronogramas e acréscimos de custos de desenvolvimento

Principais Aspectos do Software

Definição

- ***conjunto de instruções*** *que, quando executadas, produzem a função e o desempenho desejados,*
- ***estruturas de dados*** *que permitam que as informações relativas ao problema a resolver sejam manipuladas adequadamente*
- ***documentação*** *necessária para um melhor entendimento da sua operação e uso*

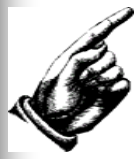
Principais Aspectos do Software

Software x Programa 

- concebidos em um contexto mais restrito
  - usuário é o próprio autor
- documentação pequena ou inexistente;
- preocupação com a existência de erros de execução não é um fator maior;
- outras boas características não são objeto de preocupação:
  - portabilidade;
  - flexibilidade;
  - possibilidade de reutilização.

Principais Aspectos do Software

Software x Programa



- desenvolvido para ser utilizado por um número maior de pessoas (sob demanda ou "de prateleira");
- deve apresentar características tais como:
  - *correção;*
  - *interface amigável;*
  - *robustez;*
  - *portabilidade;*
  - *flexibilidade;*
  - *possibilidade de reutilização.*

Principais Aspectos do Software

Software x Programa

Em *ENGENHARIA DE SOFTWARE...*

Programa ≠ **Software**

Principais Aspectos do Software

Software x Outros Produtos

- Software é um produto de engenharia e não manufaturado;
- Software não se desgasta;
- Concebidos sob medida, sem utilização de componentes pré-existentes.

Falta de Metodologia

- Processo de desenvolvimento de software pode desembocar um conjunto de problemas
  - os quais terão influência direta na qualidade do produto
- Desenvolvimento dos programas era visto como uma forma de arte
  - sem utilização de metodologias formais
  - sem qualquer preocupação com a documentação

Falta de Metodologia

- Experiência do programador era adquirida através de tentativa e erro
  - esta tendência ainda se verifica
- Com o crescimento dos custos de software no custo total de um sistema
  - processo de desenvolvimento de software tornou-se um item de fundamental importância na produção de tais sistemas

Questões Preocupantes

- Preocupações com o processo de desenvolvimento de software
  - por que demora tanto para ser concluído?
  - por que os custos têm sido tão elevados?
  - por que não é possível detectar todos os erros antes da entregue ao cliente?
  - por que é tão difícil medir o progresso durante o processo de desenvolvimento?
- Engenharia de Software pode ajudar a resolver

Causas principais

- **Alguns dos problemas que as originam**
  - **Raramente é dedicado tempo para coletar dados sobre o processo de desenvolvimento**
    - devido à pouca quantidade deste tipo de informação
    - tentativas em estimar a duração/custo de produção de um software têm conduzido a resultados bastante insatisfatórios
    - falta destas informações impede uma avaliação eficiente das técnicas e metodologias empregadas no desenvolvimento

Causas principais

- **Alguns dos problemas que as originam**
  - **os projetos de desenvolvimento são baseados em informações vagas sobre as necessidades e desejos do cliente**
    - problema de comunicação entre cliente e fornecedor
    - insatisfação do cliente com o sistema "concluído" é devido a este fato

Causas principais

- **Alguns dos problemas que as originam**
  - **qualidade do software é quase sempre suspeita**
    - problema resultante da pouca atenção que foi dada às técnicas de teste de software

Causas principais

- **Alguns dos problemas que as originam**
  - **software existente é normalmente muito difícil de manter em operação**
    - custo do software acaba sendo incrementado significativamente devido às atividades relacionadas à manutenção
    - é um reflexo da pouca importância dada à manutenibilidade no momento da concepção dos sistemas

Causas principais

- falta de experiência dos profissionais na condução de projetos de software;
- falta de treinamento no uso de técnicas e métodos formais para o desenvolvimento de software;
- “cultura de programação” que ainda é difundida e facilmente aceita por estudantes e profissionais
- a incrível "resistência" às mudanças que os profissionais normalmente apresentam

Principais Aspectos do Software

Software: Mitos do Gerenciamento

Mito 1. *"Se a equipe dispõe de um manual repleto de padrões e procedimentos de desenvolvimento de software, então a equipe está apta a encaminhar bem o desenvolvimento."*

- *Não é suficiente: manual deve especificar práticas modernas, programadores devem seguir linhas guias*

Mito 2. *"A equipe tem ferramentas de desenvolvimento de software de última geração, uma vez que eles dispõem de computadores de última geração."*

- *são necessários softwares de desenvolvimento (CASE)*

Principais Aspectos do Software

Software: Mitos do Cliente

Mito 3. *"Se o desenvolvimento do software estiver atrasado, basta aumentar a equipe para honrar o prazo de desenvolvimento."*

- *dificilmente vai ocorrer na realidade: pode atrasar o projeto*

Mito 4. *"Uma descrição breve e geral dos requisitos do software é o suficiente para iniciar o seu projeto... maiores detalhes podem ser definidos posteriormente."*

- *cliente deve procurar definir o mais precisamente possível todos os requisitos importantes*

Principais Aspectos do Software

Software: Mitos do Cliente

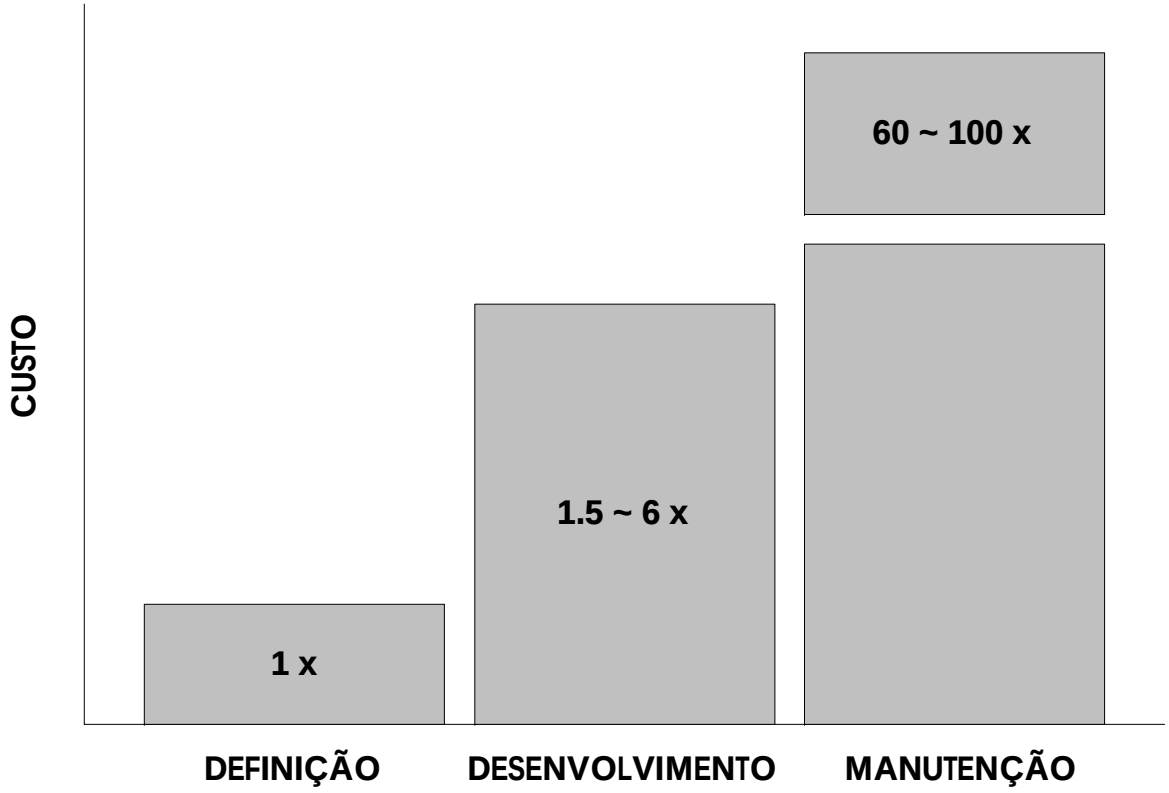
Mito 5. *"Os requisitos de projeto mudam continuamente durante o seu desenvolvimento, mas isto não representa um problema, uma vez que o software é flexível e poderá suportar facilmente as alterações."*

- *implica no aumento dos custos*
- *dependente do estágio do projeto*

Principais Aspectos do Software

Software: Mitos do Cliente

- Influência das alterações de requisitos no custo de um sistema



Principais Aspectos do Software

Software: Mitos do Programador

Mito 6. *"Após a edição do programa e a sua colocação em funcionamento, o trabalho está terminado."*

- *50 a 70% do esforço de desenvolvimento de um software é despendido após a sua entrega ao cliente (manutenção)*

Mito 7. *"Enquanto o programa não entrar em funcionamento, é impossível ter uma avaliação de sua qualidade."*

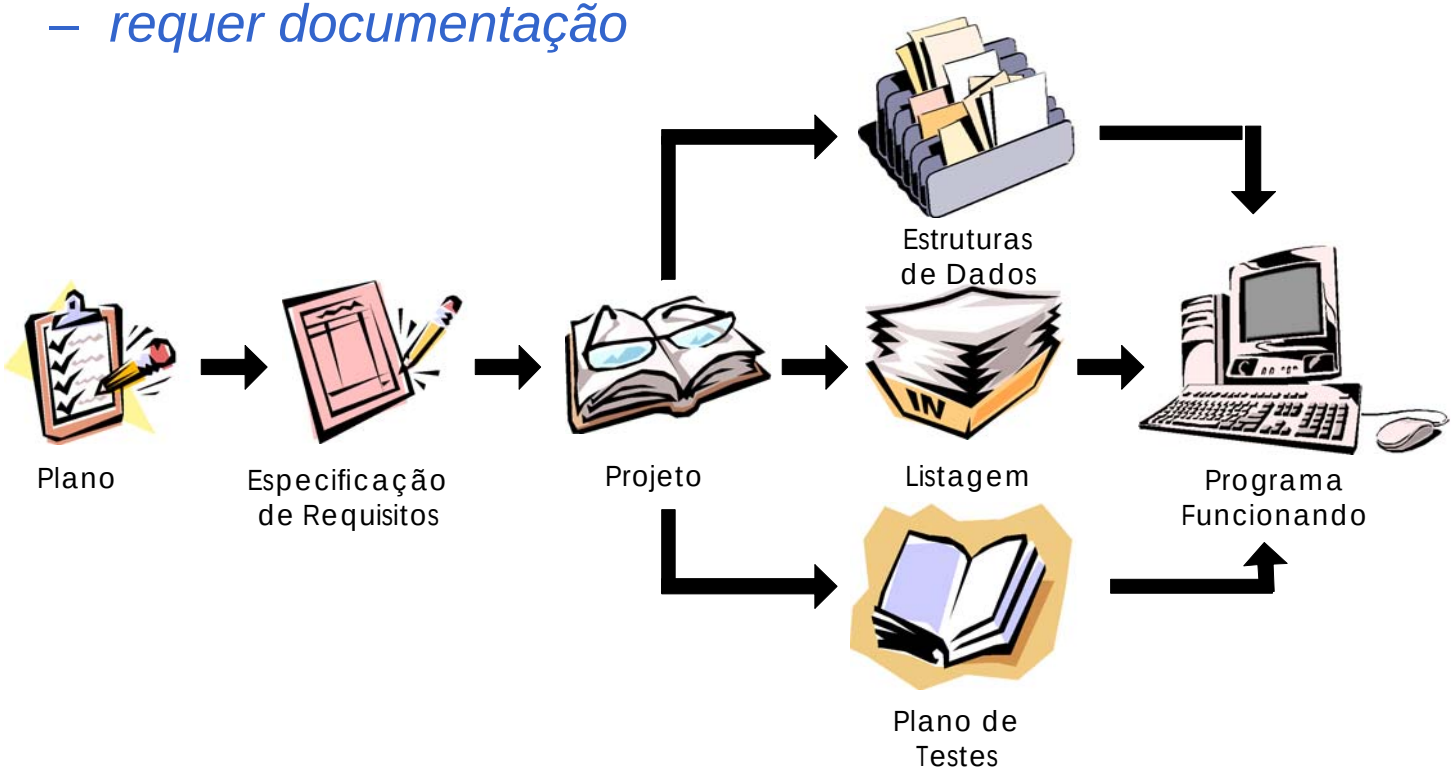
- *preocupação com a garantia do software deve fazer parte de todas as etapas do desenvolvimento*

Principais Aspectos do Software

Software: Mitos do Programador

Mito 8. *"O programa a ser entregue no final do projeto é o programa funcionando."*

– *requer documentação*



Engenharia de Software

- Não existe uma abordagem mágica que seja a melhor para a solução dos problemas
  - mas uma combinação de métodos que sejam abrangentes a todas as etapas do desenvolvimento de um software
- É importante que os métodos sejam suportados por ferramentas que permita automatizar o desenrolar destas etapas
  - juntamente com uma definição clara de critérios de qualidade e produtividade de software

Engenharia de Software

Definição

A Engenharia de Software provê a tecnologia necessária para produzir software de alta qualidade a um baixo custo.

Qualidade de Software

- **É uma definição ambígua**
  - **pois deve ser geral**
- **Trabalhos definem qualidade de software em termo de fatores**
  - **que dependem do domínio da aplicação e ferramentas utilizadas**
  - **fatores podem ser classificados em**
    - internos: visíveis aos desenvolvedores
    - externos: visíveis aos usuários

Qualidade de Software

- **Fatores**
  - **Validade**
    - capacidade de um software de cumprir exatamente suas funções, definidas pelo orçamento e especificações
  - **Confiabilidade (ou Robustez)**
    - capacidade de um software de funcionar em condições adversas
  - **Extensibilidade**
    - Facilidade com a qual um software se presta a modificações ou extensões das funções que lhe são pedidas

Qualidade de Software

- **Fatores**
  - **Reusabilidade**
    - capacidade de um software (ou parte dele) de ser reutilizado em novas aplicações
  - **Compatibilidade**
    - facilidade com a qual um software pode ser combinado com outros softwares
  - **Eficiência**
    - utilização otimizada dos recursos materiais
  - **Portabilidade**
    - facilidade com a qual um software pode ser transferido para diferentes ambientes de software e hardware

Qualidade de Software

- **Fatores**
  - **Verificabilidade**
    - facilidade de preparação de procedimentos de teste
  - **Integridade**
    - capacidade de um software de se proteger seu código e seus dados contra acessos não autorizados
  - **Facilidade de Emprego**
    - facilidade de aprendizagem, de utilização, de preparação dos dados, de interpretação dos erros e de recuperação no caso de erro de utilização

Qualidade de Software

- **Fatores são muitas vezes contraditórios**
  - **uma escolha dos compromissos devem ser efetuada em função do contexto**
- **Exemplo**
  - **Facilidade de Emprego x Confiabilidade**
    - editor de texto deve considerar primeiro a facilidade de emprego
    - controle de uma fábrica é mais importante a confiabilidade

Metodologias de Desenvolvimento de Software

- **Processo de desenvolvimento**
  - **corresponde ao conjunto de atividades e um ordenamento destas de modo a que o produto desejado seja obtido**
- **Modelo de desenvolvimento**
  - **corresponde a uma representação abstrata do processo de desenvolvimento**
    - define como as etapas do desenvolvimento do software serão conduzidas e interrelacionadas para atingir o objetivo
      - obtenção de um produto de software de alta qualidade a um custo relativamente baixo

Ciclo de Vida de um Software

- Como uma criatura viva, um software é concebido, nasce, se desenvolve, amadureçe e morre
- A vida de um software se compõe de três fases gerais

Visão Geral da Engenharia de Software

Fase de Definição

- Determina o que vai ser feito, identificando:
  - identificação das informações que deverão ser manipuladas
  - funções a serem processadas
  - qual o nível de desempenho desejado
  - que interfaces devem ser oferecidas
  - as restrições do projeto
  - os critérios de validação

Visão Geral da Engenharia de Software

Fase de Definição

- **É constituída basicamente de 3 etapas:**
  - **Engenharia (Definição) de Sistemas**
    - vai permitir determinar o papel de cada elemento (hardware, software, equipamentos, pessoas) no sistema
      - objetivo é determinar as funções atribuídas ao software
  - **Planejamento do Projeto do Software**
    - análise de riscos e a definição dos recursos, custos e a programação do processo de desenvolvimento
  - **Análise de Requisitos**
    - determinar o conjunto das funções a serem realizadas
    - principais estruturas de informação a serem processadas

Paradigmas da Engenharia de Software

Fase de Desenvolvimento

- Determina como o software será realizado
  - arquitetura, dados, algoritmos, forma como o projeto será transformado em linguagem de programação, geração de código e procedimentos de teste

Paradigmas da Engenharia de Software

Fase de Desenvolvimento

- **As etapas desta fase são:**
  - **Projeto de Software**
    - representações gráficas, tabulares ou textuais, dos requisitos do software
      - permitirão definir, com um alto grau de abstração, aspectos do software como a arquitetura, os dados, lógicas de comportamento (algoritmos) e características da interface;

Paradigmas da Engenharia de Software

Fase de Desenvolvimento

- **As etapas desta fase são:**
  - **Codificação**
    - representações serão mapeadas numa ou em várias linguagens de programação
    - geração de código de implementação
      - a partir do uso de ferramentas (compiladores, linkers, etc...)

Paradigmas da Engenharia de Software

Fase de Desenvolvimento

- **As etapas desta fase são:**
  - **Testes de Software**
    - programa obtido será submetido a uma bateria de testes para verificar (e corrigir) defeitos relativos às funções, lógica de execução, interfaces, etc...

Paradigmas da Engenharia de Software

Fase de Manutenção

- **É a fase que se inicia a partir da entrega do software**
  - **caracterizada pela realização de alterações de naturezas as mais diversas**
    - corrigir erros residuais da fase anterior
    - para incluir novas funções exigidas pelo cliente
    - para adaptar o software a novas configurações de hardware

Paradigmas da Engenharia de Software

Fase de Manutenção

- **Existem diferentes tipos de manutenção:**
  - **Correção ou Manutenção Corretiva**
    - correção de erros observados durante a operação do sistema
  - **Adaptação ou Manutenção Adaptativa**
    - alterações no software para que ele possa ser executado sobre um novo ambiente
      - CPU, arquitetura, novos dispositivos de hardware, novo sistema operacional, etc...
  - **Melhoramento Funcional ou Manutenção Perfectiva**
    - alterações para melhorar alguns aspectos do software
      - desempenho, a sua interface, a introdução de novas funções, etc...

Paradigmas da Engenharia de Software

Fase de Manutenção

- Envolve etapas de análise do sistema existente
  - entendimento do código e dos documentos associados
  - teste das mudanças
  - teste das partes já existentes

  ⇨o que a torna uma etapa complexa e de alto custo

Paradigmas da Engenharia de Software

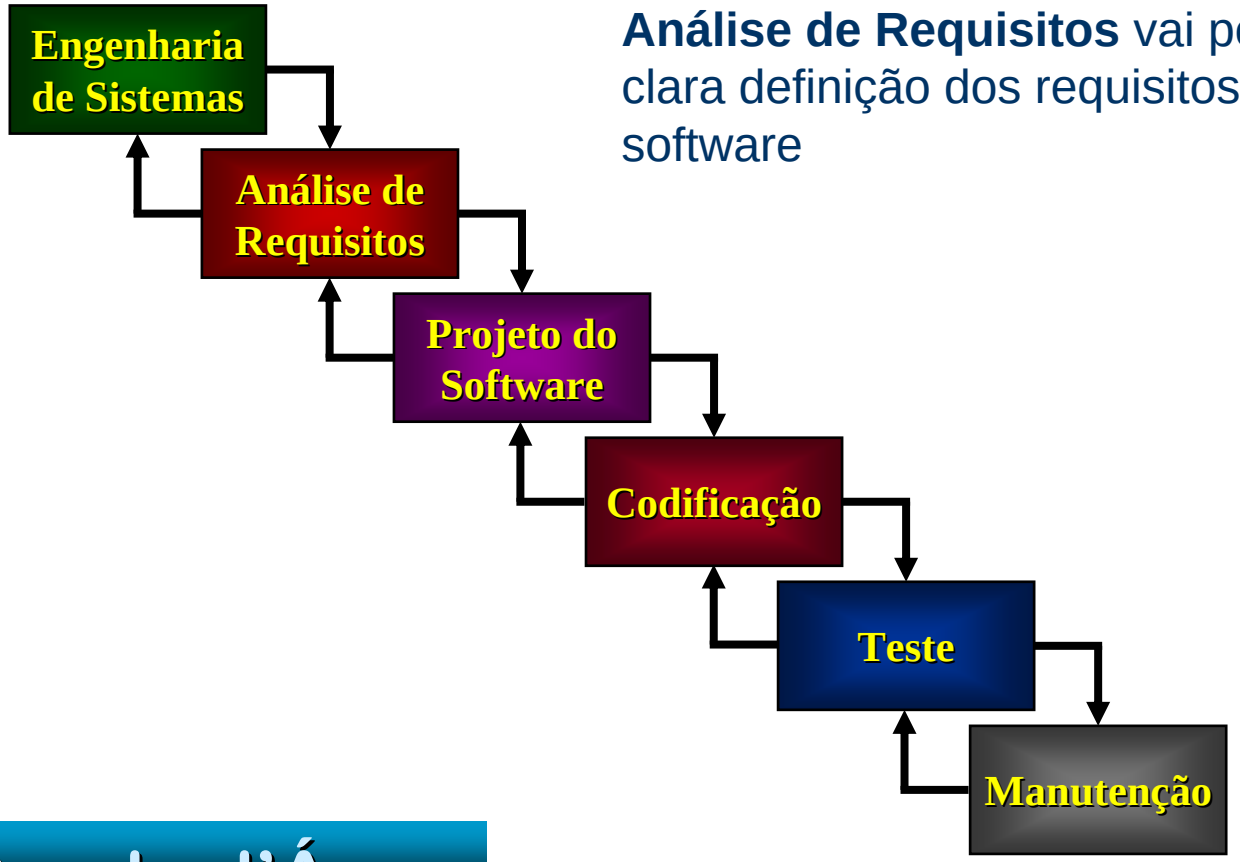
Fase de Manutenção

- Pesquisas demonstram que 53% do custo de um software é de manutenção
  - 50% melhoramento funcional
  - 17% manutenção corretiva
  - 10% manutenção adaptativa
  - 21% outros

Ciclo de Vida de um Software

- Vários modelos de desenvolvimento de software foram propostos e aplicados
  - Modelo Queda d'Água (cascata)
    - ciclo de vida clássico ou tradicional
  - Prototipação
  - Desenvolvimento iterativo
  - Modelo em Espiral
  - Modelo de Reusabilidade
  - Técnicas de Quarta Geração
- Eles englobam as fases gerais e se distinguem quanto a composição e a forma dos resultados de cada fase

Paradigmas da Engenharia de Software



Análise de Requisitos vai permitir uma clara definição dos requisitos de software

Queda d'Água

Queda d'Água: Características

- Filosofia associada
  - alcançar os objetivos pelo alcance ordenado dos sub-objetivos
- Processo seqüencial
  - cada etapa deve ser concluída antes da seguinte começar
- Toda etapa gera um produto ou documento
  - será entrada da próxima etapa

Queda d'Água: Características

- **Correções dos sub-produtos**
  - **A cada etapa o produto é verificado e validado**
    - Verificação: o produto é correto? (exato)
    - Validação: é o produto requerido?
      - comparado ao enunciado da etapa

Queda d'Água: Etapas

- **Engenharia de Sistemas (Estudo das necessidades)**
  - **Vai permitir determinar o papel de cada elemento (hardware, software, equipamentos, pessoas) no sistema**
    - objetivo é determinar as funções atribuídas ao software
  - **Resultado: um conjunto de necessidades (requisitos) que o software deveria satisfazer (documento informal)**

Queda d'Água: Etapas

- **Análise dos Requisitos**
  - **Análise detalhada de todas as funções e outras características que o software deverá realizar para o usuário**
  - **Resultado: documento contendo a especificação de «o quê» do software**
    - deve ser redigido em uma linguagem tão formal quanto possível

Queda d'Água: Etapas

- **Projeto do Software**
  - **Concepção geral**
    - Definição da arquitetura geral do software
      - sem detalhes da maneira na qual os elementos serão implementados
    - **Resultado**: Documento da concepção geral
  - **Concepção detalhada**
    - Especificação da maneira na qual cada componente de software será implementado e como eles vão interagir
    - **Resultado**: Documento da concepção detalhada

Queda d'Água: Etapas

- Codificação e Testes de Unidade
  - Fabricação de todos os componentes do sistema
  - Teste de cada um dos programas individualmente
  - Resultado: Sistema completo
- Teste de Integração (Teste)
  - Testes do conjunto do sistema
  - Resultado: software funcional

Queda d'Água: Etapas

- **Implantação**
  - Montagem do sistema no seu ambiente operacional, formação dos usuários, colocação em operação
  - **Resultado**: sistema operacional
- **Manutenção**
  - Correção, adaptação e melhoramento do sistema
  - **Resultado**: atualização dos documentos das etapas precedentes, caderno de manutenção

Queda d'Água: divisão dos esforços

- Recomendação de (Pressman, 1992)
  - Análise e Concepção : 40% - 50%
  - Codificação: 15% - 20%
  - Testes: 30% - 40%
- A realidade é diferente
  - as etapas de início do ciclo consomem menos recursos,
  - a programação muito mais

Queda d'Água: divisão dos esforços

- Os esforço da engenharia de software vão no sentido de
  - Aumentar a parte de esforço consagrada às etapas de análise e concepção
  - Diminuir a parte de esforço consagrado à programação
  - Diminuir o número de erros a detectar e a corrigir durante os testes
  - Racionalizar a etapa de testes

Queda D'Água: Prós e Contras

- **Benefícios**
  - **Integralidade**
    - é um dos poucos modelos completos
  - **Benefícios técnicos**
    - processo claro e sistemático
    - conduziu ao desenvolvimento de métodos por todas as etapas
  - **Benefícios de gestão**
    - framework para planificação e controle
    - visibilidade do progresso (dos resultados)

Queda D'Água: Prós e Contras

- **Problemas**
  - **Burocracia**
    - possibilidade de controle são algumas vezes exploradas de maneira abusiva
  - **Seqüencialidade**
    - é normalmente necessário fazer retornos
      - mal adaptado a isto (do ponto de vista técnico e de gestão)
  - **Especificações escritas**
    - nem sempre claras para os usuários

Queda D'Água: Prós e Contras

- **Problemas**
  - **Necessidade de especificações escritas, completas, antes de começar**
    - especificações completas nem sempre são possíveis
      - sistemas complexos que não substituem sistemas existentes
      - nem todos os requisitos são completamente definidos na etapa de análise
    - modelo assume que os requisitos são inalterados ao longo do desenvolvimento
      - isto em boa parte dos casos não é verdadeira

Queda D'Água: Prós e Contras

- **Problemas**
  - **Modelo impõe que todos os requisitos sejam completamente especificados antes do prosseguimento das etapas seguintes**
    - às vezes é mais interessante poder especificar completamente somente parte do sistema
      - prosseguir com o desenvolvimento do sistema
      - e só então encaminhar os requisitos de outras partes
  - **Primeiras versões operacionais do software são obtidas nas etapas mais tardias do processo**
    - o que na maioria das vezes inquieta o cliente
      - uma vez que ele quer ter acesso rápido ao seu produto

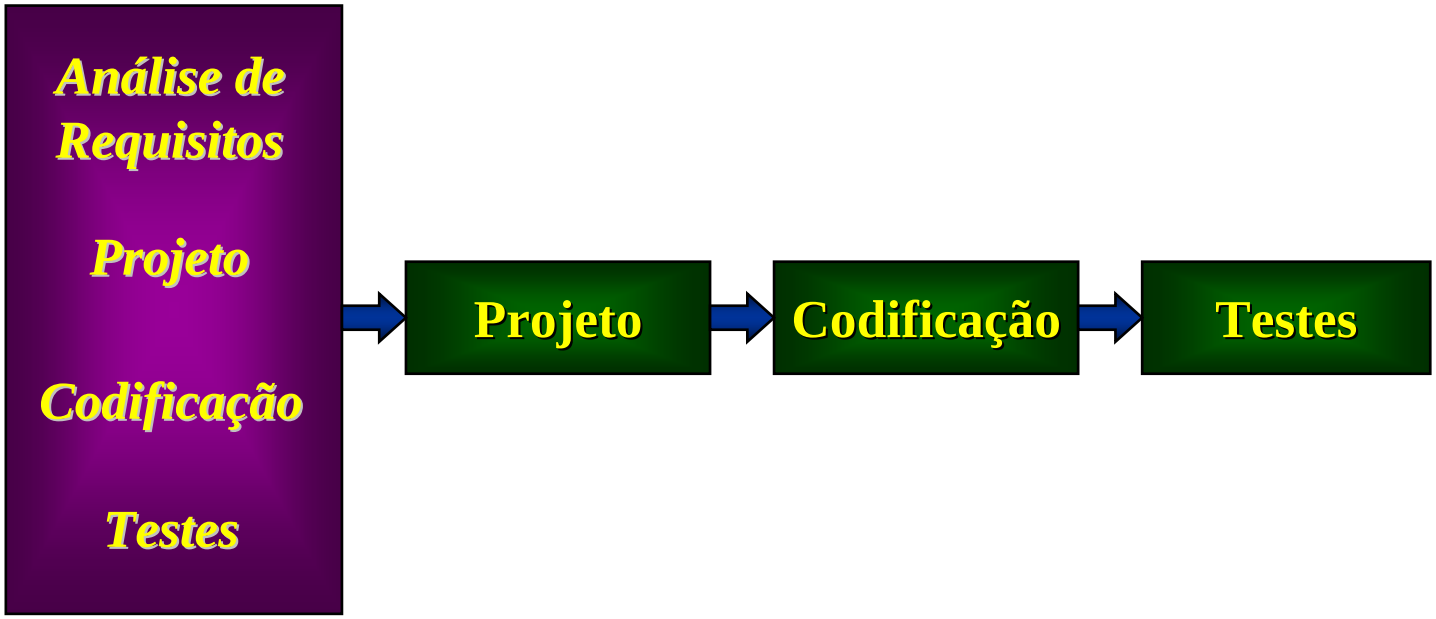
Prototipação

- **Modelo desenvolvido nos anos 80**
  - **inspirado pela prototipagem nos outros domínios da engenharia**
- **Objetivo: solucionar problemas do ciclo clássico**
  - **problemas de seqüencialidade**
  - **de má comunicação entre usuários e desenvolvedores**
    - muito tempo para ver o resultado
  - **necessidade de especificações completas**
    - eliminar a política de "congelamento" dos requisitos antes do projeto do sistema ou da codificação

Prototipação

- **Baseado no desenvolvimento de um protótipo**
  - **com base no conhecimento dos requisitos iniciais para o sistema**
  - **desenvolvimento é feito obedecendo à realização das diferentes etapas**
    - análise de requisitos, o projeto, a codificação e os testes
      - não necessariamente estas etapas devem ser realizadas de modo muito explícito ou formal

Prototipação



Prototipação

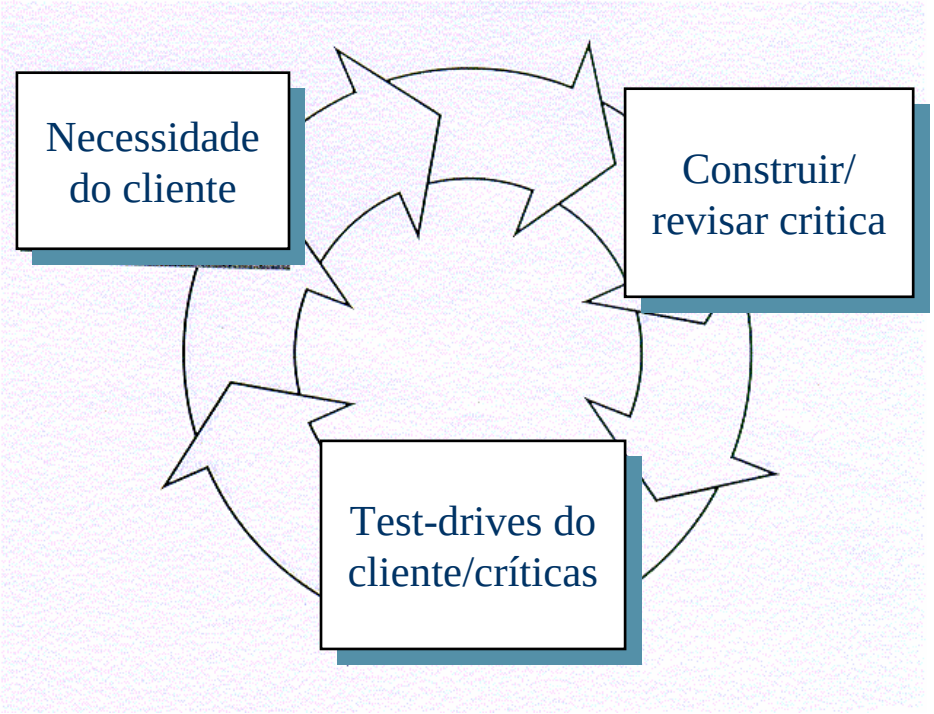
- **Protótipo pode ser oferecido ao cliente em diferentes formas**
  - **protótipo em papel**
  - **modelo executável em PC**
    - retratando a interface homem-máquina capacitando o cliente a compreender a forma de interação com o software;
  - **protótipo de trabalho**
    - que implemente um subconjunto dos requisitos indicados
  - **programa existente (pacote)**
    - que permita representar todas ou parte das funções desejadas para o software a construir

Prototipação

- Colocado à disposição do cliente
  - protótipo vai ajudá-lo a melhor compreender o que será o sistema desenvolvido
  - através da manipulação deste protótipo
    - é possível validar ou reformular os requisitos para as etapas seguintes do sistema

Prototipagem

- É um processo que permite a criação de um modelo do sistema alvo
  - serve como mecanismo para clarificar as necessidades



Prototipagem

- **Características interessantes**
  - **modelo de desenvolvimento interessante para alguns sistemas de grande porte**
    - que representem um certo grau de dificuldade para exprimir rigorosamente os requisitos
  - **é possível demonstrar a realizabilidade**
    - através da construção de um protótipo do sistema
  - **é possível obter uma versão, mesmo simplificada do que será o sistema, com um pequeno investimento inicial**

Prototipagem

- **Características interessantes**
  - **experiência adquirida no desenvolvimento do protótipo vai ser de extrema utilidade nas etapas posteriores do desenvolvimento do sistema real**
    - permitindo reduzir o seu custo
    - resultando num sistema melhor concebido

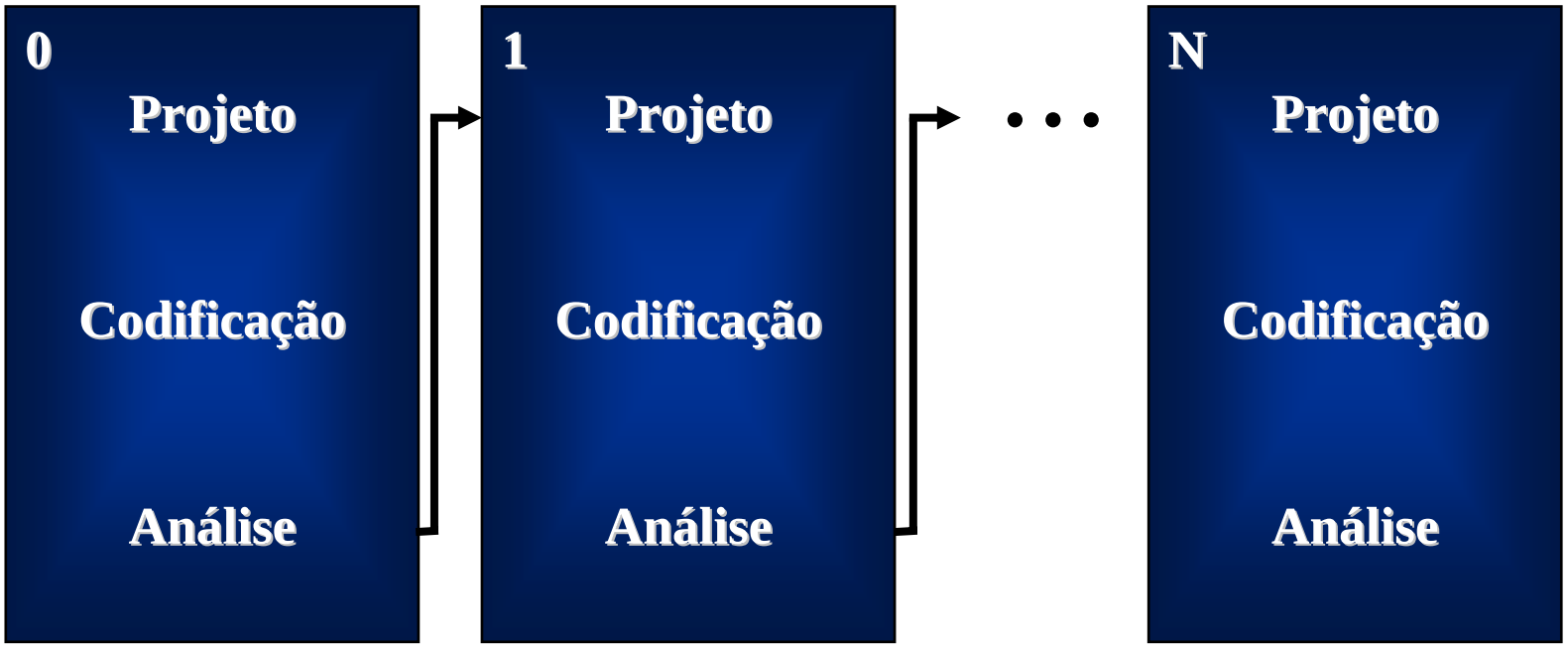
Prototipagem

- **Problemas**
  - **protótipos não são sistemas completos e deixam a desejar em alguns aspectos**
    - normalmente a interface com o usuário
    - esforços de desenvolvimento são concentrados nos algoritmos que implementem as principais funções associadas aos requisitos apresentados
      - a interface sendo, a este nível parte supérflua do desenvolvimento

Desenvolvimento Iterativo

- **Modelo concebido com base nas limitações do modelo Queda d'Água e combinar as vantagens deste modelo com as do modelo Prototipação**
  - **idéia principal é a de que um sistema deve ser desenvolvido de forma incremental**
    - cada incremento vai adicionando ao sistema novas capacidades funcionais
    - até a obtenção do sistema final
      - a cada passo realizado, modificações podem ser introduzidas

Desenvolvimento Iterativo



Desenvolvimento Iterativo

- **No primeiro passo**
  - **uma implementação inicial do sistema é obtida**
    - na forma de um subconjunto da solução do problema global
  - **deve contemplar os principais aspectos que sejam facilmente identificáveis no que diz respeito ao problema a ser resolvido**

Desenvolvimento Iterativo

- **Lista de controle de projeto**
  - **contem todos os passos a serem realizados para a obtenção do sistema final**
    - definindo quais tarefas devem ser realizadas a cada iteração
  - **gerencia todo o desenvolvimento**
    - serve para se medir, num dado nível, o quão distante se está da última iteração

Desenvolvimento Iterativo

- **Cada iteração do modelo**
  - **consiste em retirar um passo da lista de controle de projeto através da realização de três etapas**
    - o projeto, a implementação e a análise
  - **até que a lista esteja completamente vazia**

Desenvolvimento Iterativo

- **Uma vantagem desta abordagem**
  - **facilidade em testar o sistema**
    - uma vez que a realização de testes em cada nível de desenvolvimento é mais fácil do que testar o sistema final
  - **obtenção de um sistema (mesmo incompleto) rapidamente**
    - pode oferecer ao cliente interessantes informações que sirvam de subsídio para a melhor definição de futuros requisitos do sistema
    - como na Prototipação

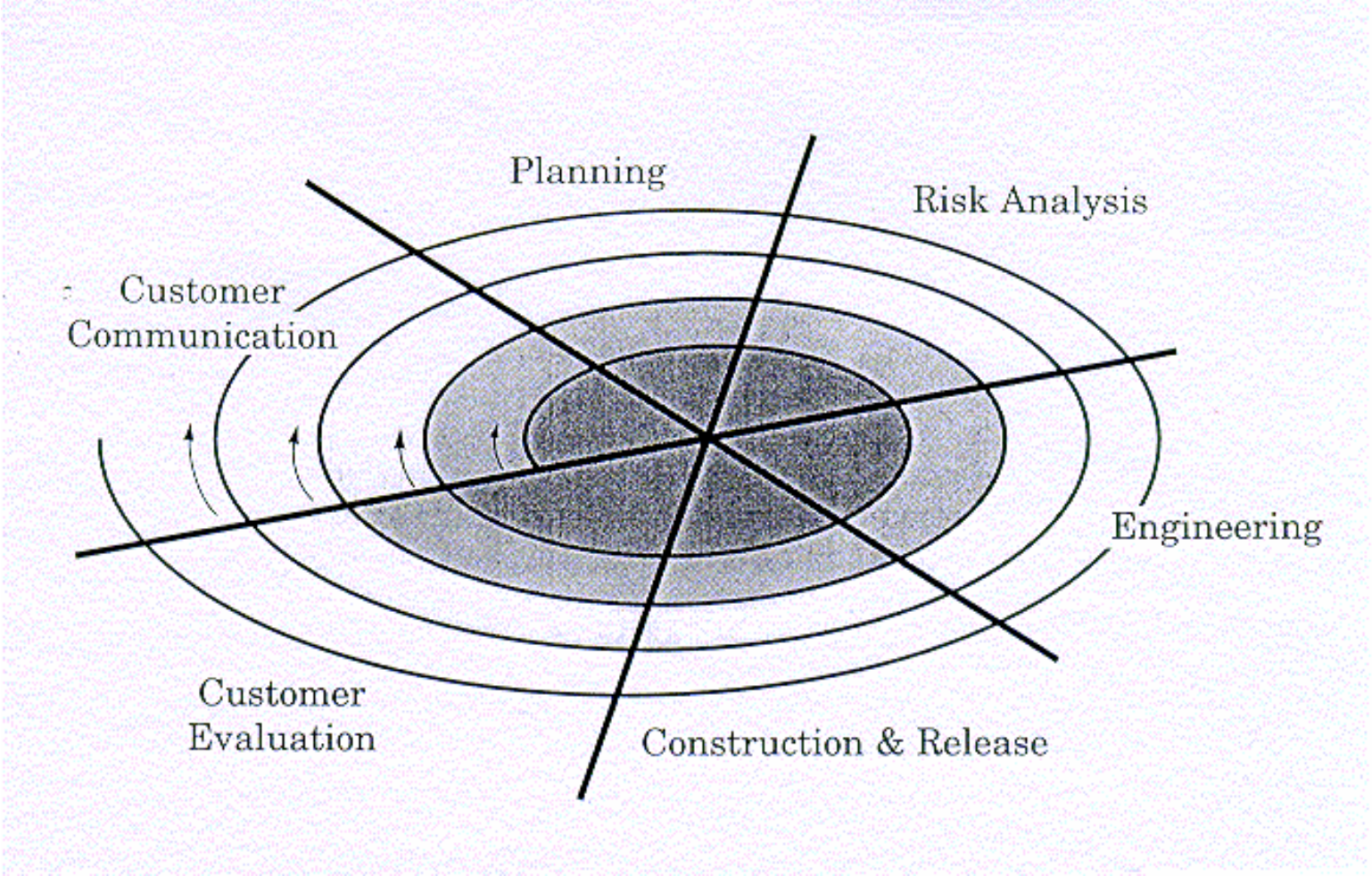
Modelo Espiral

- Modelo incremental proposto por B. Boehm em 1988
  - sugere uma organização das atividades em espiral, a qual é composta de diversos ciclos
- Integração da análise dos riscos no modelo clássico/prototipagem
  - identificação, análise das alternativas e resolução dos riscos
- Finalidade do modelo
  - se adequa principalmente a sistemas que representem um alto risco de investimento para o cliente

Modelo Espiral

- **Etapas do Modelo**
  - **Planificação**: determinação dos objetivos, alternativas e limitações
  - **Análise dos riscos**
  - **Engenharia**: desenvolvimento do produto do próximo nível
    - abordagem clássica, prototipagem ou outra
  - **Avaliação pelo cliente**
- **A cada iteração em torno da espiral**
  - se constrói uma versão mais completa do sistema

Modelo Espiral



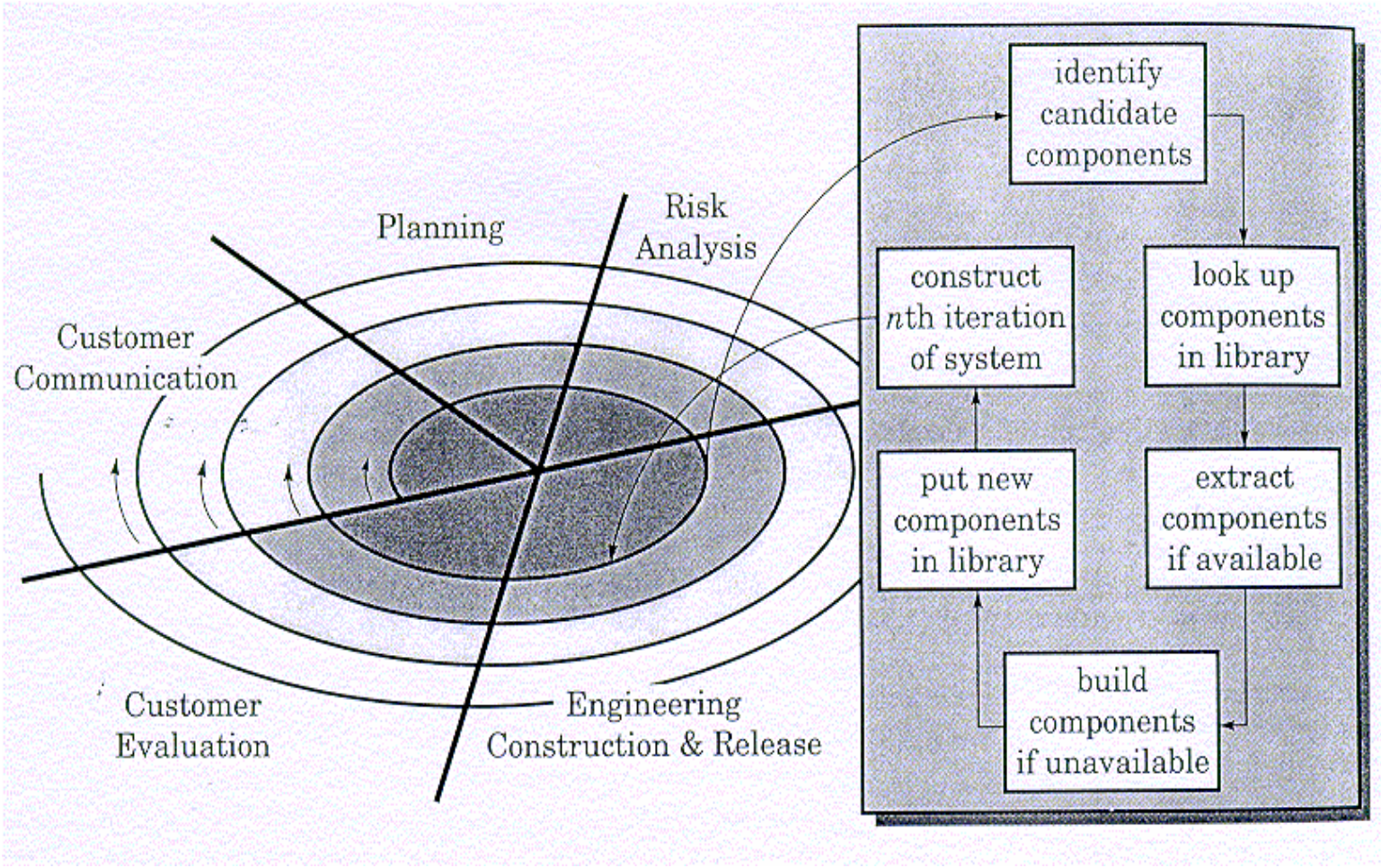
Modelo Espiral

- **Continuidade do processo**
  - **é definida em função de riscos remanescentes**
    - se riscos de desempenho ou de interface são maiores do que aqueles relacionados ao desenvolvimento
      - próximo passo é desenvolver um protótipo que elimine os riscos considerados
    - caso os riscos de desenvolvimento de programa sejam considerados os mais importantes e se o protótipo obtido no passo corrente já resolve boa parte dos riscos ligados a desempenho e interface
      - então o próximo passo pode ser simplesmente a evolução segundo o modelo Queda d'Água

Modelo de Reusabilidade

- **Reusabilidade**
  - **significa a criação e reutilização dos componentes de software**
- **Técnicas orientadas a objetos**
  - **oferecem um quadro técnico para o ciclo de vida baseada sobre componentes**
    - uma classe encapsula os dados e algoritmos que manipulam os dados
  - **reutilização dos componentes (classes) em várias aplicações e plataformas**
    - baseada na definição de bibliotecas de classes

Modelo de Reusabilidade



Modelo de Reusabilidade

- Bibliotecas de classes
  - não tem regras semânticas para juntar novas classes nas bibliotecas
  - problemas de manutenção e de garantia de qualidade
  - não se sabe se os componentes já existem
  - interfaces padronizadas não são respeitadas