MANUTENÇÃO DE SOFTWARE

Modelos de Manutenção

D.Sc. Jacilane Rabelo
jacilane.rabelo@ufc.br

Objetivo

- Modelo Quick-Fix
- Modelo de Melhoria Iterativa
- Modelo de Boehm
- Modelo de Osborne's
- Modelo Orientado a Reuso

Modelo Quick-Fix

Modelo Quick-Fix

- Abordagem **adhoc (informal)**
- **Espera o problema ocorrer** para resolver o mais rápido possível
- Mudanças são realizadas **sem análises detalhadas**
- Pode funcionar se sistema é mantido por apenas uma pessoa ou em ambientes que com pressão (mas com **alto risco para o longo prazo**)
- As **alterações** são feitas no **nível do código** o mais cedo possível, sem aceitar problemas futuros
  - Não leva em consideração seu impacto na estrutura geral do sistema de software
- Como resultado deste modelo, **a estrutura do software se degrada rapidamente**

Modelo Quick-Fix - Vantagens

- A principal vantagem é que realiza seu trabalho **com baixo custo** e **muito agilidade**
- Às vezes, os usuários não esperam muito tempo. Em vez disso, eles exigem que o software modificado seja entregue a eles no menor tempo possível.
  - Como resultado, a equipe de manutenção de software precisa usar um modelo de correção rápida para evitar o processo demorado de ciclo de vida de manutenção de software.
- Este modelo também é vantajoso em situações em que o sistema de software deve ser **mantido com determinados prazos e recursos limitados**

Modelo Quick-Fix - Desvantagens

- Este modelo não é adequado para sistemas de projetos grandes
- Este modelo não é adequado para corrigir erros por um período mais longo, pois a estrutura do sistema de software se degrada rapidamente

Modelo Quick-Fix – Estudo de Caso

- **Armazenamento cronológico de dados clínicos - ACME Health Clinic**
  - Sistema originalmente desenvolvido para fornecer **um único registro por paciente** para itens como pressão arterial, peso, medicamento e assim por diante.
    - Isso ocorreu devido a um mal-entendido durante a análise de requisitos, que não apareceu até que o sistema estivesse em uso.
  - Na verdade, o **sistema precisava armazenar várias gravações** para cada paciente
  - Logo, a necessidade de armazenamento de dados cronológicos foi imediata
  - O programador de manutenção designado para a tarefa concebeu um modelo mental de dados mantidos em **pequenas matrizes para permitir uma recuperação rápida** e começou a implementar a mudança

Modelo Quick-Fix – Estudo de Caso

- **Armazenamento cronológico de dados clínicos - ACME Health Clinic**
  - Com o método de correção rápida identificou a necessidade de matrizes, para **alterar estruturas de dados para permitir que os dados cronológicos fossem vinculados**, para um pequeno programa de reestruturação para modificar os dados existentes
  - **Não houve nenhuma atualização de documentação, nenhuma documentação das mudanças além de alguns comentários no código e nenhuma análise aprofundada.**

Modelo Quick-Fix – Estudo de Caso

Como uma solução rápida, problemas como **transbordamento da matriz não foi considerado.**

- Uma vez **armazenadas informações suficientes**, os dados "cairiam no final" dos arrays e desapareceriam.
- Isso levaria à corrupção de dados, onde os links cronológicos seriam quebrados e os links ausentes apareceriam no meio das cadeias de dados

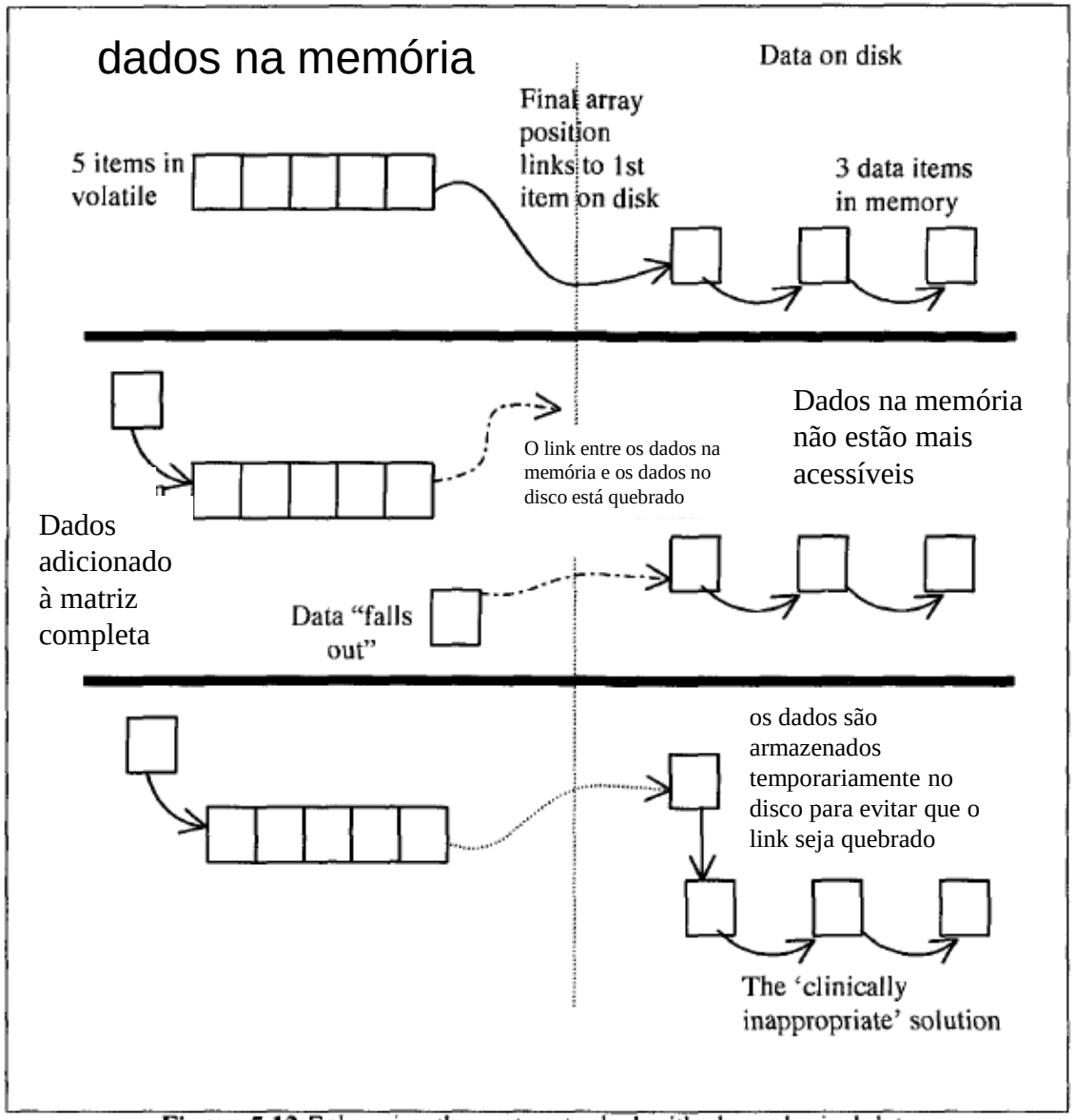


Figure 5.13 Enhancing the system to deal with chronological data

Modelo Quick-Fix – Estudo de Caso

- **Armazenamento cronológico de dados clínicos - ACME Health Clinic**
- Isso foi percebido enquanto **outro aprimoramento estava sendo testado**,
  - a gravidade potencial do **problema foi reconhecida** e começou a pensar em soluções para armazenar os dados dos clientes
- Isso impôs **outro prazo apertado** e a solução mais rápida teve que ser encontrada
- **A 'melhor' solução, uma reestruturação radical dos dados e procedimentos para recuperação e armazenamento de dados**, foi reconhecida, mas não pôde ser implementada devido a limitações de tempo.
- Outra solução rápida precisava ser encontrada

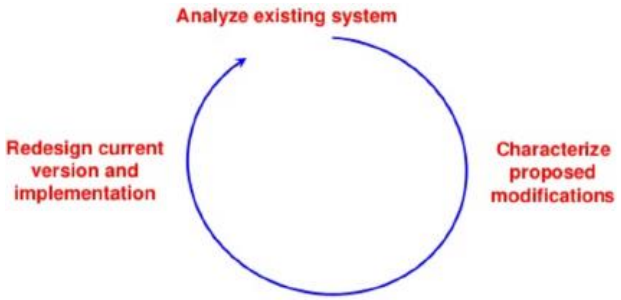
Modelo Quick-Fix – Estudo de Caso

- **Armazenamento cronológico de dados clínicos - ACME Health Clinic**
- A única solução era "obter" um array temporário de dados transbordando e armazená-lo no arquivo do paciente
- Isso significava que os vínculos cronológicos eram mantidos, mas os dados eram armazenados no arquivo do paciente sem serem salvos explicitamente pelo médico
- Isso não apenas levou a **um código menos bem estruturado**, **documentação ainda mais desatualizada** e **uma situação ainda mais difícil de recuperar**, mas também **violou um requisito original de salvamento permanente de dados.**

Modelo de Melhoria Iterativa

Modelo de Melhoria Iterativa

- Foca em melhorar o sistema de forma iterativa
- Originalmente proposto como um modelo de desenvolvimento
- Adaptado para manutenção
- Estágios:
  - Análise
  - Caracterização da proposta de modificação
  - Novo projeto e implementação



Modelo de Melhoria Iterativa

- Foca em melhorar o sistema de forma iterativa
- Originalmente proposto como um modelo de desenvolvimento
- Adaptado para manutenção
- Estágios:
  - Análise
  - Caracterização da proposta de modificação
  - Novo projeto e implementação

Modelo de Melhoria Iterativa

- Modelo foi proposto com base no princípio de que **a implementação de mudanças em um sistema de software é um processo iterativo** e **envolve o aprimoramento** de tal sistema de forma iterativa
- Adaptado para manutenção, o modelo assume uma **documentação completa**, pois depende da modificação disso como ponto de partida para cada iteração
- A **documentação existente para cada estágio** (requisitos, design, codificação, teste e análise) é modificada começando com o documento de nível mais alto afetado pelas mudanças propostas
  - Essas modificações são propagadas através do conjunto de documentos e o sistema redesenhado

Modelo de Melhoria Iterativa

- O modelo suporta explicitamente a **reutilização** e também acomoda outros modelos, por exemplo, o **modelo de correção rápida**
- Uma **solução rápida pode ser realizada, áreas problemáticas identificadas** e a próxima iteração tratará delas especificamente
- Embora o uso mais amplo de modelos de manutenção estruturados levará a uma cultura em que a documentação tende a ser mantida atualizada e completa, a **situação atual é que nem sempre é esse o caso.**

Modelo de Boehm

- Em 1983, Boehm propôs um **modelo de processo para manutenção do software**
- O processo de manutenção é representado através de um **ciclo fechado**

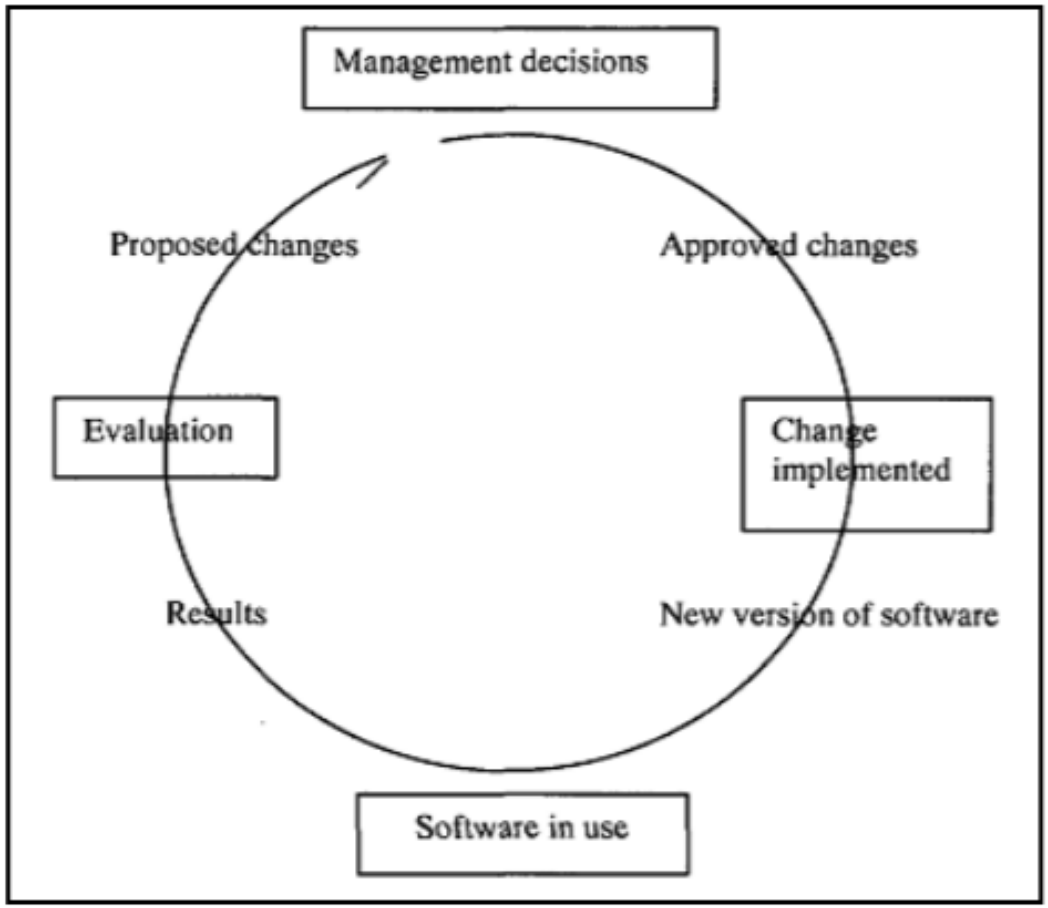


Figura 1 – Modelo de Boehm. (AGGARWAL, 2007).

Modelo de Boehm

- Modelo possui **uma etapa para o gerenciamento de decisões**, nesta etapa são realizadas estratégias particulares e avaliação do custo benefício para um conjunto de alterações propostas
- As propostas de alterações aprovados, seguem o ciclo para implementação das mudanças, geração da nova versão, utilização, resultados e avaliação

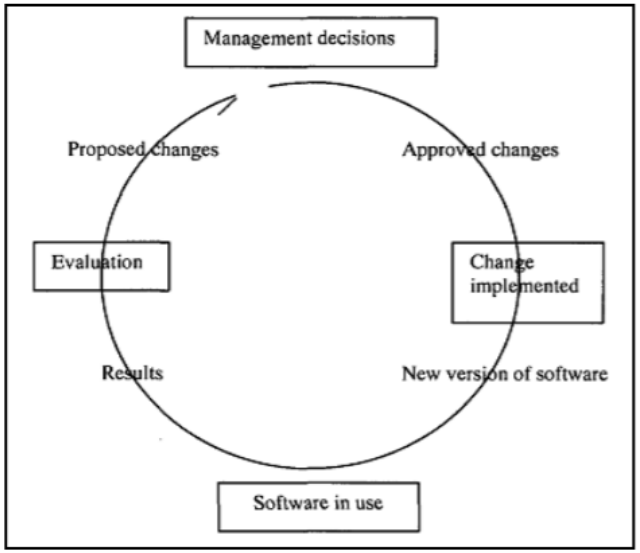


Figura 1 – Modelo de Boehm. (AGGARWAL, 2007).

Modelo de Boehm

- O modelo reflete a economia de investimentos e a relação entre os ganhos em três fases.
  - 1ª. fase - correspondente a um produto de software recém lançado que possui alta exigência de correções e melhorias de emergência obrigatórias
  - 2ª. Fase - de alto retorno, onde é visível o aumento do benefício do produto de software e os problemas inicias são resolvidos
  - 3ª. fase - retornos decrescentes, onde a taxa de aumento de benefício cumulativo diminui, nesta fase mudanças radicais tornam-se menos custosas e menos eficazes também.

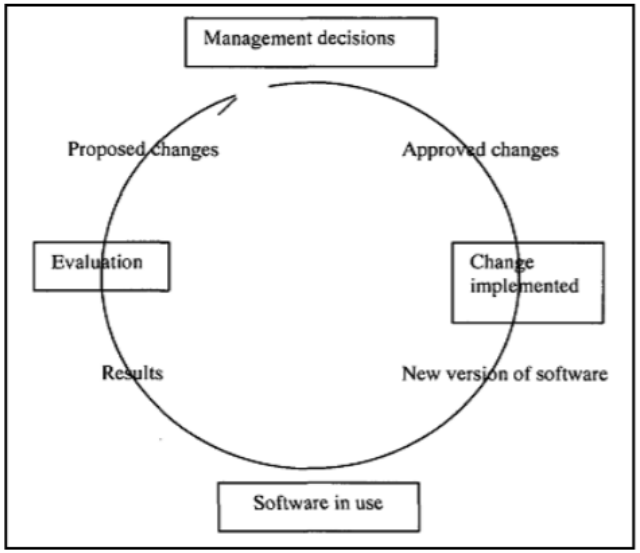
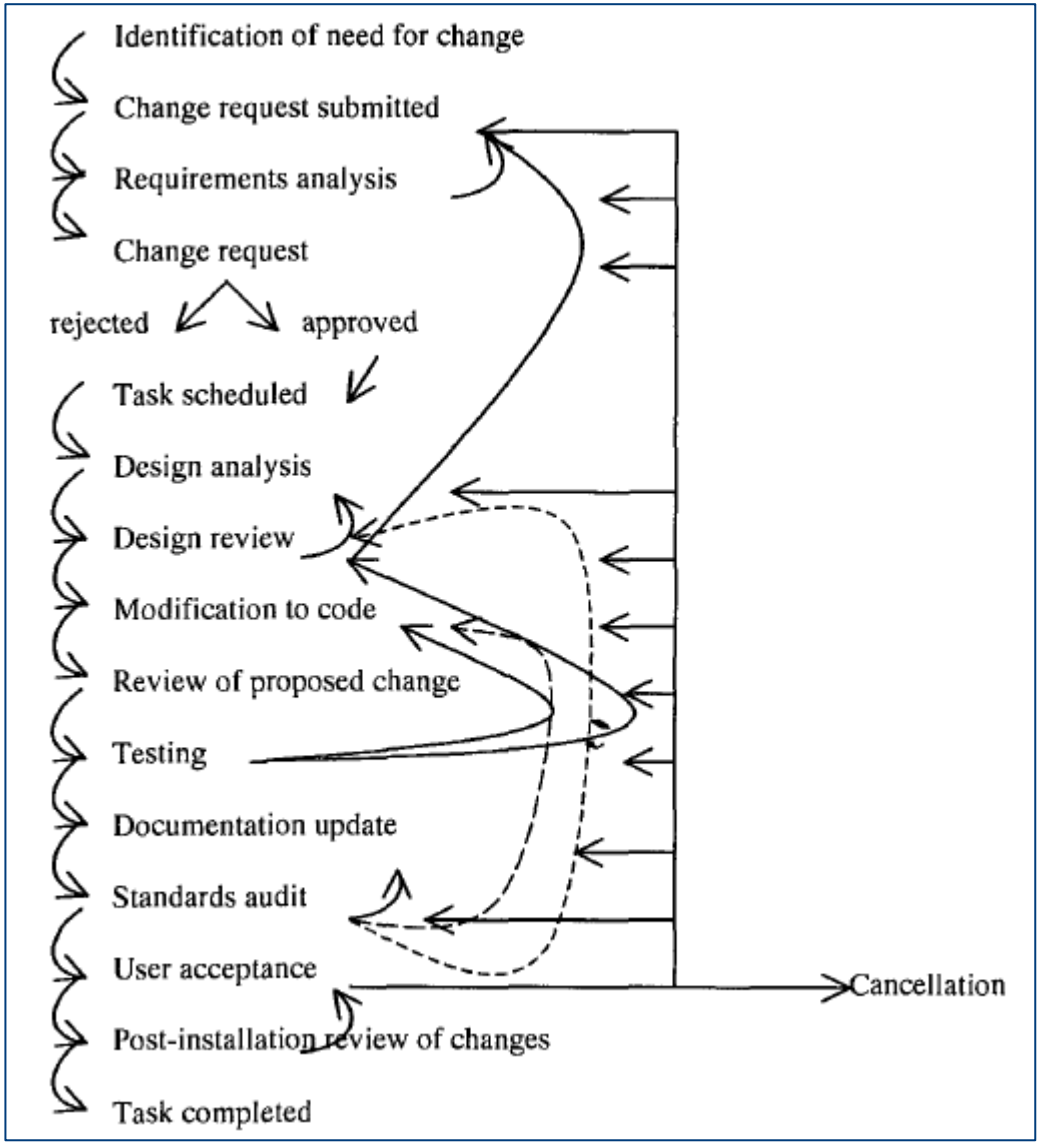


Figura 1 – Modelo de Boehm. (AGGARWAL, 2007).

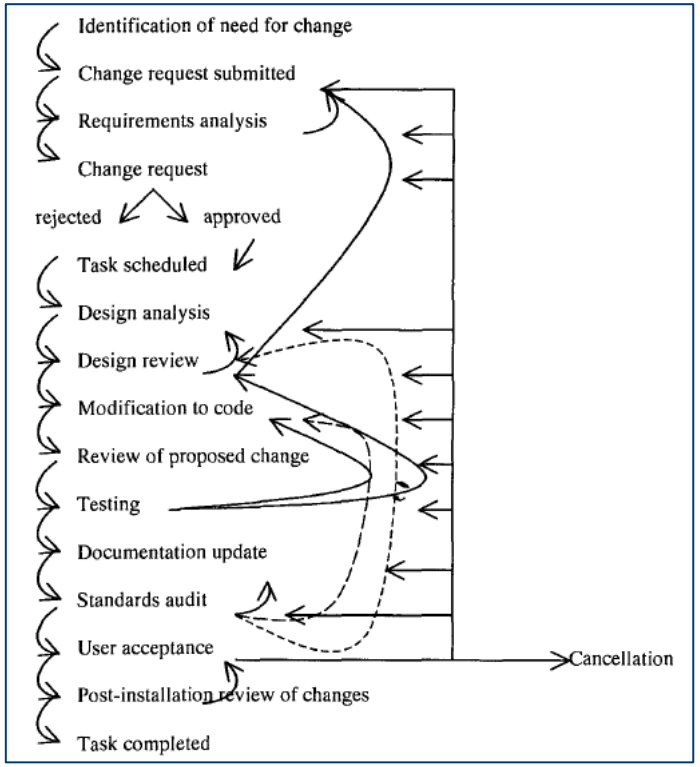
Modelo Osborne's

- A diferença entre este modelo e os outros descritos aqui é que ele lida diretamente com a realidade do ambiente de manutenção
- O modelo de Osborne leva em consideração como **as coisas são e não como gostaríamos que fossem**.



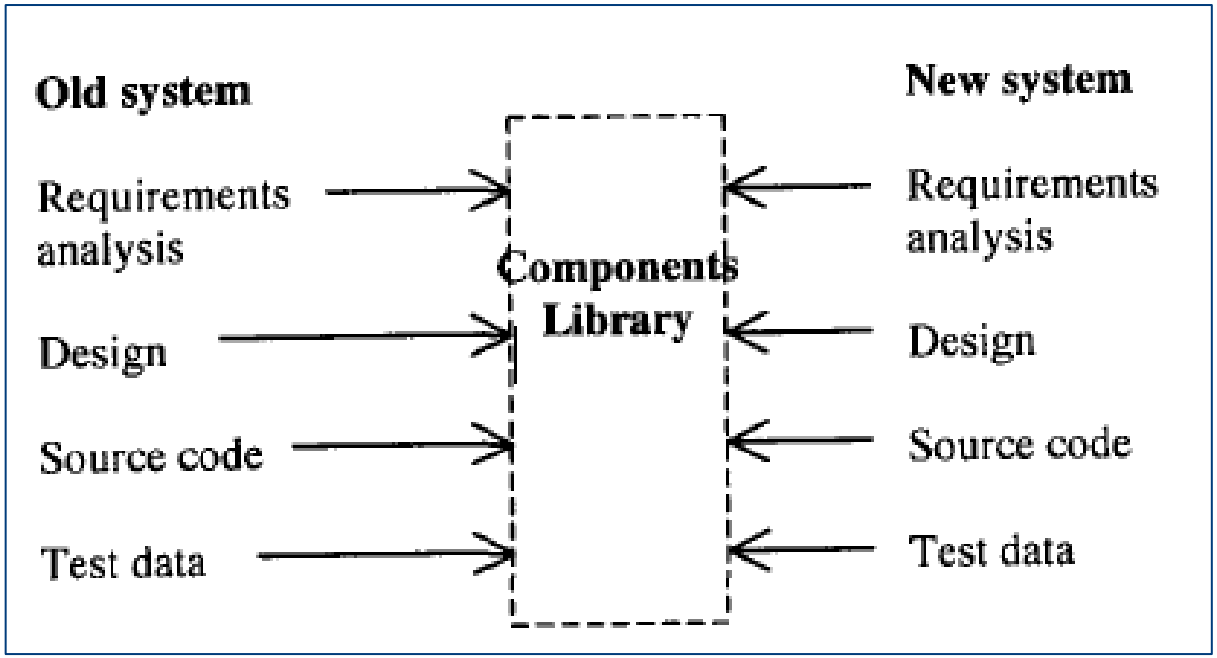
Modelo Osborne's

- Muitos **problemas técnicos** que surgem durante a manutenção são devidos a **comunicações e controle de gestão inadequados**, e recomenda uma estratégia que inclui:
  - a inclusão de requisitos de manutenção na especificação da mudança
  - programa de garantia de qualidade de software que estabelece a garantia da qualidade dos requisitos
  - avaliação de desempenho para fornecer feedback aos gerentes



Modelo Orientado a Reuso

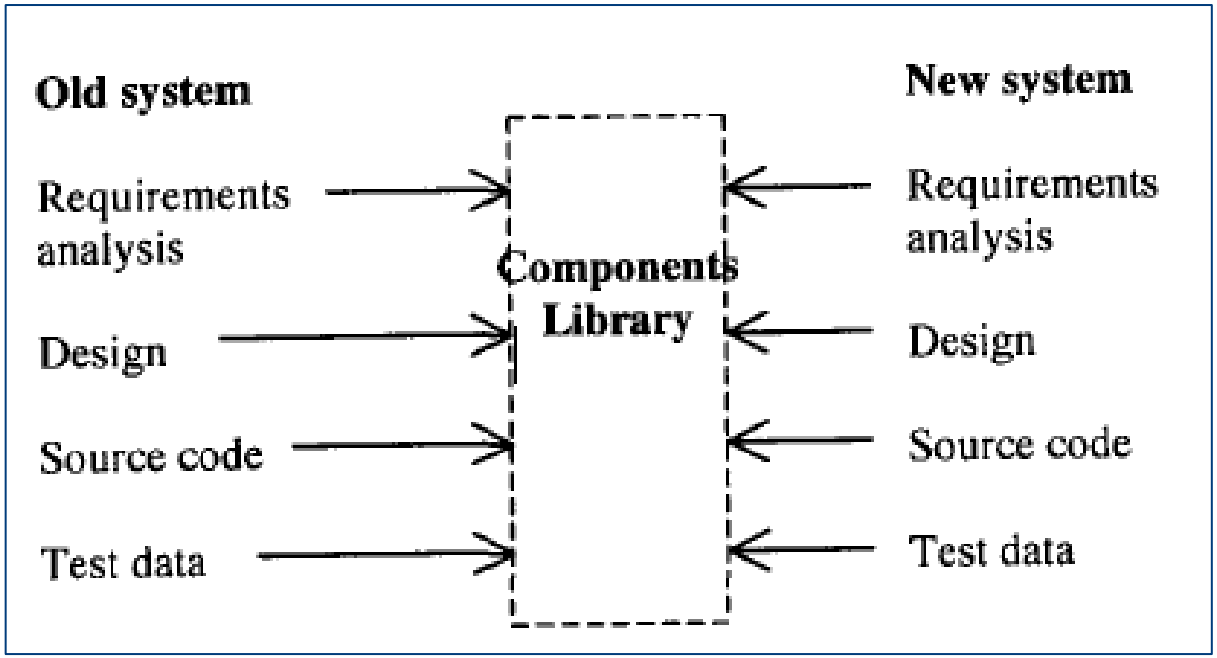
- É baseado no princípio de que a manutenção pode ser vista como uma **atividade que envolve a reutilização de componentes de programas existentes**



- O modelo de reutilização possui quatro principais etapas:
  - Identificação das partes do sistema antigo candidatos a reutilizar
  - Entendimento das partes do sistema
  - Modificação das partes do sistema antigo adequando para os novos requisitos
  - Integração das partes alteradas no novo sistema

Modelo Orientado a Reuso

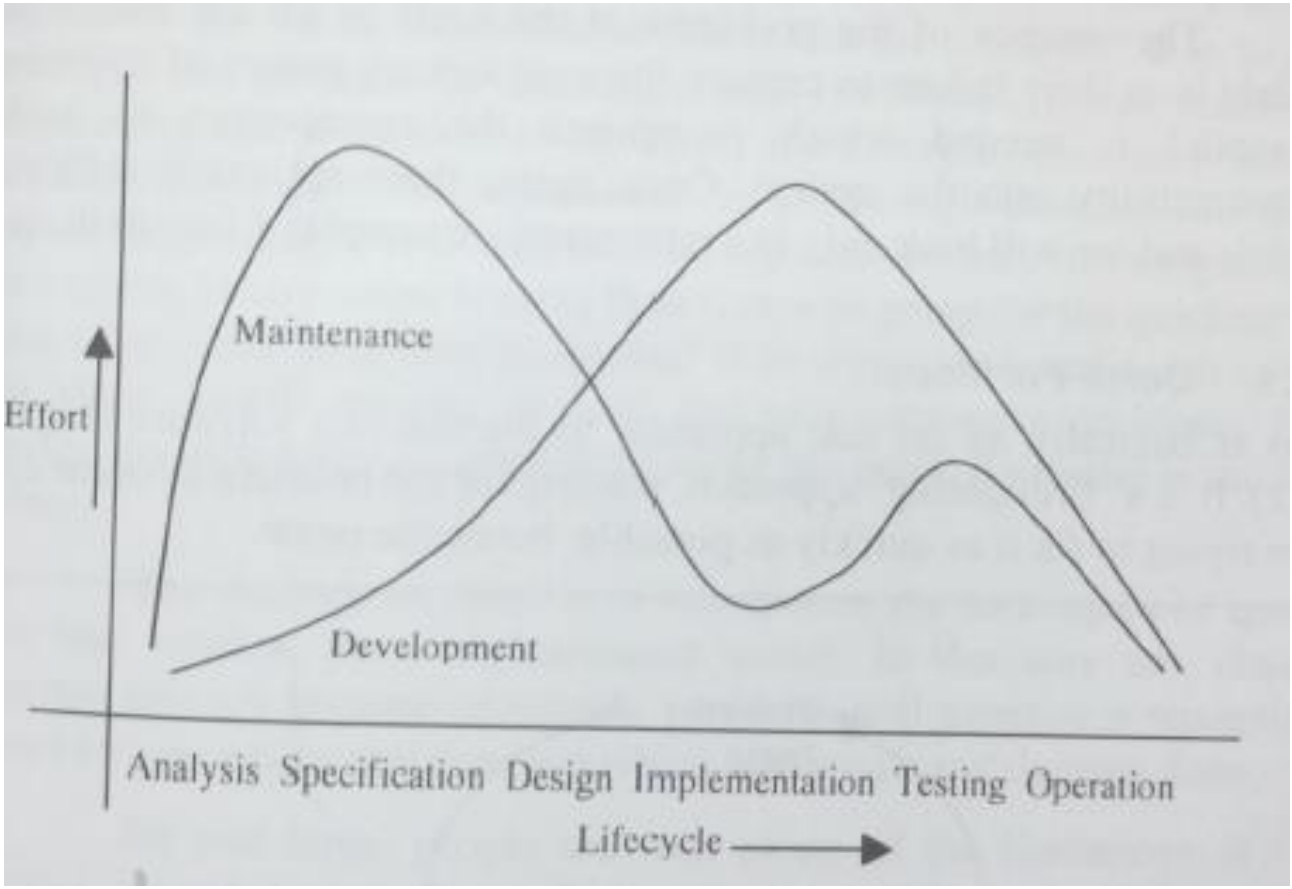
- É baseado no princípio de que a manutenção pode ser vista como uma **atividade que envolve a reutilização de componentes de programas existentes**



- A premissa de reutilização é baseada na utilização de componentes, onde é necessário que estes **componentes possuam documentações completas** e detalhadas para a **classificação de componentes e análise de possíveis mudanças**
- Permite a **reaplicação de conhecimentos de um sistema** em um **sistema semelhante**, no sentido de reduzir os esforços de desenvolvimento e manutenção do sistema

Exercício

- Interpretar o gráfico abaixo sobre esforço de Desenvolvimento x Manutenção



Exercício

- Compare e contraste o processo de manutenção do modelo de Osborne com os outros modelos de processo de manutenção mostrados na aula de hoje

Exercício

- Descreva como o sistema ACME Health Clinic pode ser modificados de forma mais eficaz. Assuma o mesmo prazos apertados, mas investigue a incorporação do modelo de correção rápida em outro modelo mais estruturado

Quando fazer uma mudança?

- Até agora, a discussão tem sido sobre a introdução de mudanças em um sistema, sem considerar se essas mudanças devem ou não ser feitas
- Em outras palavras, as maneiras como diferentes modelos abordam a implementação da mudança foram consideradas, mas sem abordar a importante questão de como decidir quando uma mudança deve ser feita
- Não se pode simplesmente **presumir que todos os envolvidos com um sistema, desde os desenvolvedores aos usuários, podem lançar suas ideias e implementá-las automaticamente**
  - Nem todas as mudanças são viáveis
  - Uma mudança pode ser desejável, mas muito cara
- Deve haver um meio de decidir quando implementar uma mudança

Exercício

- Você é o gerente de TI encarregado de um grande sistema de software de biblioteca que falhou inesperadamente em uma manhã de segunda-feira. Como você faria para resolver este problema
  - 1o. se for obrigatório que esteja instalado e funcionando em duas horas?
  - 2o. se a biblioteca é capaz de funcionar adequadamente por vários dias sem seu sistema de software?

Atividades de Desenvolvimento x Manutenção

- Processos de manutenção possuem as atividades e técnicas necessárias para modificar um sistema existente
- Muitas atividades da manutenção são similares as atividades do desenvolvimento (ex: projeto, implementação, etc)
- Algumas práticas são específicas da manutenção

Exercício

- Apresente práticas de desenvolvimento específicas da manutenção

Práticas de Manutenção

- Compreensão de programas: atividades para entender o software existente
- Refatoração de código
- Solicitação de mudanças: equipe para estimar priorização, custo (ex: tamanho, esforço, complexidade)
- Análise de impacto: áreas impactadas pela mudança em potencial

Atividades de Suporte à Manutenção

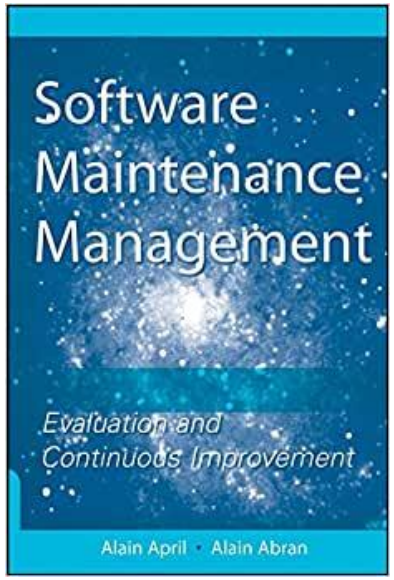
- Gerência de configuração
- Verificação, validação e teste
- Garantia de qualidade (métricas)

Técnicas de Manutenção

- Reengenharia
- Engenharia Reversa
- Migração

Referências Bibliográficas

- Grubb, P., & Takang, A. A. (2003). Software maintenance: concepts and practice. World Scientific.
- Software Maintenance Management: Evaluation And Continuous Improvement. APRIL, ALAIN; ABRAN, ALAIN



- **Perguntas?**

E-mail: jacilane.rabelo@ufc.br