DCC / ICEx / UFMG

Evolução de Software

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

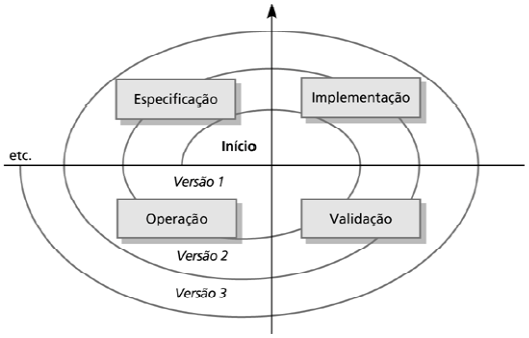
Evolução de Software

- Depois de implantados, sistemas devem inevitavelmente mudar para permanecerem úteis
- Mudanças no ambiente requerem atualizações do sistema
  - Ao implantar um sistema modificado, este sistema causa uma mudança no ambiente

Evolução de Software

- Sistemas de software geralmente têm vida útil muito longa
  - Organizações são dependentes dos sistemas que custaram milhões ($$$)
- O Modelo Espiral reflete os ciclos contínuos e ininterruptos de desenvolvimento e evolução

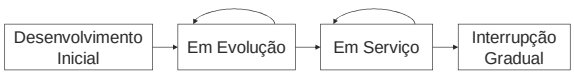
Modelo Espiral de Evolução



Fases de um sistema...

- Desenvolvimento Inicial
  - Primeira solicitação do cliente
- Em Evolução
  - Sistema é intensamente usado
- Em Serviço
  - Sistema é pouco usado
- Interrupção gradual
  - Empresa considera a substituição do sistema

Em Evolução vs. Em Serviço



- Em Evolução
  - Mudanças significativas são feitas tanto na arquitetura quando nas funcionalidades
  - A estrutura tende a gradativamente se degradar
- Em Serviço
  - Apenas mudanças pequenas e essenciais ocorrem (software é pouco usado)

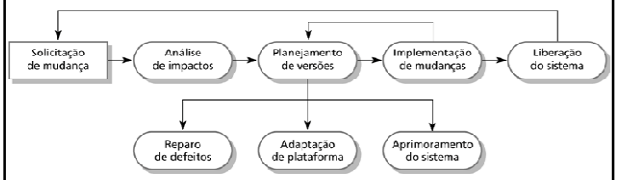
Processos de Evolução

Processo de Evolução

- A evolução pode variar entre organizações
  - **Processo informal**: solicitação de mudanças provém de conversas entre usuários e desenvolvedores
  - **Processo formalizado**: documentação produzida e avaliada em cada atividade do processo

Modelo de Processo

- Cada solicitação de mudança é avaliada, planejada e implementada
  - O processo se repede em novas solicitações



Análise e Planejamento

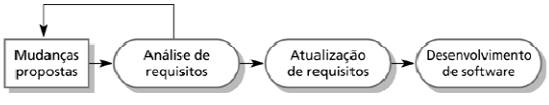
- Análise de Impactos
  - Avalia o custo e a porção do sistema afetado pela mudança solicitada
  - Decide se a mudança será aceita
- Planejamento de Versões
  - Caso sejam aceitos, diferentes tipos de mudanças podem ser agrupados para a próxima versão
  - Correção de defeitos, adaptações de plataforma e aperfeiçoamentos

Implementação e Liberação

- Implementação de Mudanças
  - Geralmente, uma nova versão inclui um conjunto com várias mudanças solicitadas
- Liberação do Sistema
  - Uma nova versão é liberada após validação com o cliente
- O processo se repete com um novo conjunto de solicitação de mudanças

Implementação da Mudança

- A implementação da mudança deve atualizar a documentação correspondente
  - Especificação, modelos de projeto, implementação, testes, etc.



Compreensão de Programa

- A implementação da mudança envolve atividades semelhante ao desenvolvimento inicial do sistema
  - Uma diferença fundamental é ser necessário compreender o programa
- Compreensão inclui
  - Entender como a funcionalidade a ser alterada encontra-se implementada
  - Avaliar o impacto da mudança em outras partes do programa

Situações Emergenciais

- Algumas solicitações de mudanças podem ser urgente, exemplo:
  - Se ocorrer um defeito grave que impede o funcionamento normal do sistema
  - Alterações do ambiente impedem o funcionamento do sistema
  - Alteração no negócio do cliente ou uma nova legislação
  - Entrada de um novo concorrente no mercado

Correção de Emergência

- Pode não haver tempo hábil para atualização da documentação
  - Ao invés de seguir as atividades "normais" de desenvolvimento, a alteração é feita diretamente no código

Refazer a Correção

- Código para correção de emergência geralmente tem baixa qualidade
- Portanto, a correção de emergência deveria ser idealmente re-implementada
  - E a documentação correspondente atualizada
- Na prática, este re-trabalho tem baixa prioridade e acaba esquecido

Dinâmica da Evolução (Leis de Lehman)

M. M. Lehman. **Rules and Tools for Software Evolution Planning and Management**. Annals of Software Engineering, 2001.

Leis de Lehman

- O crescimento e a evolução de vários sistemas de grande porte foram examinados
- Objetivo
  - Definir uma teoria unificada para evolução de software
- Resultados
  - Um conjunto de oito leis que "governam" a evolução de sistemas

1 - Mudança Contínua

- Sistemas devem ser continuamente adaptados ou eles se tornam progressivamente menos satisfatórios
  - O ambiente muda, novos requisitos surgem e o sistema deve ser modificado
  - Após o sistema modificado ser re-implantado, ele muda o ambiente

2 - Complexidade Crescente

- A medida que um sistema evolui, sua complexidade aumenta, a menos que seja realizado esforço para mantê-la ou diminuí-la
  - Manutenções preventivas são necessárias
  - Precisa de recursos extras, além dos necessários para implementar as mudanças

3 - Auto-Regulação

- A evolução de software é um processo auto-regulável
  - Atributos do sistema como tamanho, tempo entre versões e número de erros reportados é quase invariável em cada versão do sistema
- Consequência de fatores estruturais e organizacionais

4 - Estabilidade Organizacional

- Durante o ciclo de vida de um programa, sua taxa de desenvolvimento é quase constante
  - Independe de recursos dedicados ao desenvolvimento do sistema
  - Alocação de grandes equipes é improdutivo, pois o *overhead* de comunicação predomina

5 - Conservação de Familiaridade

- Durante o ciclo de vida de um sistema, mudanças incrementais em cada versão são quase constantes
  - Um grande incremento em uma release leva a muitos defeitos novos
  - A release posterior será dedicada quase exclusivamente para corrigir os defeitos
- Ao orçar grandes incrementos, deve-se considerar as correções de defeitos

6 - Crescimento Contínuo

- O conteúdo funcional de sistemas devem ser continuamente aumentado para manter a satisfação do usuário
  - A cada dia, o usuário fica mais descontente com o sistema
  - Novas funcionalidades são necessárias para manter a satisfação do usuário

7 - Qualidade Declinante

- A qualidade de sistemas parecerá estar declinando a menos que eles sejam mantidos e adaptados às modificações do ambiente

8 - Sistema de Feedback

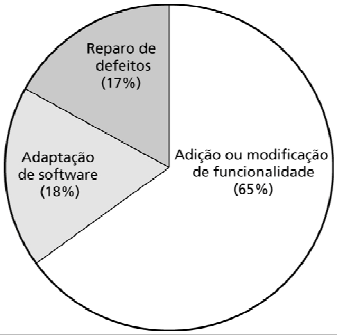
- Os processos de evolução incorporam sistemas de *feedback* com vários agentes e *loops*
  - Estes agentes, *loops* e *feedback* devem ser considerados para conseguir melhorias significativas do produto

Manutenção de Software

Manutenção de Software

- Processo geral de mudanças depois que o sistema é entregue
- Três tipos principais de manutenção
  - **Manutenção corretiva**: mudanças para reparo de defeitos de software
  - **Manutenção adaptativa**: mudanças para adaptar o software a outro ambiente
  - **Manutenção evolutiva**: mudanças para adicionar funcionalidade ao sistema

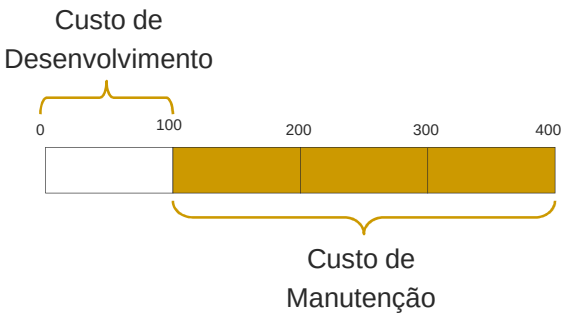
Distribuição do Esforço



Custos de Manutenção

- O custo de manutenção é geralmente muito maior que o custo de desenvolvimento
- Cada vez menos sistemas são desenvolvidos “do zero”
  - Sistemas são desenvolvidos/adaptados a partir de outros sistemas
- Faz mais sentido considerar desenvolvimento e manutenção como atividades contínuas

Desenvolvimento x Manutenção

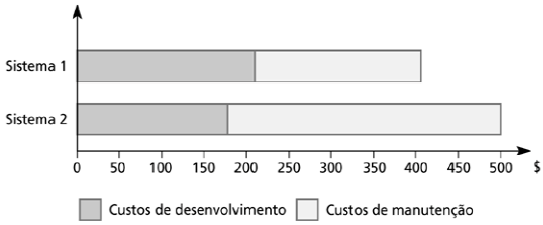


Investir no Desenvolvimento

- É geralmente benéfico investir esforço no desenvolvimento para reduzir custos de manutenção
  - É mais caro adicionar a funcionalidade depois que o sistema entra em operação
- Exemplos de investimento
  - Especificação precisa do software
  - Uso de ferramentas e orientação a objetos
  - Gerência de configuração, etc.

Retorno do Investimento

- Sistema 1 têm investimento maior na fase de desenvolvimento



Fatores que Elevam o Custo

- Vários fatores estão associados aos elevados custos de manutenção
  - Estabilidade da equipe
  - Responsabilidade contratual
  - Habilidade do pessoal
  - Idade e estrutura do programa

Equipe e Contrato

- Estabilidade da equipe
  - Os custos de manutenção são reduzidos se o mesmo pessoal estiver envolvido no projeto por algum tempo
- Responsabilidade contratual
  - Os desenvolvedores de um sistema podem não ter responsabilidade contratual pela manutenção
  - Portanto, não há incentivo para desenvolver pensando em mudanças futuras

Habilidade e Estrutura do Programa

- Habilidade do pessoal
  - A atividade de manutenção é sub-valorizada
  - Assim, o pessoal da manutenção geralmente é inexperiente e tem conhecimento limitado de domínio
- Idade e estrutura do programa
  - À medida que os programas envelhecem, sua estrutura é degradada
  - Se torna mais difícil de ser compreendida e modificada

Bibliografia da Aula

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Capítulo 9 Evolução de Software (Seções 9.1 a 9.3)