Universidade Presbiteriana Mackenzie

Processo Verificação e Validação

Ana Claudia Rossi

Faculdade de Computação e Informática

Tópicos Abordados

- Introdução
- Processo V & V
- Planejamento de Verificação e Validação
- Inspeção de Software
- Análise estática
- Verificação e Métodos Formais

Garantia de Qualidade de Software

- É um conjunto de atividades técnicas aplicadas durante todo o processo de desenvolvimento
  - O objetivo é garantir que tanto o processo de desenvolvimento quanto o produto de software atinjam níveis de qualidade especificados
  - VV&T - Verificação, Validação e Teste

No Processo de Engenharia de Software

- Essas atividades começam desde a fase de levantamento dos requisitos
- Continuam ao longo da fase de projeto através de revisões
- Utilizam-se as inspeções de código na fase de desenvolvimento e implementação
- E, ao final, são realizados testes do produto

No Processo de Engenharia de Software

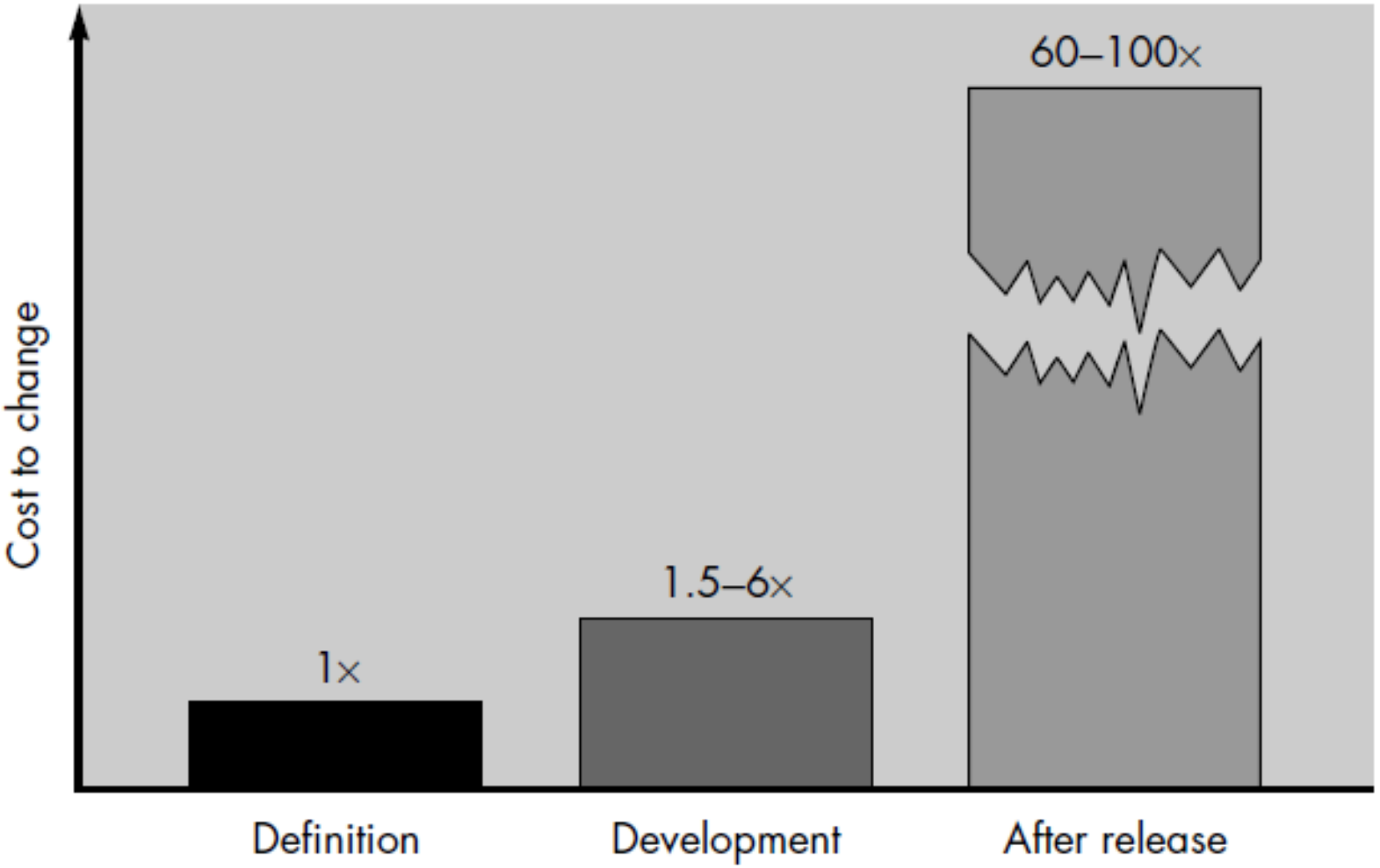
- A maior parte é de origem humana
- São gerados na comunicação e na transformação de informações
- Permanecem presentes nos diversos produtos de software produzidos e liberados. (+/- 10 defeitos a cada 1000 linhas de código)
- A maioria encontram-se em partes do código raramente executadas

No Processo de Engenharia de Software

- Quanto antes a presença do defeito for revelada, menor o custo de correção do defeito e maior a probabilidade de corrigi-lo corretamente

- Resumindo
  - Tradução incorreta de informações
  - Introduzir atividades de VV&T ao longo de todo o ciclo de desenvolvimento

Impacto da Mudança - Custo



Verificação e Validação

Verificação X Validação

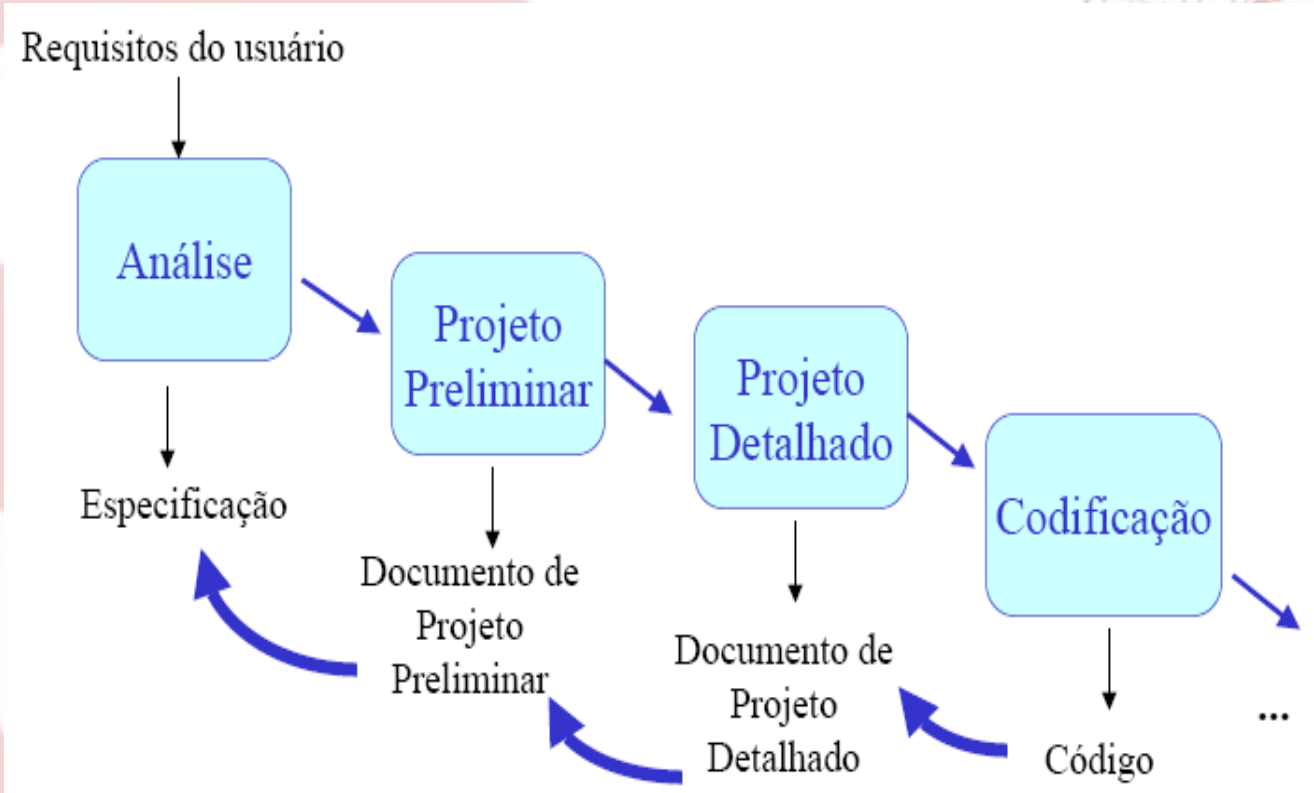
- Verificação:
  - ”Estamos construindo certo o produto?“
  - O software cumpre com suas especificações

- Validação:
  - ”Estamos construindo o produto certo?"
  - O software deve estar de acordo com o que o usuário deseja

Asseguram que o software cumpra com suas especificações e atenda às necessidades dos usuários

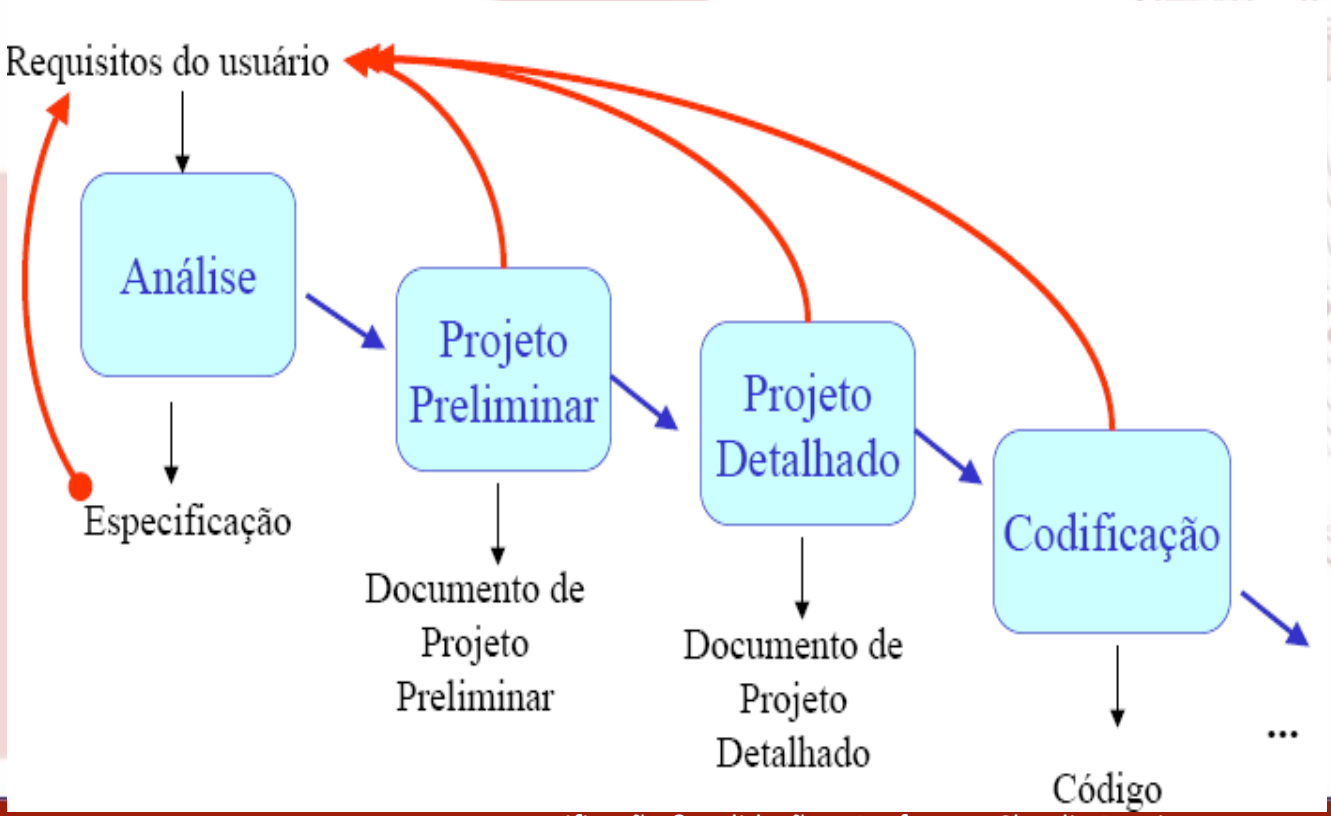
Verificação

- Estamos construindo o produto corretamente?
  - Verificar se o software está de acordo com as especificação
  - Verificar se atende aos requisitos funcionais e não-funcionais
  - Um software deve realizar as suas tarefas de forma satisfatória e correta



Validação

- Estamos construindo o produto correto?
  - Um processo mais geral
  - Assegurar que o sistema atende às expectativas do cliente
  - Mostra que o software realiza o que o cliente espera que ele faça
  - Um software deve fazer o que ele foi especificado para fazer

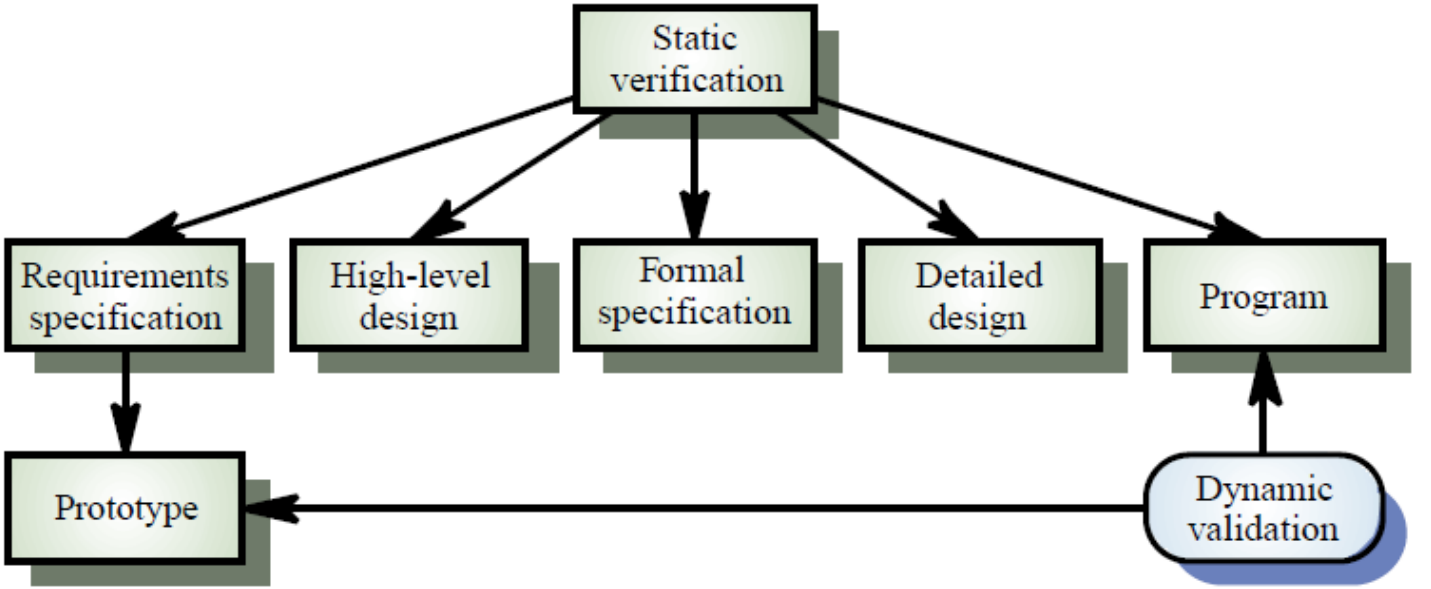


Processo V & V

- É um processo que engloba todo o ciclo de vida
  - V & V deve ser aplicado em cada estágio no processo de desenvolvimento
- Tem dois objetivos principais:
  - a descoberta de defeitos no sistema
  - Assegurar se o sistema é ou não utilizável em uma situação operacional

Verificação e Validação - Dinâmica e Estática

- *Inspeções de software - preocupadas com a* análise estática das representações do sistema para descobrir problemas (V&V estática))
  - pode ser complementadas por alguma análise automática do texto de origem de um sistema ou dos documentos associados
- *Teste de software - preocupado com a execução* e observação do comportamento do produto (V&V dinâmica)
  - O sistema é executado com dados de teste e o seu comportamento operacional é observado



Teste de Software

- Pode revelar a presença de erros e NÃO a Ausência
- Um teste bem sucedido é um teste que descobre um ou mais erros
- É a única técnica de validação para requisitos não funcionais (desempenho, confiabilidade)
- Deve ser usado em conjunto com a verificação estática para uma cobertura total das atividades de V&V

Tipos de Teste

- Os testes de defeito
  - Testes projetados para descobrir defeitos no sistema
  - Um teste bem sucedido é aquele que revela a presença de defeitos em um sistema

- Testes estatísticos
  - Usado para testar o desempenho e a confiabilidade do programa
  - Confiabilidade: número de falhas no sistema, etc
  - Desempenho Tempo de execução, tempo de resposta, etc

Metas V & V

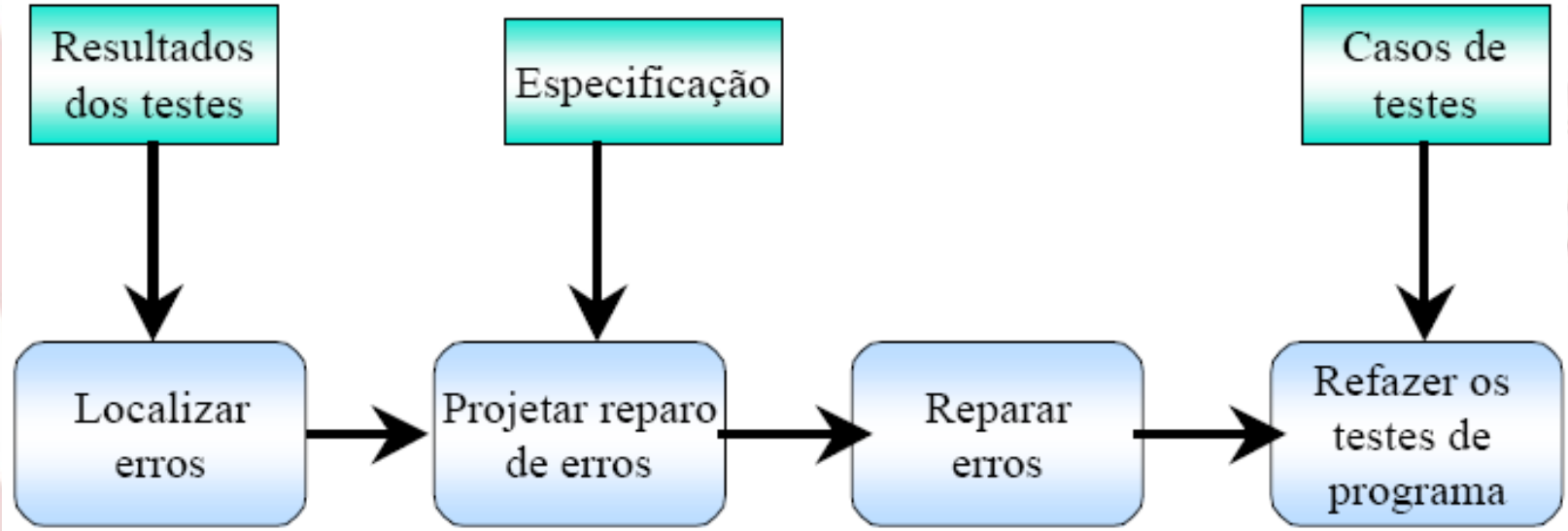
- Verificação e validação devem estabelecer a confiança de que o software é “adequado a seu propósito”
- Isso NÃO significa que o programa tenha que ser livre de defeitos
- Ao invés disso, significa que o sistema deve ser suficientemente bom para o uso pretendido. O tipo de uso irá determinar o grau de confiança que será necessário.

Confiança de V & V

- Depende do propósito do sistema, as expectativas do usuário e o ambiente de mercado.
  - Função do software
    - O nível de confiança depende do tipo de sistema e o quanto é importante para a organização.
  - Expectativas do usuário
    - Usuários podem ter poucas expectativas de certos tipos de software
  - Ambiente de mercado
    - Colocar um produto no mercado pode ser mais importante do que encontrar todos os defeitos no programa.

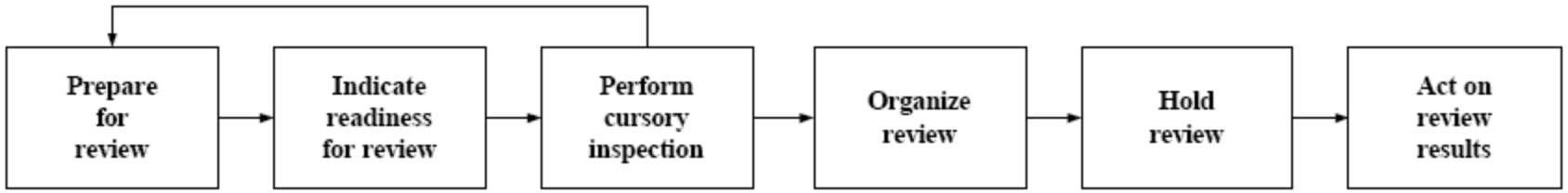
Testes e depuração

- Testar e depurar um programa são atividades distintas
- Verificação e validação e teste estão preocupados em estabelecer a existência de defeitos em um programa
- Depuração está preocupada com a localização e remoção desses defeitos
- Depuração envolve a formulação de uma hipótese sobre o comportamento do programa e testar essa hipótese para encontrar o erro no sistema



Inspeção

- Envolve pessoas examinando uma representação de software para descobrir anomalias e defeitos
- Não há a necessidade de execução do sistema, assim pode ser usada antes da implementação
- Pode ser aplicada a qualquer representação do sistema (requisitos, projeto, dados de teste, etc)



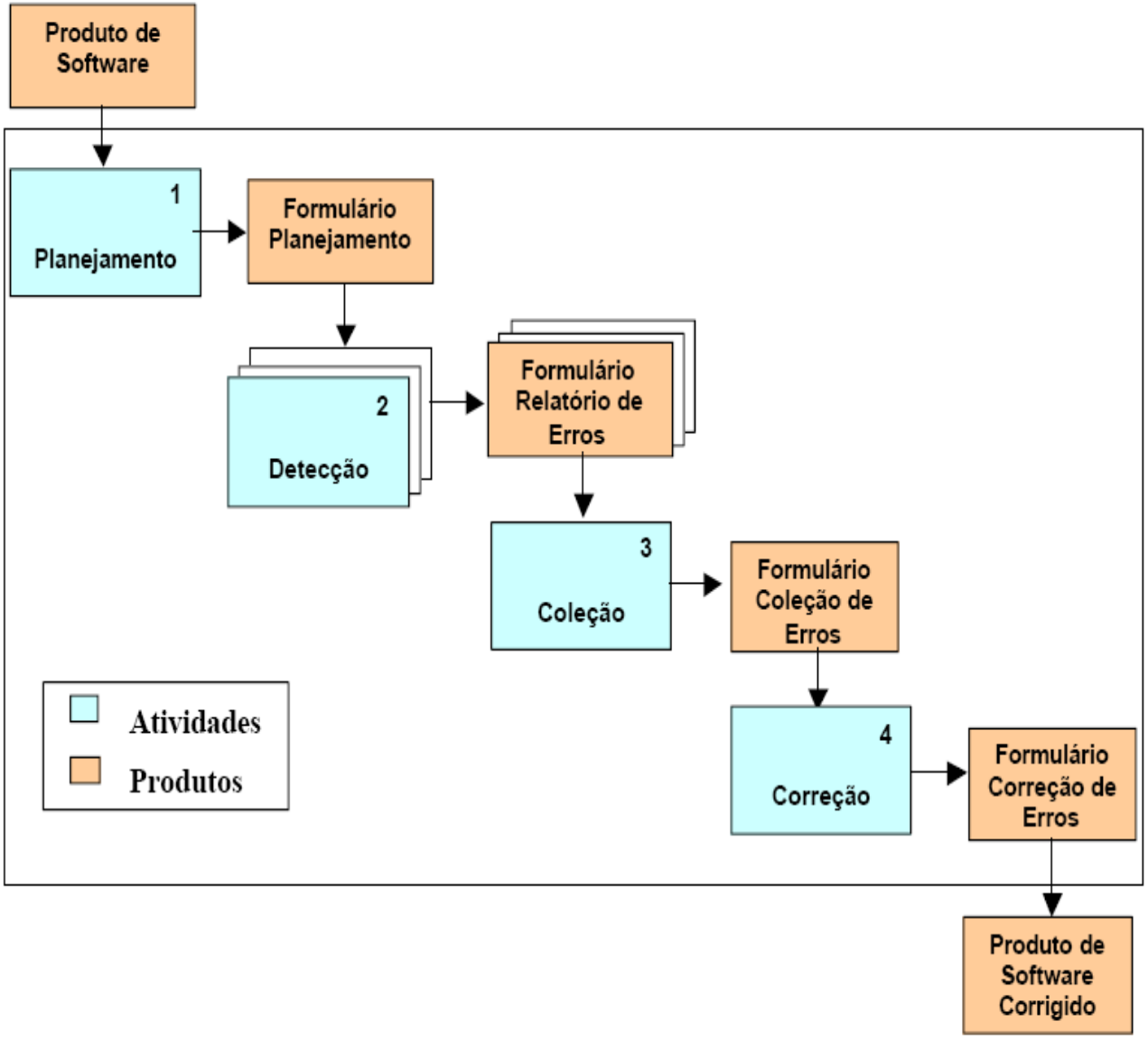
Inspeção

- Técnica muito eficaz para a descoberta de erros
  - 60% a 90% dos defeitos encontrados
- Método de análise estática para verificar propriedades de qualidade de produtos de software.
- Características
  - Processo estruturado e bem definido
  - A equipe de inspeção consiste, geralmente, de pessoal técnico
  - Os participantes possuem papel bem definido
  - Os resultados da inspeção são registrados

Inspeção

- Vantagens
  - Não há a necessidade de se preocupar com a interação entre erros
    - Defeito mascarado
  - Versões incompletas do software
    - Custo na geração de conjunto de testes especializados
  - Atributos de qualidade de software
    - Padrões organizacionais
    - Portabilidade
    - Facilidade de manutenção
    - Algoritmos ineficientes ou inapropriados

Processo de Inspeção



Processo de Inspeção

- Existe uma relutância entre Teste de Software e Inspeção de Software
  - Experiência dos Engenheiros de Software
  - Conflitos gerenciais dos custos iniciais (inspeção) e finais (testes)

- Pesquisas mostram vantagens em inspeções
  - Sua organização leva tempo
  - Evita o tempo de depuração

Inspeção de Programa

- Revisão cuidadosa, linha por linha, do código fonte do programa
- Objetivo é a detecção de defeitos (não correção)
- Defeitos podem ser erros lógicos, anomalias no código que podem indicar uma condição errônea (por ex. uma variável não inicializada) ou a não conformidade com padrões organizacionais

Inspeção de Programa

- Uma especificação precisa deve estar disponível
- Os membros da equipe de inspeção devem estar familiarizados com os padrões organizacionais
- Código sintaticamente correto deve estar disponível
- Uma lista de erros deve ser preparada
  - Checklist de possível problemas

Inspeção de Programa

- Checklist
  - Defeitos de dados
    - Inicialização das variáveis?
    - Índices dos vetores?
  - Defeitos de controle
    - Condição está correta?
    - Repetição com término?
  - Defeitos de I/O
    - Variáveis de entrada e de saída são usadas?
  - Defeitos de interface
    - Quantidade de parâmetros?
    - Tipos dos parâmetros?
  - Etc.

Inspeção de Programa

Fault class	Inspection check
Data faults	Are all program variables initialised before their values are used? Have all constants been named? Should the lower bound of arrays be 0, 1, or something else? Should the upper bound of arrays be equal to the size of the array or Size -1? If character strings are used, is a delimiter explicitly assigned?
Control faults	For each conditional statement, is the condition correct? Is each loop certain to terminate? Are compound statements correctly bracketed? In case statements, are all possible cases accounted for?
Input/output faults	Are all input variables used? Are all output variables assigned a value before they are output?
Interface faults	Do all function and procedure calls have the correct number of parameters? Do formal and actual parameter types match? Are the parameters in the right order? If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	If a linked structure is modified, have all links been correctly reassigned? If dynamic storage is used, has space been allocated correctly? Is space explicitly de-allocated after it is no longer required?
Exception management faults	Have all possible error conditions been taken into account?

Inspeção de Programa

- Gerente deve aceitar que a inspeção aumentara os custos no começo do processo de software

- Algumas organizações abandonaram o Teste de Componentes
  - As inspeções são mais eficientes
  - Inspeção de Componentes + Teste de Sistema

Análise Estática Automatizada

- Inspeção é uma Análise Estática
- Alguns defeitos podem ser identificados automaticamente
  - Automatização do processo com indicação de possíveis erros
  - Complementam as atividades do compilador
- As anomalias não são necessariamente defeito!
  - Apenas enfatizam o que poderia sair errado

Análise Estática Automatizada

- Defeitos de dados
  - Atribuição dupla, sem uso intermediário
  - Declaração sem uso
- Defeitos de controle
  - Código inacessível
- Defeitos de I/O
  - Variáveis de output usada duas vezes sem intervenção
- Defeitos de interface
  - Tipos de parâmetros incompatíveis
  - Retorno não utilizado
- Defeitos de Gerenciamento de Armazenamento
  - Aritmética de ponteiros
- Etc.

Análise Estática Automatizada

Fault class	Static analysis check
Data faults	Variables used before initialisation Variables declared but never used Variables assigned twice but never used between assignments Possible array bound violations Undeclared variables
Control faults	Unreachable code Unconditional branches into loops
Input/output faults	Variables output twice with no intervening assignment
Interface faults	Parameter type mismatches Parameter number mismatches Non-usage of the results of functions Uncalled functions and procedures
Storage management faults	Unassigned pointers Pointer arithmetic

Inspeção

- Linguagens de tipagem forte
  - Java, C#
- LINT
  - Comando *NIX para verificação de programas C
  - Verificação estática
    - Estrutura de controle
    - Tipos de dados
    - Verificação de variáveis e funções
    - Portabilidade
    - Estilo de código ineficiente
  - http://www.pdc.kth.se/training/Tutor/Basics/lint/index-frame.html
  - http://www.opengroup.org/onlinepubs/007908775/xcu/lint.html

Inspeção

```
#include <stdio.h>
printarray (Anarray)
int Anarray;
{   printf(“%d”,Anarray);  }
main ()
{
  int Anarray[5]; int i; char c;
  printarray (Anarray, i, c);
  printarray (Anarray) ;
}
```

138% more lint_ex.c

```
#include <stdio.h>
printarray (Anarray)
  int Anarray;
{
  printf("%d",Anarray);
}
main ()
{
  int Anarray[5]; int i; char c;
  printarray (Anarray, i, c);
  printarray (Anarray) ;
}
```

139% cc lint_ex.c
140% lint lint_ex.c

```
lint_ex.c(10): warning: c may be used before set
lint_ex.c(10): warning: i may be used before set
printarray: variable # of args. lint_ex.c(4) :: lint_ex.c(10)
printarray, arg. 1 used inconsistently lint_ex.c(4) ::
lint_ex.c(10)
printarray, arg. 1 used inconsistently lint_ex.c(4) ::
lint_ex.c(11)
printf returns value which is always ignored
```

Planejamento de V & V

- Planejamento cuidadoso é necessário para obter sucesso nos processos de inspeção e teste
- O planejamento deve iniciar no começo do processo de desenvolvimento
- O processo de planejamento deve decidir sobre o equilíbrio entre as abordagens estáticas e dinâmicas
- O planejamento de testes se ocupa em estabelecer os padrões para o processo de testes

Estrutura de um Plano de teste

- O processo de teste
- Facilidade de rastreamento dos requisitos
- Itens a serem testados
- Cronograma de testes
- Procedimentos de registros de testes
- Requisitos de hardware e de software
- Restrições

Algumas influências na Qualidade

- Função
  - Tipo do sistema e domínio do problema
- Expectativa
  - Tolerância aos erros do software
- Mercado
  - Concorrência
    - Primeiro no mercado
    - Alto preço no release

Planejamento

- Análise estática
  - Não requerem a execução propriamente dita do produto
  - Podem ser aplicadas em qualquer produto intermediário do processo de desenvolvimento
    - Documento de requisitos, diagramas de projeto, código-fonte, planos de teste, ...
  - As revisões são o exemplo mais clássico de análise estática

Planejamento

- Análise dinâmica
  - Requerem a execução do produto
    - Código
    - Quaisquer outras representações executáveis do sistema
      - Especificação executável
  - As atividades de simulação e teste constituem uma análise dinâmica do produto

Planejamento

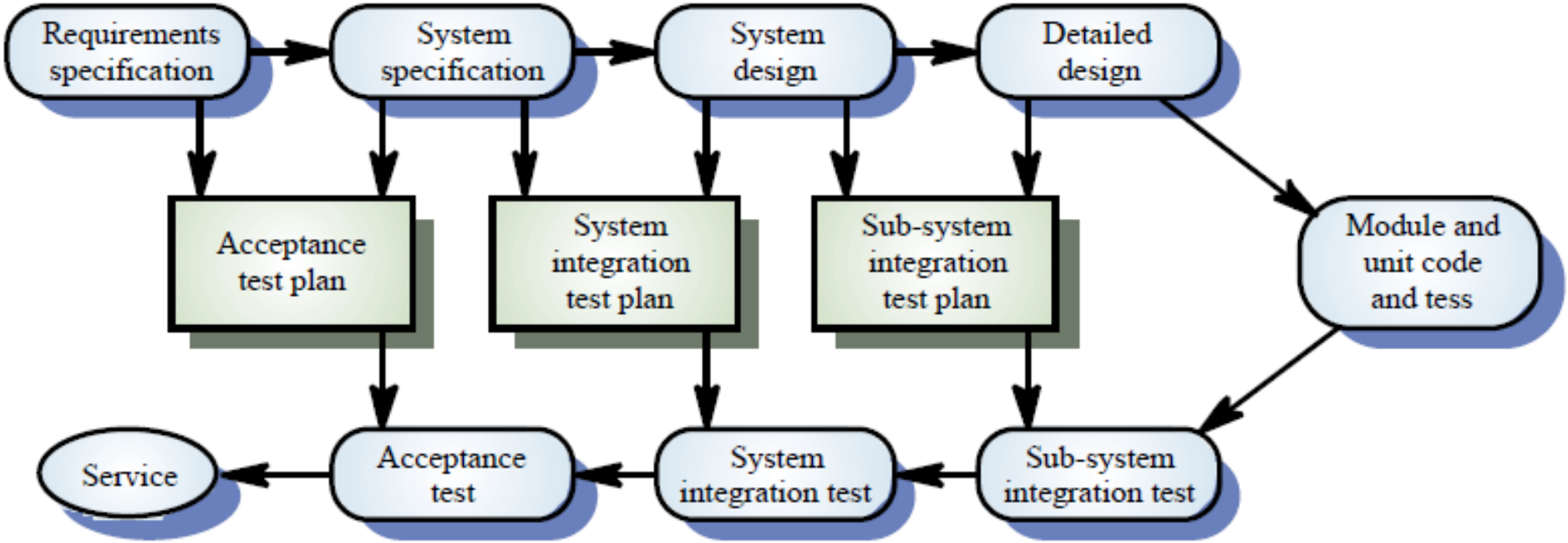
- O planejamento deve iniciar no começo do processo de desenvolvimento
  - Plano de Teste Mestre
  - Gerência de Qualidade

- O processo de planejamento deve decidir sobre o equilíbrio entre as abordagens estáticas e dinâmicas
  - Tipo de sistema
  - Habilidade organizacional

Planejamento

- O planejamento de testes se preocupa em estabelecer os padrões para o processo de testes
  - Gerentes
  - Engenheiros de software

Planejamento de V & V



MATERIAL DE ESTUDO

- Sommervile, I. Engenharia de Software
  - Capitulo Verificação e Validação
- Moodle
  - Slides de Aula

Obrigado

Ana Claudia Rossi

ana.rossi@mackenzie.br