

Verificação Estática

Outras Técnicas

Baseado em notas de aula da profa. Eliane Martins

Tópicos

- Outras Técnicas de Análise Estática
- Ferramentas
- O Processo Sala Limpa

Referências

- I. Sommerville. “Software Engineering”. 6^a ed, 2001, cap 19.3-4
- R.C.Linger. “Cleanroom Process Model”. IEEE Sw, mar/94, pp50-58.
- S.J.Zeil. Verification and Validation. Notas de Curso, 1999. Obtido em fev/2003:www.???
- R.S.Pressman. “Software Engineering. A Practitioner’s Approach”. 4^a edição, 1997, c. 25.

Outras Técnicas

- Leitura
- Prova matemática
- Análise estática automatizada
- O Processo Sala Limpa (Cleanroom Process)

Leitura

- O produto é lido por outra pessoa
- fácil de realizar
- eficaz quando o leitor for alguém ligado ao projeto (projetista, equipe de testes, equipe de controle de qualidade)
- resultados dependem muito do leitor
- baixo potencial para encontrar falhas de omissão e inconsistência, especialmente para sistemas mais complexos

Verificação matemática

- A verificação é baseada em argumentos matemáticos que demonstram que a implementação é consistente com a especificação
 - a semântica da linguagem de programação deve ser definida formalmente
 - a especificação deve ser formal

Prova de correção de programas

- Provas matemáticas rigorosas de que um programa satisfaz a sua especificação são difíceis de produzir, além de serem longas.
- Histórico:
 - McCarthy (1962)
 - Floyd (1967)
 - Hoare (1973) e Dijkstra (1976)

Prova de correção

- Para que possa ser realizada é preciso
 - uma especificação formal do sw
 - uma semântica bem definida da linguagem de programação

Prova de correção: considerações finais

- Não existem provas para todos os programas. Por exemplo, programas que tratam interrupções.
- Desenvolver prova de correção tem custo alto (precisa de ferramenta) por isso não é usada na prática na maioria dos projetos de sw.

Verificação de conformidade

- Menos formal do que a prova, também pode usar argumentos matemáticos para verificar conformidade de um programa com sua especificação
 - deve demonstrar que um programa foi implementado conforme foi especificado
 - deve demonstrar que o programa termina

Análise Estática Automatizada

- Analisadores estáticos de programas:
 - ferramentas que varrem o código fonte à procura de falhas e anomalias
 - são um complemento muito útil para a inspeção
 - tipos de checagem:
 - falhas de dados
 - falhas de controle
 - falhas de E/S
 - falhas de interface
 - falhas de armazenamento

Analísadores estáticos

- Falha de dados
 - como são usadas as variáveis do programa?
 - variáveis usadas antes de serem inicializadas
 - variáveis declaradas mas que nunca são utilizadas
 - variáveis definidas duas vezes mas não utilizadas nenhuma vez entre as duas definições
 - possibilidade de índice fora de limites para arrays
 - variáveis não declaradas

Analísadores estáticos

- Falhas de controle
 - trechos de código não alcançáveis
 - desvio para interior de laços
- Falhas de entradas/saída:
 - variáveis são usadas em comandos de saída duas vezes sem que seu valor seja alterado entre uma e outra

Analísadores estáticos

- Falhas de interface
 - uso de rotinas é consistente com sua declaração?
 - tipos de parâmetros incompatíveis
 - número de parâmetros incompatível
 - valor de retorno de funções não é utilizado
 - funções ou procedimentos não são chamados
- Falhas de armazenamento
 - apontadores não inicializados
 - aritmética de apontadores

Processo Sala Limpa

- Objetivo:
 - prevenção de falhas: evitar falhas usando métodos rigorosos e precisos
- Como:
 - em hw:
 - unidades de fabricação de semi-condutores, denominadas de salas limpas (cleanroom), onde as falhas de manufatura são evitadas pela fabricação em atmosfera ultra limpa
 - em sw:
 - produção de sw livre de falhas pelo uso de métodos formais + verificação rigorosa ao longo do desenvolvimento + testes estatísticos

Processo convencional x processo sala limpa

- centrado no indivíduo x revisão por pares
- seqüencial x incremental
- projeto informal x especificação e projeto formais
- testes de unidade x verificação da correção
- testes visando cobertura x testes estatísticos visando perfil de uso
- confiabilidade ??? x medição da confiabilidade

{Zeil 2003}

Histórico

- 1970-1980: Origem das idéias
 - proposta de abordagem baseada em princípios de matemática, estatística e engenharia (Mills); programação estruturada (Dijkstra); refinamentos sucessivos (Wirth); modularidade (Parnas)
- 1987: Origem do nome
 - inspirado no processo de fabricação de semicondutores: foco na prevenção e não na remoção de falhas
- 1988: Programa STARS, do DoD, adota o processo. Mills cria a Software Engineering Technology (SET) Inc. para comercializar o processo
- 1989: Pesquisa em confiabilidade e reuso, com a definição de método para planejamento da confiabilidade incorporando informação sobre componentes a serem reutilizados

Histórico

- 1990: Pesquisa sobre estruturas de caixas com Ada
 - uso da estrutura de caixas do processo foi aplicado em um estudo de caso na linguagem Ada
- 1992: Protótipo de um framework para avaliação de componentes reutilizáveis
 - documento: Criteria and Implementation Procedures for Evaluating Reusable Software Engineering Assets,” CDRL 04014-002B, Março/1992. Uso em biblioteca de componentes (comercial)
- 1991-1992: Protótipo de um guia para o uso do processo sala limpa
 - protótipo e guia de um modelo de processo foi desenvolvido e documentado em “Process Guide and Model for the Cleanroom Engineering Process” (colaboração STARS/SEI)

{Zeil 2003}

Histórico

- 1992: Demonstração de uso do processo em projeto piloto das Forças Armadas dos EUA
- 1993: Demonstração de uso do processo em projeto piloto da Força Aérea americana
- 1993-1994: Protótipos de ferramentas de apoio ao processo, desenvolvidas pela SET

{Zeil 2003}

Histórico

- 1995. Comercialização da ferramenta para certificação
 - toolSET_Certify, criada para apoio a criação de modelos de uso e testes de certificação com base nesses modelos
 -
- 1995: Processo Sala Limpa e CMM
 - processo para reuso de componentes desenvolvido para o SEI foi revisado e mostrou ser consistente com várias das Áreas-Chave de Processo do CMM

{Zeil 2003}

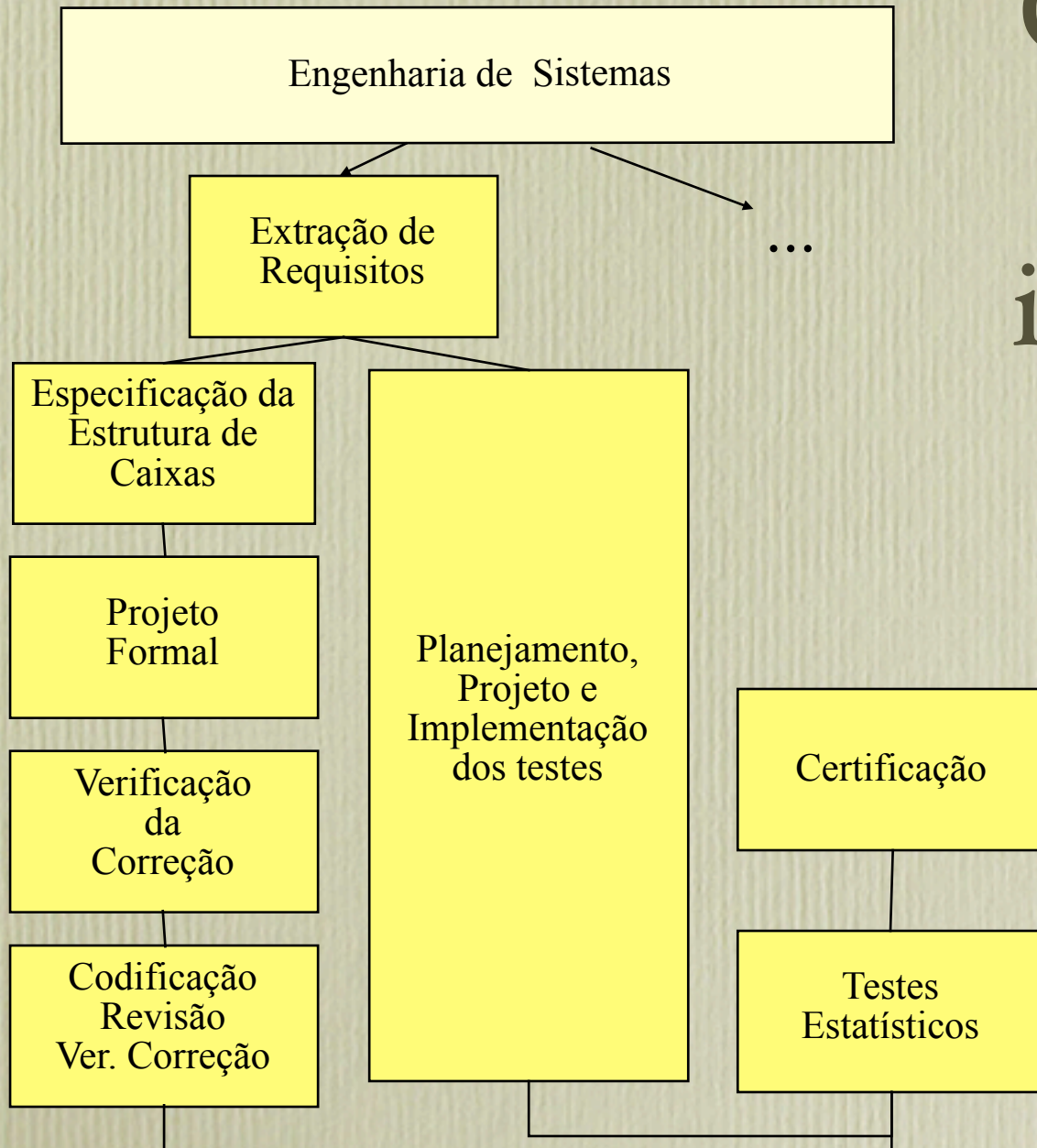
Histórico

- 1996. Integração do Processo Sala Limpa a desenvolvimento OO
 - ainda no âmbito do programa STARS
 -
- 1997: Extensões ao processo para adaptação a metodologias OO
 - Booch
 - Shlaer & Mellor
 - Objectory

Características

- O processo:
- cobre todo ciclo de vida: planejamento, medição, especificação, projeto, codificação, teste, certificação
- combina especificação e projetos formais, verificação da correção baseada em funções matemáticas, testes estatísticos para medição e certificação da confiabilidade
- é baseado no desenvolvimento incremental

O processo para um incremento



[Pressman 97]

Estratégia (I)

- Planejamento dos incrementos:
 - definem-se as funções que cada incremento deve realizar
 - planejam-se as atividades para o desenvolvimento sala limpa de cada incremento
- Extração de requisitos de cada incremento
- Especificação da estrutura de caixas:
 - especificação funcional estruturada em caixas
- Projeto formal
 - refinamentos da especificação para se obter projeto preliminar e detalhado
- Verificação da correção
 - verificação rigorosa realizada em todas as fases, podendo fazer uso de métodos formais

Estratégia (2)

- Geração, Revisão e Verificação de Código
 - as caixas são convertidas para a linguagem de programação escolhida
 - o código produzido é revisado (passeio ou inspeção)
 - verificação rigorosa da correção do código é realizada
- Planejamento dos Testes
 - definição do perfil de uso do sw (distribuição de probabilidade)
 - projeto e implementação de testes de acordo com a distribuição de probabilidades identificada
- Testes estatísticos
 - testes exercitam cada entrada de acordo com sua probabilidade de ocorrência
- Certificação
 - medição da confiabilidade para cada incremento e para o sistema

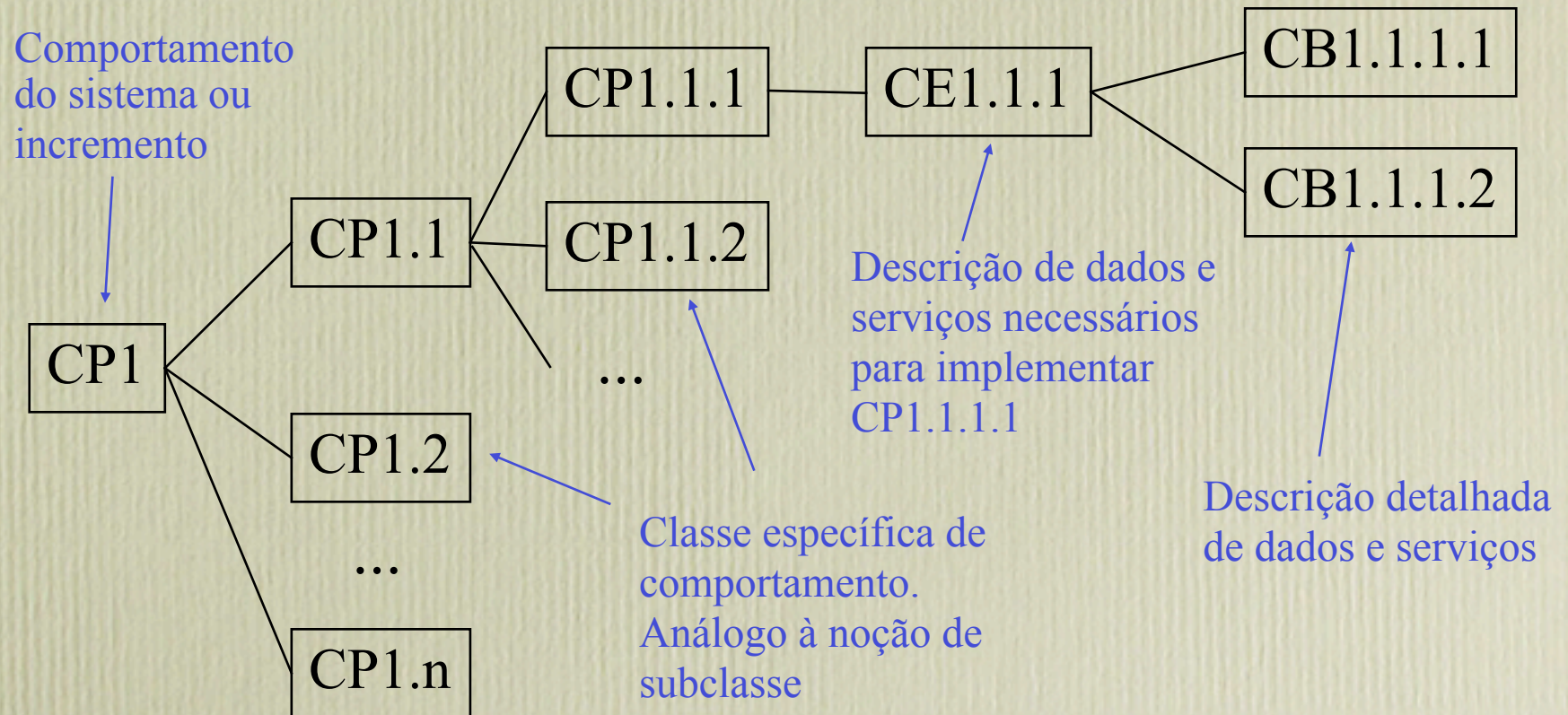
Equipes

- Equipe de especificação
 - responsável pelo desenvolvimento e pela manutenção da especificação do sistema, tanto da especificação orientada ao cliente (definição de requisitos) quanto da especificação matemática para verificação
 -
- Equipe de Desenvolvimento
 - responsável pelo desenvolvimento e verificação estática do sw
 -
- Equipe de Certificação
 - responsável pela definição do perfil de uso do sistema, bem como pelo planejamento, projeto e implementação dos testes estatísticos

Especificação Funcional

- Organizada hierarquicamente utilizando estrutura de caixas
- Caixas englobam processamento e dados (como em OO)
- Caixas vão englobando mais detalhes a medida em que avançam as fases de desenvolvimento (refinamentos sucessivos):
 - caixa preta (CP): especifica o comportamento do sistema (ou de um incremento) em termos de estímulos (ou eventos) e respostas
 - caixa de estados (CE): encapsula dados e serviços (ou operações), da mesma forma que um objeto
 - caixa branca (CB): detalha os serviços definidos na caixa de estados

Estrutura Hierárquica



Exemplo

caixa preta

```
[calcula y, a
parte inteira da
raiz quadrada de
um valor inteiro
de entrada, x]
```



caixa de estado

```
[calcula y, a parte
inteira da raiz
quadrada de um valor
inteiro de entrada, x]
do
[inicializa y]
[y recebe a parte
inteira da raiz
quadrada de x]
end
```



caixa branca

```
[inicializa y]
do
y ← 0;
[y recebe a parte inteira
da raiz quadrada de x]
while  $y^2 \leq x$ 
y ← y+1;
end
```

Exemplo com Predicados

pré: [$x \geq 0$]

[calcula y, a parte inteira da raiz quadrada de um valor inteiro de entrada, x]

pós: [x não muda e $y^2 \leq x \leq (y + 1)^2$]



pré: [$x \geq 0$]

[calcula y, a parte inteira da raiz quadrada de um valor inteiro de entrada, x]

do
[inicializa y com 0]
init: [$x \geq 0$ e $y = 0$]

[y recebe a parte inteira da raiz quadrada de x]
end

pós: [x não muda e $y^2 \leq x \leq (y + 1)^2$]

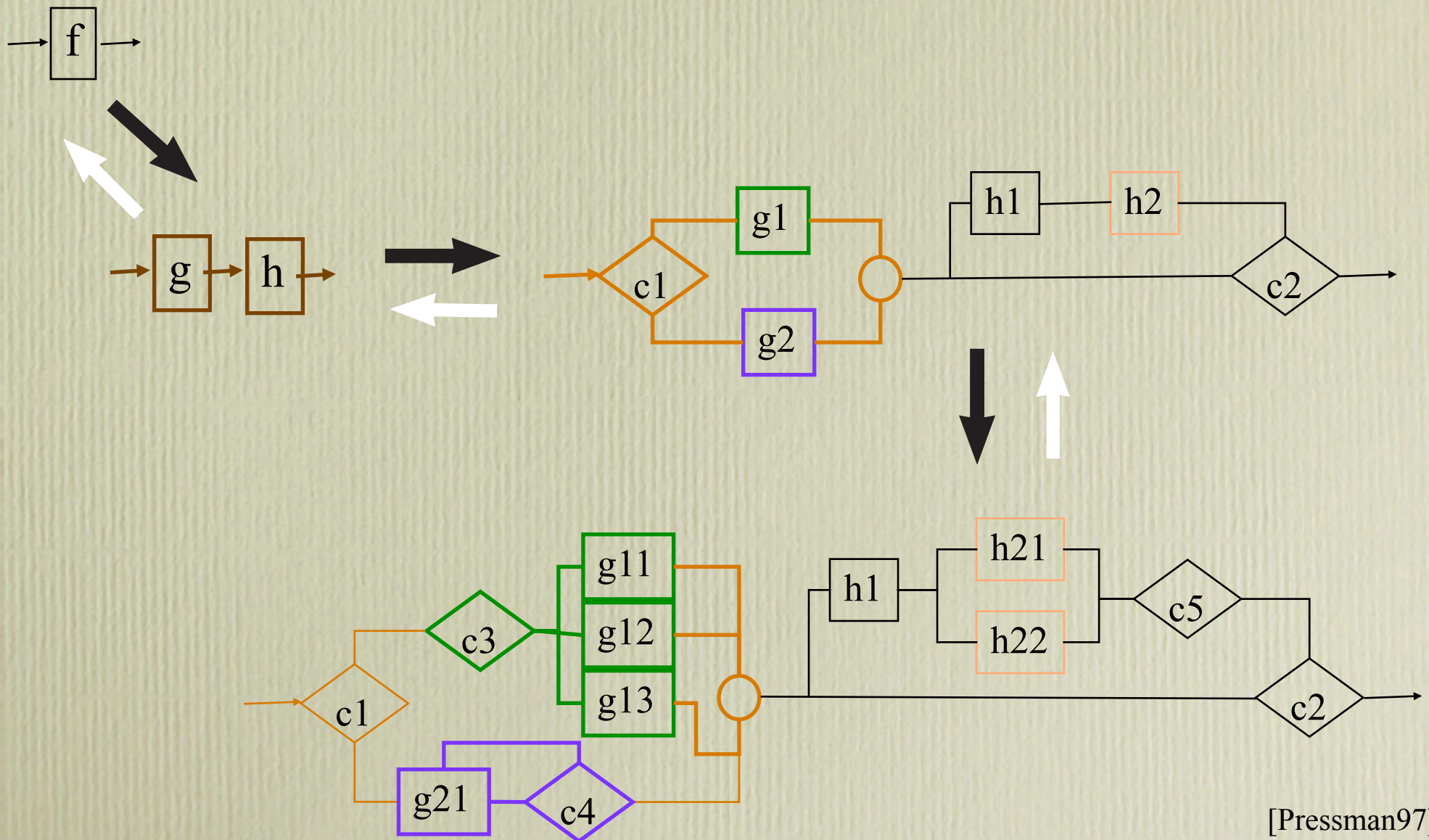


pré: [$x \geq 0$]

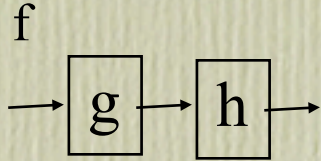
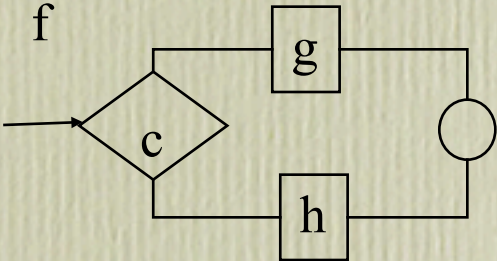
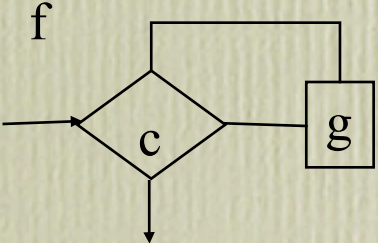
[inicializa y]
do
y \leftarrow 0;
init: $x \geq 0$ e $y = 0$
[y recebe a parte inteira da raiz quadrada de x]
loop: $y^2 \leq x$
while $y^2 \leq x$
sim: $(y+1)^2 \leq x$
y \leftarrow y+1;
cont: $y^2 \leq x$
end

pós: [x não muda e $y^2 \leq x \leq (y + 1)^2$]

Refinamento e Verificação



Verificação

Tipo	Estrutura de controle	Condição de correção
seqüência		g seguido de h realiza f ?
seleção		Quando c é verdadeira, g realiza f e quando c é falsa h realiza f ?
repetição		Quando c é verdadeira, g seguida de f realiza f ? Quando c é falsa, pular o loop ainda realiza f ? O loop termina ?

Pontos fortes do processo

- O uso desse processo tem resultado em sistemas com baixo n° de falhas e a um custo não muito elevado com relação a sistemas convencionais
- verificação estática é eficaz na descoberta de falhas antes da execução: 2,3 falhas / KLOC é a taxa durante os testes
- desenvolvimento incremental permite constante realimentação do usuário, o que acomoda melhor mudanças nos requisitos. Também permite que funções mais complexas ou mais críticas sejam desenvolvidas mais cedo e sejam validadas mais vezes
- testes estatísticos permitem medir a confiabilidade do sistema
- útil em reuso de componentes: o componente é entregue com todos os modelos, testes e medidas obtidas [Pressman97]

Pontos fracos do processo

- necessidade de equipe experiente, bem treinada e empenhada
- adoção relativamente pequena
- relatos de sucesso na indústria têm vindo, na maior parte dos casos, de projetos envolvendo seus proponentes

{Sommerville01}

Fim