Estudos de Técnicas e Desenvolvimento de Ferramentas para Slicing de Programas

Autores

m rcio Barbosa

Orientador

Waldo Luis de Lucca

Apoio Financeiro

Fapic

1. Introdução

Este trabalho tem o objetivo de apresentar a especificação de uma ferramenta que utiliza as técnicas do teste de software para manipular elementos requeridos de teste. Ela permite avaliar a possibilidade de que atributos internos, de partes do código, tenham a chance de conter defeitos e, utilizando a técnica do *slicing* de programas, selecionar estes trechos de código.

Esta técnica junto a um ambiente automatizado poderá auxiliar programadores no teste de software para reduzir o esforço e, conseqüentemente, o custo desta atividade.

2. Objetivos

Os objetivos deste trabalho estão relacionados de modo geral ao estudo da Engenharia de Software e sua metodologia, com ênfase ao teste de software e confecções de ferramentas. Há, na literatura da área de Ciência da Computação, diversos trabalhos sobre o teste de software, técnicas para *slicing* de programas e sua avaliação, verificando a eficácia de seu algoritmo e a viabilidade de automação, especificação de uma ferramenta para extração de segmentos de código de um programa utilizando as técnicas de *slicing* e, a partir destas referências, pode-se fazer a avaliação da contribuição que a ferramenta pode trazer para o aumento da eficiência do teste de software.

3. Desenvolvimento

Este projeto faz parte de um projeto mãe que visa gerar uma ferramenta que explore aspectos de complexidade dos elementos requeridos, para definir novos métodos de seleção de dados de teste e novas estratégias de seleção de dados de teste. Seu objetivo se baseia na hipótese de que um trecho de código mais complexo tem maior probabilidade de conter defeitos do que um trecho menos complexo. O propósito é de garantir a eficiência dos critérios em relação ao custo de aplicação e em relação a sua eficácia em detectar a presença de defeitos.

Na estrutura básica da ferramenta do ambiente de teste, o *slicing* de programas terá como função receber os elementos requeridos da ferramenta de teste, aplicar o *slicing* e gerar os *slices* dos elementos requeridos e preparar estes *slices* para a ferramenta de métricas que gera o cálculo da complexidade dos elementos requeridos. Aplicar a técnica do *slicing* manualmente não é viável, em função do tempo necessário para analisar estaticamente as dependências do programa, então há a necessidade de automatizar este processo.

4. Resultados

A técnica de *slicing* de programas é utilizada para melhor abstrair o programa, emprega métodos que decompõem os programas em partes menores. No projeto desenvolvido, a partir dos elementos requeridos de teste, gerados a partir da técnica de teste baseado em fluxo de dados, utilizando as técnicas de *slicing*, estas partes menores selecionadas são pedaços do algoritmo que contém as associações definição-uso de uma determinada variável. Posteriormente, podem ser coletadas algumas métricas de complexidade e de teste destes pedaços de programa e, a partir delas, pode-se fazer a comparação dos atributos deste trecho de código.

Segundo Weiser (1984), as partes menores, ou os pedaços do algoritmo são na verdade um subconjunto de um programa. A técnica de *slicing* produz um programa reduzido, um *slice*, que é uma representação fiel do programa original, dentro do domínio do subconjunto especificado de seu comportamento. Assim um programador utilizando as técnicas do teste de software em um subconjunto do programa com a técnica do *slicing*, poderá analisar partes menores do projeto e avaliar o comportamento do programa aplicando as técnicas do teste, o que pré-resume as técnicas utilizadas neste projeto.

No levantamento teórico foi discutido também a importância do grafo de dependência para utilizar as técnicas do *slicing* de programas. O estudo apresentou a característica do *slicing* com a necessidade de implementar um grafo de dependência para analisar um algoritmo e separar suas informações linha por linha transformando-as em nós do grafo. Visa também destacar que todo o processo de caminhamento do código pode ser representado no grafo de dependência, utilizando os arcos de controle e do fluxo de dados e, principalmente este último, as técnicas do *slicing* serão aplicadas, percorrendo o algoritmo e selecionando o caminho da associação definição-uso de um determinado elemento requerido.

No estudo apresentado constatou-se que, para aplicar as técnicas do grafo de dependência é necessário utilizar a técnica de compiladores para manipular as linhas do algoritmo, transformando-as em nós de um grafo.

A primeira etapa, para manipular as técnicas do *slicing*, inicia-se ao determinar um algoritmo para ser analisado. A figura 1 destaca um algoritmo proposto. Neste processo o algoritmo precisa passar por uma análise para determinar os elementos requeridos de teste e assim para cada elemento requerido aplicar as técnicas do *slicing* de programas.

Figura1. Algoritmo proposto da linguagem PDL, para executar o *slice*.

Deste algoritmo será selecionado, como elemento requerido, o uso da variável d na linha 11, tendo sido definida na linha 4. Considerando que uma das ferramentas que auxiliam o teste baseado em fluxo de dados é o Grafo de Fluxo de Controle, este elemento requerido pode ser representado por < 2 , 6 , d >, onde 2 representa o nó onde ocorre a definição (linha 4) e 6 corresponde ao nó onde ocorre a utilização da variável (linha 11). Este elemento requerido é um dos 20 elementos requeridos de teste para o algoritmo em questão (a geração da lista completa de elementos requeridos foi feita com ou uso de outra ferramenta).

A próxima etapa consiste em criar o grafo de dependência para analisar as dependências deste algoritmo e a partir do elemento requerido < 2 , 6 , d >, aplicar uma seleção na definição e todo o uso da variável “d”, dentro do algoritmo percorrido, como critério de corte, e obter o *slice*.

As aplicações do algoritmo de geração de *slice* para o elemento requerido selecionado anteriormente, irá manipular o grafo de dependência gerado a partir dos nós e arcos de controle e de fluxo correspondentes ao *slice* desejado, incluindo todas as instruções necessárias para o processamento adequado da variável e desprezando todas as demais instruções.

Já a figura 2 apresenta o *slice* correspondente ao subconjunto do algoritmo resultante do elemento requerido de teste.

Figura 2. Algoritmo resultante do elemento requerido, < 2, 6, d >.

Observa-se que este algoritmo resultante apresenta apenas as linhas de código que foram utilizadas para as dependências do fluxo de dados do elemento requerido de teste selecionado. Observa-se também neste algoritmo, que a ferramenta do *slicing* deve preparar este subconjunto, o *slice*, para que possa aplicar as métricas e determinar a complexidade deste elemento requerido.

Depois de analisar todo o processo de como aplicar as técnicas do *slicing* de programas e verificar a importância de uma ferramenta automatizada, a figura 3 apresenta o modelo conceitual da ferramenta em questão.

Figura 3. Modelo conceitual da ferramenta para *slicing* de programas.

Este modelo destaca o programa sendo dividido em três partes, a primeira a manipulação do grafo de dependência, armazenando os arcos de controle e fluxo, a segunda a passagem de parâmetro relacionado ao nó, armazenando a definição e uso da variável do elemento requerido e a terceira aplicar as técnicas do *slicing* e a extração do *slice*.

Para a realização do *slicing*, dois algoritmos foram elaborados, sendo um para montagem do grafo de dependência e outro para extração do *slice*.

5. Considerações Finais

A ferramenta para *slicing* de programas demonstrada neste trabalho utiliza uma técnica intraprocedimental para a extração de segmentos de programa, tendo como base um elemento requerido no teste estrutural baseado em fluxo de dados, correspondente à implementação de uma associação definição-uso de uma variável do programa. A utilidade desta ferramenta está na facilidade com que permitirá a realização de coleta e cálculo das métricas de complexidade de software, o que seria extremamente difícil de fazer sem o isolamento do código em questão. Esta análise é útil para a avaliação da eficácia de teste e relacionamento da complexidade com o número de defeitos do programa.

A aplicação da técnica de *slicing* manualmente poderá ser algo muito trabalhoso e complicado, tornando-se cada vez mais impraticável para programas com muitas linhas de código. Se considerarmos que, mesmo um programa pequeno, possui vários elementos requeridos, determinar os *slices* correspondentes a cada elemento requerido é inviável de se fazer manualmente para a grande maioria dos programas.

Observou-se também a necessidade de construir um compilador para manipular as técnicas do grafo de dependência e extrair o *slice*. Pesquisadores já alertavam para esta técnica no levantamento teórico. No entanto destaca-se que o compilador tem a capacidade de manipular os algoritmos e gerar os modelos que auxiliem nos casos de teste da Engenharia de Software.

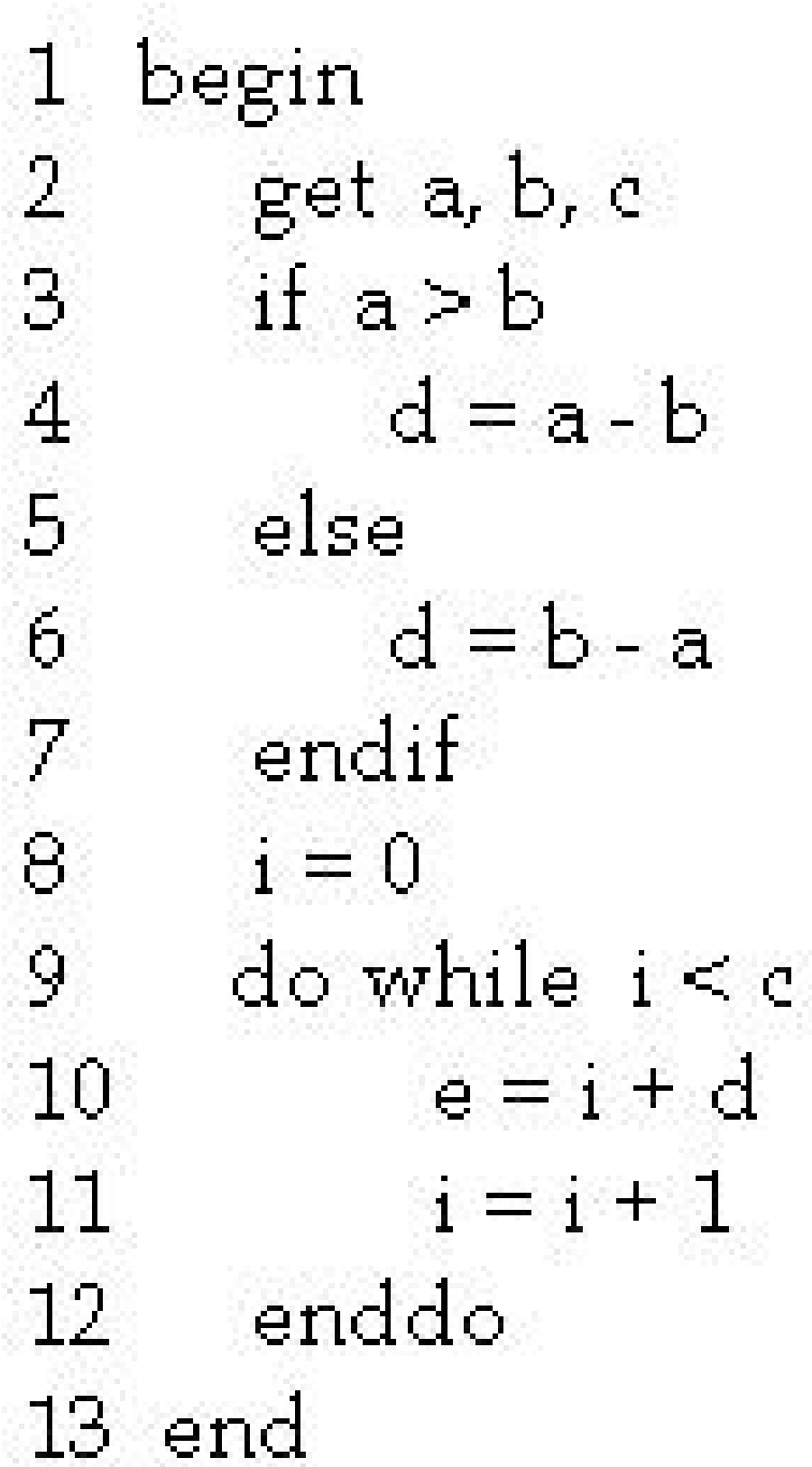
A proposta do projeto de estudar as técnicas do *slicing* de programas foram atingidas, incluindo a criação de algoritmos e suas especificações para gerar o grafo de dependência e extrair o *slice* correspondente. O projeto em questão permite continuidade, em forma de trabalho de conclusão de curso, em fase de início, que permitirá aprimorar a especificação e a implementação da ferramenta, bem como de sua documentação, completando os testes necessários para sua validação.

Referências Bibliográficas

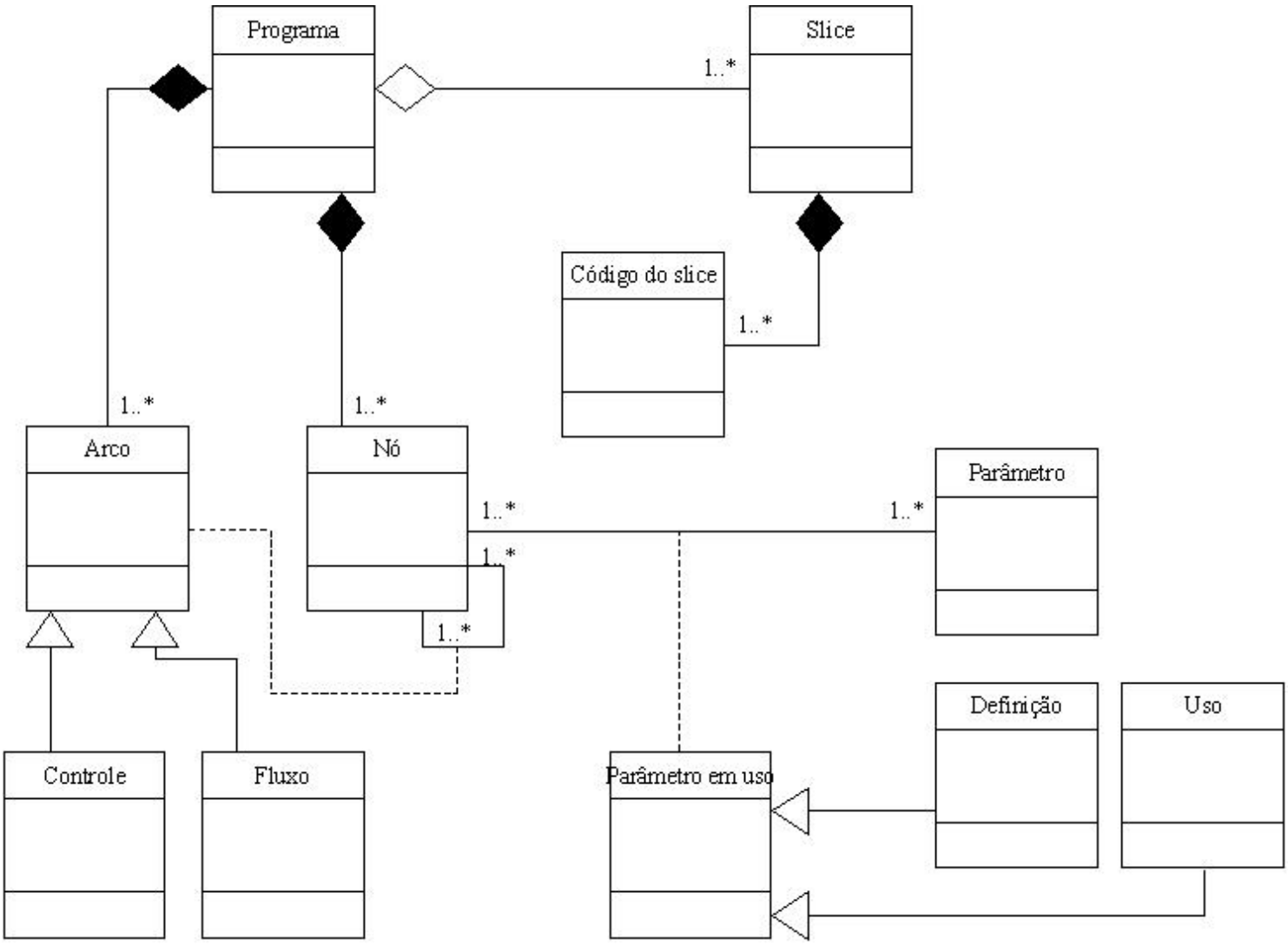
NTAFOS, S., Software Testing: Theory and Pratice. IEEE International Test Conference, 1992.

WEISER, M., Program *Slicing*. IEEE Transactions on Software Engineering SE-10(4) pp 352-357, 1984.

Anexos



```
1  begin
2      get a, b, c
3      if a > b
4          d = a - b
5      else
6          d = b - a
7      endif
8      i = 0
9      f = 0
10     do while i < c
11         e = i + d
12         f = f + e
13         print e
14         i = i + 1
15     enddo
16     print f
17 end
```


Programa
Slice
1..*
Código do slice
1..*
1..*
1..*
Arco
Nó
Parâmetro
1..*
1..*
1..*
1..*
Definição
Uso
Controle
Fluxo
Parâmetro em uso