Especificação Formal

Métodos formais

- A especificação formal é parte de um coleção mais geral de técnicas que são conhecidas como 'métodos formais'.
- São todas baseadas na representação matemática e na análise de software.
- Os métodos formais incluem
  - Especificação formal;
  - Análise e prova de especificação;
  - Desenvolvimento transformacional;
  - Verificação de programa.

Aceitação de métodos formais

- Os métodos formais não se tornaram técnicas principais de desenvolvimento de software conforme foram previstos
  - Outras técnicas de engenharia de software tiveram sucesso no aumento da qualidade de sistema.
  - Mudanças de mercado fizeram do tempo de entrega o fator chave, ao invés do software com uma baixa contagem de erros.
  - O escopo de métodos formais é limitado.
  - Os métodos formais ainda são difíceis de serem extendidos para sistemas de grande porte.

O uso de métodos formais

- Um dos principais benefícios dos métodos formais é a redução do número de defeitos nos sistemas.
- Principal área de aplicação é a engenharia de sistemas críticos.
  - O uso de métodos formais é mais apropriado em termos de custo, porque os altos custos de falha de sistema devem ser evitados.

Uso da especificação formal

- A especificação formal envolve o investimento de maior esforço nas fases iniciais do desenvolvimento de software.
- Isso reduz os erros de requisitos à medida que força uma análise detalhada dos requisitos.
- A não completeza e as inconsistências podem ser descobertas e resolvidas.
  - Economiza-se à medida que a quantidade de retrabalho diminui devido à redução dos problemas de requisitos.

Técnicas de especificação

- Especificação algébrica
  - O sistema é especificado em termos de operações e de seus relacionamentos.
- Especificação baseada em modelos
  - O sistema é especificado em termos de um modelo de estados que é desenvolvido usando construções matemáticas, tais como conjuntos e sequências. As operações são definidas pelas modificações no estado do sistema.

Tabela 10.1 Linguagens formais de especificação

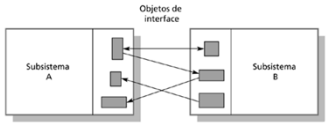
	Seqüencial	Concorrente
Algébrica	Larch (Guttag, et al., 1993); OBJ (Futatsugi, et al., 1985)	Lotos (Bolognesi e Brinksma, 1987)
Baseada em modelos	Z (Spivey, 1992); VDM (Jones, 1980); B (Wordsworth, 1996)	CSP (Hoare, 1985); redes de Petri (Peterson, 1981)

Especificação de interface

- Sistemas de grande porte são decompostos em subsistemas com interfaces bem definidas entre eles.
- A especificação das interfaces dos subsistemas permite o desenvolvimento independente dos diferentes subsistemas.
- As interfaces podem ser definidas como tipos de dados abstratos ou classes de objetos.
- A abordagem algébrica para a especificação formal é, particularmente, bem adequada para a especificação da interface, já que ela se concentra nas operações definidas em um objeto.

Interfaces de subsistemas

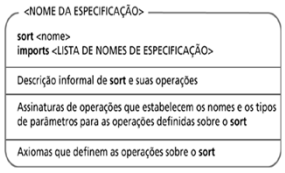
Figura 10.4
Objetos de interface de subsistemas.



A estrutura de uma especificação algébrica

- Descrição por meio de propriedades
  - Não há modelo explícito, nem valores para os tipos de dados

Figura 10.5
Estrutura de uma especificação algébrica.



Componentes da especificação

- Introdução
  - Define o sort (o nome do tipo) e declara outras especificações que são usadas.
- Descrição
  - Descreve informalmente as operações sobre o tipo.
- Assinatura
  - Define a sintaxe das operações na interface e seus parâmetros.
- Axiomas
  - Define a semâmtica de operação pela definição de axiomas que caracterizam o comportamento.

Operações da especificação

- Operações de construtor. São operações que criam entidades do tipo que está sendo especificado.
- Operações de inspeção. Operações que avaliam entidades do tipo que está sedo especificado.

Operações sobre um tipo abstrato de dados lista (ADT)

- Operações do construtor que avaliam a sort List
  - Create, Cons e Tail.
- Operações de inspeção que tomam a sort List como parâmetro e retornam algum outro sort
  - Head e Length.

Especificação algébrica de uma lista

Figura 10.6
Simples especificação de lista.

LIST (Elem)

sort List
imports INTEGER

Define uma lista na qual os elementos são adicionados no fim e removidos do início. As operações são Create, que cria uma lista vazia, Cons, que cria uma nova lista com um membro adicionado, Length, que avalia o tamanho da lista, Head, que avalia o primeiro elemento da lista, e Tail, que cria uma lista ao remover o primeiro elemento de sua lista de entrada. Undefined representa um valor indefinido do tipo Elem.

Create → List
Cons (List, Elem) → List
Head (List) → Elem
Length (List) → Integer
Tail (List) → List

Head (Create) = Undefined **exception** (empty list)
Head (Cons (L, v)) = **if** L = Create **then** v **else** Head (L)
Length (Create) = 0
Length (Cons (L, v)) = Length (L) + 1
Tail (Create) = Create
Tail (Cons (L, v)) = **if** L = Create **then** Create **else** Cons (Tail (L), v)

Recursão nas especificações

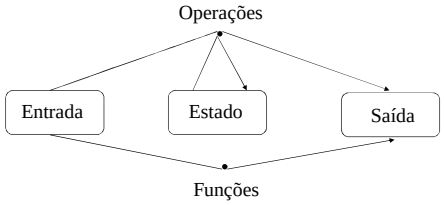
- As operações são frequentemente especificadas recursivamente.
- Tail (Cons (L, v)) = **if** L = Create **then** Create **else** Cons (Tail (L), v).
  - Cons ([5, 7], 9) = [5, 7, 9]
  - Tail ([5, 7, 9]) = Tail (Cons ([5, 7], 9)) = Cons (Tail ([5, 7]), 9) = Cons (Tail (Cons ([5], 7)), 9) = Cons (Cons (Tail ([5]), 7), 9) = Cons (Cons (Tail (Cons ([], 5)), 7), 9) = Cons (Cons ([Create], 7), 9) = Cons ([7], 9) = [7, 9]

Especificação de comportamento

- A especificação algébrica pode ser incômoda quando as operações do objeto não são independentes de seu estado.
- A especificação baseada em modelos expõe o estado do sistema e define as operações em termos de mudanças para esse estado.

Baseados em Modelos (1)

- Objetivo: modelo matemático abstrato
  - Entrada, saída, estado do sistema
  - Funções e operações



Baseados em Modelos (2)

- Estado
  - Componentes de estado
    - Conjuntos, mapeamentos, sequências
  - Invariante
    - Propriedades do estado que devem ser preservadas por todas as operações
- Operações
  - Definidas em termos dos tipos dos objetos
  - Acesso de leitura e escrita ao estado
  - Pode receber argumentos e produzir resultados

Baseados em Modelos (3)

- Funções
  - Não utilizam o conceito de estado
  - Implícita: relação entre argumentos e resultado definada por um predicado
  - Explícita: grafo, i.e., conjuntos de pares (a,b), a pertence ao domínio e b à imagem

Baseados em Modelos (4)

- Exemplo

var

pilha, pilha' : seq[Int]

operações (procedimentos, métodos)

vazia = (pilha' = [])

push(i? : Int) = pilha'=[i?]^pilha

pop() = (pilha # []) => pilha'= tail pilha

top(i! : Int) = (pilha # []) => i! = head pilha

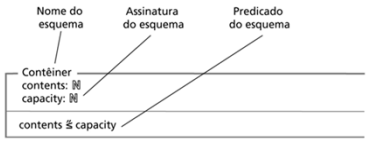
e_vazia (b! : Bool) = b! <=> (pilha = [])

Linguagem Z

- Técnica madura para especificação baseada em modelos. Ela combina a descrição formal e informal e usa destaques gráficos na apresentação das especificações.
- Baseado no uso de conjuntos e lógica de primeira ordem
- Modularidade obtida a partir de Esquemas

A estrutura de um esquema em Z

Figura 10.8
Estrutura de um esquema Z.



Agenda de aniversário em Z

Estado e inicialização

BirthdayBook

$known : \mathbb{P}\, NAME$

$birthday : NAME \pfun DATE$

$known = \mathrm{dom}\, birthday$

InitBirthdayBook

BirthdayBook

$known = \varnothing$

Agenda de aniversário em Z - Operações

AddBirthday

$\Delta BirthdayBook$

$name? : NAME$

$date? : DATE$

$name? \notin known$

$birthday' = birthday \cup \{name? \mapsto date?\}$

DelBirthday

$\Delta BirthdayBook$

$name? : NAME$

$date? : DATE$

$name? \in known$

$birthday' = birthday \setminus \{name? \mapsto date?\}$

Agenda de aniversário em Z - Operações

FindBirthday

$\Xi BirthdayBook$

$name? : NAME$

$date! : DATE$

$name? \in known$

$date! = birthday(name?)$

Remind

$\Xi BirthdayBook$

$today? : DATE$

$cards! : \mathbb{P}\, NAME$

$cards! = \{ n : known \mid birthday(n) = today? \}$

Pontos-chave

- A especificação formal de sistemas complementa as técnicas informais de especificação.
- As especificações formais são precisas e não ambíguas. Elas removem áreas duvidosas em uma especificação.
- A especificação formal força uma análise dos requisitos de sistema nos estágios iniciais. A correção de erros nesse estágio é mais barata que a modificação de um sistema já entregue.
- Técnicas de especificação formal são mais adequadas no desenvolvimento de sistemas críticos e padrões.

Pontos-chave

- Técnicas algébricas são adequadas para especificação de interfaces onde a interface é definida como um conjunto de classes de objeto.
- Técnicas baseadas em modelos modelam o sistema usando conjuntos e funções. Isso simplifica alguns tipos de especificação de comportamento.
- As operações são definidas em uma especificação baseada em modelos pela definição de pré e pós-condições sobre o estado do sistema.