



6. Modellare la specifica

Come descrivere cosa

Andrea Polini

Ingegneria del Software
Corso di Laurea in Informatica

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

 Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli

Fondamenti di Ingegneria del Software, 2^a Ed. Italiana

Prentice Hall, 2004

(Capitolo 5)

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Generalità

Una specifica termine piuttosto generale utilizzato in **differenti fasi della produzione del software**.

In generale una specifica fornisce una precisa definizione delle caratteristiche del “manufatto” che deve essere realizzato nelle fasi successive, fino a giungere all’implementazione.

Specifica dei requisiti contiene dunque una definizione delle caratteristiche e delle funzionalità attese dal cliente.

Il problema della specifica

Specifiche dei requisiti in linguaggio naturale presenta diverse problematiche e difficoltà di verifica.

- linguaggio tecnico
- ambiguità
- sinonimi
- completezza
- inconsistenza
- comprensibilità
- ...

Il problema del **significato**

Pensate alle differenze con specifiche matematiche (e.g. addizione)

Formalismi

Formalismo: notazione con precisa sintassi e semantica

Specifiche formali vengono definite attraverso l'uso di notazioni formali

Vantaggi principali

- comprensione ?
- ambiguità ?
- completezza ?
- inconsistenza ?

Verifica formale: dato un formalismo ed una specifica formale è possibile **verificare esattamente** se alcune proprietà valgono oppure no direttamente **“ragionando” sul modello.**

Specifiche

formali, informali, semiformali

Specifiche formali sono tipicamente costose da definire. In alcuni casi può comunque essere utile poter avere specifiche più semplici e comprensibili ad un più ampio “pubblico”

Si distingue dunque tra:

- Formali
- semiformali
- informali

Specifiche

Operazionali vs. Descrittive

Le specifiche si distinguono in due grandi categorie:

- **operazionali**: descrivono un sistema attraverso il comportamento necessario a raggiungere gli obiettivi
- **descrittive**: descrivono proprietà del sistema che devono essere vere

Un esempio dal mondo matematico.

Un esempio nel mondo “informatico”

Come al solito le distinzioni non sono sempre nette!

Specifiche

Operazionali vs. Descrittive

Le specifiche si distinguono in due grandi categorie:

- **operazionali**: descrivono un sistema attraverso il comportamento necessario a raggiungere gli obiettivi
- **descrittive**: descrivono proprietà del sistema che devono essere vere

Un esempio dal mondo matematico.

Un esempio nel mondo “informatico”

Come al solito le distinzioni non sono sempre nette!

Specifiche

Operazionali vs. Descrittive

Le specifiche si distinguono in due grandi categorie:

- **operazionali**: descrivono un sistema attraverso il comportamento necessario a raggiungere gli obiettivi
- **descrittive**: descrivono proprietà del sistema che devono essere vere

Un esempio dal mondo matematico.

Un esempio nel mondo “informatico”

Come al solito le distinzioni non sono sempre nette!

Specifiche

Operazionali vs. Descrittive

Le specifiche si distinguono in due grandi categorie:

- **operazionali**: descrivono un sistema attraverso il comportamento necessario a raggiungere gli obiettivi
- **descrittive**: descrivono proprietà del sistema che devono essere vere

Un esempio dal mondo matematico.

Un esempio nel mondo “informatico”

Come al solito le distinzioni non sono sempre nette!

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Modelli Operazionali

- Data Flow Diagram
- Finite State Machines
- Petri Nets

Sommario

1 Specifiche - generalità

2 **Modelli Operazionali**

- **Data Flow Diagram (DFD)**
- Finite State Machines (FSM)
- Petri Nets (PN)

3 **Modelli Descrittivi**

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Caratteristiche

DFD sono una notazione molto utilizzata per **descrivere le funzioni di un sistema informativo** e su come i dati fluiscono all'interno del sistema. Il **sistema è in effetti visto come un insieme di funzioni** che manipolano dati.

Come possono essere manipolati i dati:

- I dati possono essere immagazzinati in repository
- I dati possono fluire all'interno del sistema
- I dati possono entrare o uscire dal sistema

Sintassi



Simbolo utilizzato per rappresentare le funzioni



Simbolo utilizzato per rappresentare il flusso del dato



Simbolo utilizzato per rappresentare un repository



Simbolo utilizzato per rappresentare un dispositivo di input



Simbolo utilizzato per rappresentare un dispositivo di output

Regole per combinare i simboli...

Qualche esempio

- dal mondo matematico
- Il distributore del caffè
- La biblioteca

Semantica ed arricchimento delle informazioni in un DFD

Esempio di ambiguità

Formali? Semiformali? Informali?

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- **Finite State Machines (FSM)**
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Caratteristiche

Una macchina a stati finiti è formalmente definita da una tupla $\langle Q, I, \delta, q_0 \rangle$ dove:

- Q è un insieme di stati
- I è un insieme finito di input
- δ è la funzione di transizione ($Q \times I \rightarrow Q$)
- $q_0 \in Q$ è lo stato iniziale

Notazione grafica ...

FSM sono particolarmente utili a modellare il **controllo** di un sistema software.

Qualche esempio

- interruttore
- buffer di lettura scrittura
- la macchinetta del caffè

Questioni

Formali? semiformali? informali?

FSM sono molto usate per verificare formalmente proprietà dei sistemi

Una FSM si trova **sempre in un solo stato** e permette di specificare **soltanto uno stato successivo** per una stessa azione. Allo stesso tempo il sistema è un sistema sincrono che non ha attività concorrenti.

Esplosione degli stati (e.g. somma di due interi) ed estensione del formalismo di base. **Memoria limitata dal numero degli stati.**

Possibili estensioni. I/O FSM. **Transizioni con guardia ed azioni.**

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- **Petri Nets (PN)**

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Reti di Petri

Le reti di Petri sono un formalismo ideato negli anni 60 per modellare sistemi concorrenti. Formalmente sono definite da una quadrupla $\langle P, T, F, W \rangle$ dove:

- P è un insieme finito di “piazze”
- T è un insieme finito di transizioni
- $P \cup T = \emptyset$
- $F \subseteq \{P \times T\} \cup \{T \times P\}$ è il flusso della relazione
- $W: F \rightarrow \mathbb{N} - \{0\}$ è la funzione peso che associa un valore non nullo agli elementi di F .

Stato di un rete di petri. $M: P \rightarrow \mathbb{N}$ (Token)

Reti di Petri

notazione grafica



Evoluzione di una rete di Petri:

- Piazze di input ed output
- Una transizione è abilitata se tutte le piazze di input contengono un numero di token uguale o maggiore dei pesi degli elementi flusso entranti sulla transizione
- Transizione con nessuna piazza di input sono sempre abilitate

Petri Net

Tipicamente le **piazze possono rappresentare una risorsa** e l'accesso alla risorsa può essere modellato dalla presenza di Token.

Politiche di risoluzione dei conflitti non sono intrinseche nel formalismo ma devono invece essere implementate. Ad esempio il formalismo non è *fair*. Esempio....

Una rete di petri si dice in deadlock se non ci sono transizioni abilitate. Esempio...

Esempio

- Buffer di due posizioni
- Semaforo
- filosofi a cena

Problemi e possibili estensioni al modello

Formalismo che si focalizza sul **controllo**. Comunque presenta alcune lacune in particolare per la gestione dei dati.

- Non è possibile modificare il flusso in **dipendenza dei dati**.
- Possibilità di selezionare una transizione tra più transizioni attive. Modello **intrinsecamente non-deterministico**.
- Possibilità di specificare **tempo e deadline**.

Token con associati valori. Token tipati e dunque gestione del contenuto. (**Coloured Petri Nets** - CPN)

Politiche di priorità. $\text{pri:T} \rightarrow \mathbb{N}$.

Reti di Petri temporizzate. Alle transizioni sono associati valori di tempo minimo e massimo

Reti temporizzate con associate funzioni di distribuzione (**Stochastic Petri Nets** - SPN)

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Modelli Descrittivi

Questo tipo di modelli forniscono una formalizzazione del sistema in termini delle proprietà che questo soddisfa. Non viene descritto direttamente il suo comportamento.

- Diagrammi Entità Relazione (ER)
- Specifiche Logiche
- Specifiche Algebriche

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Diagrammi Entità-Relazione

Descrizione concettuale della struttura dei dati e delle loro relazioni. In relazione a DFD:

- Diversamente DFD si focalizzavano sul flusso del dato ma non specificavano la struttura
- Possono essere utilizzati insieme per modellare diversi aspetti

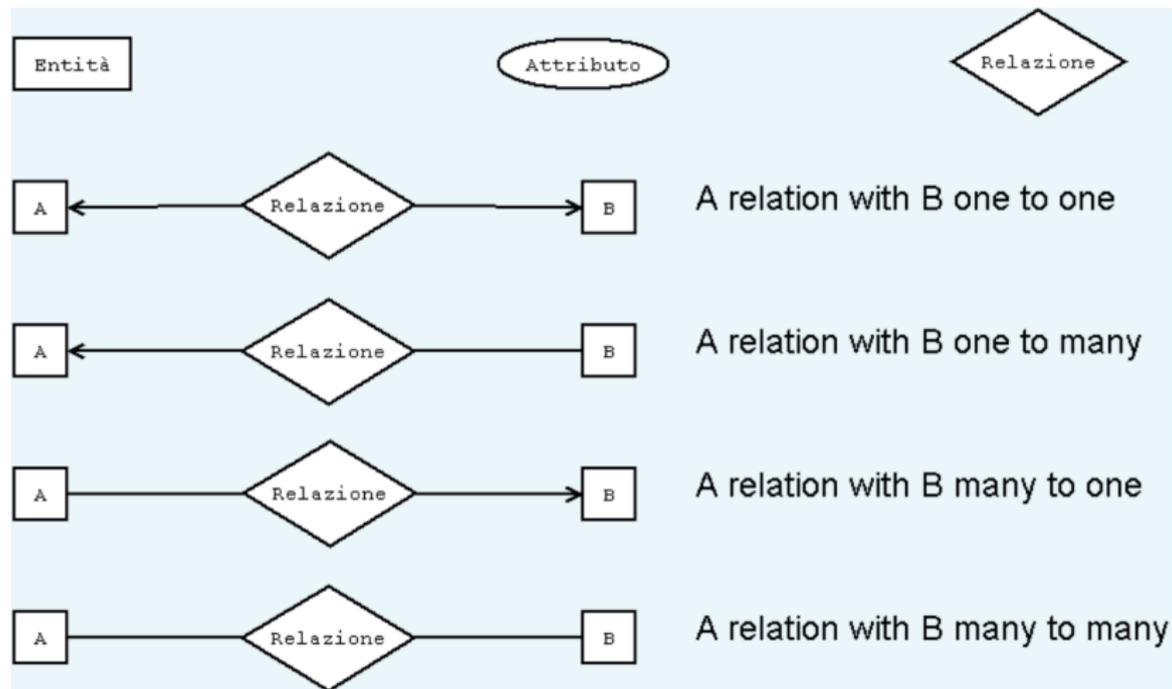
Vengono modellati tre concetti fondamentali:

- Entità
- Attributi
- Relazioni

Diagrammi Entità-Relazione

notazione grafica

ER non sono stati standardizzati, dunque molte varianti



Limitazioni

Potere espressivo è piuttosto limitato. Concetti complessi di relazione non possono essere rappresentati. (e.g. Ereditarietà).

Non è possibile specificare che un'entità esiste solo se il numero di istanze in una relazione sarà maggiore di un certo numero.

Come al solito poi per i problemi più "spinosi" sono proposte estensioni. In generale il diagramma viene complementato con informazioni aggiuntive. Formali e non.

Diagramma delle classi ed ER

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- **Specifiche logiche**
- Specifiche algebriche

Specifiche logiche

generalità

Utilizzo di formule FOT per descrivere **proprietà** di un programma

Una proprietà sarà soddisfatta se il **corrispondente predicato** assumerà **valore positivo**.

FOT è un'espressione che può comprendere:

- variabili (a, b, \dots)
- costanti numeriche ($1, 3.4, 4.233e10, \dots$)
- funzioni $f(x, y, \dots), \dots$
- predicati $p(), \dots$
- parentesi $(), [], \{ \}, \dots$
- connettivi logici ($\wedge, \vee, \neg, \Rightarrow, \dots$)
- quantificatori (\forall, \exists)

Specifiche logiche

- Variabili **libere e legate** e dipendenza del valore di verità
- Valore di verità dipende dal **dominio scelto** per le variabili della formula

Dato un programma P con input $\langle i_1, i_2, \dots, i_n \rangle$ ed output $\langle o_1, o_2, \dots, o_m \rangle$ una specifica logica per P è definita come:

$\{\mathbf{Pre}(i_1, i_2, \dots, i_n)\}$

P

$\{\mathbf{Post}(o_1, o_2, \dots, o_m, i_1, i_2, \dots, i_n)\}$

Esempi...

Esercizio:

1. programma che genera permutazione
2. programma che verifica che il primo elemento di un vettore sia duplicato

Specifiche di parti

Specifiche logiche possono essere utilizzate anche per specificare parti di programma ... richiede di poter utilizzare le variabili di programma all'interno della specifica logica

Esempi...

Il caso di una classe nei linguaggi OO e le invarianti.

- Stato di un oggetto
- Esecuzione dei metodi e pre e post condizioni
 - $\{INV \wedge Pre(i_1, i_2, \dots, i_n)\}$
 $m(i_1, i_2, \dots, i_n)$
 $\{INV \wedge Post(o_1, o_2, \dots, o_m, i_1, i_2, \dots, i_n)\}$

La classe Queue

discussione ...

```
public class Queue {
    Object[] queue;
    int length, head, tail, elementInQueue;
    public Queue(int length) {
        queue = new Object[length];
        this.length=length; this.head=0; this.tail=0; this.elementInQueue=0;
    }
    public void insert(Object o) {
        queue[head]=o; head++; elementInQueue++;
        if (head==length) head=0;
    }
    public Object remove() {
        Object o = queue[tail]; tail++; elementInQueue--;
        if (tail==length) tail=0;
        return o;
    }
    public boolean isEmpty() {
        return (head==tail);
    }
}
```

La classe Queue - 2a versione

discussione ...

```
public class Queue {
    Object[] queue;
    int length, head, tail;
    public Queue(int length) {
        queue = new Object[length];
        this.length=length;this.head=0;this.tail=0;
    }
    public void insert(Object o) {
        queue[head]=o;head++;
        if (head==length) head=0;
    }
    public Object remove() {
        Object o = queue[tail];tail++;
        if (tail==length) tail=0;
        return o;
    }
    public boolean isEmpty() {
        return (head==tail);
    }
}
```

Design by Contract (DbC)

- Principio ingegneristico secondo il quale il **programmatore aggiunge pre-, post-condizioni ed invarianti** alla specifica dei metodi di una classe
- La specifica costituisce il **contratto** con l'utente della classe
- possibilità di verifica a run-time per verificare eventuali violazioni
- supporto da parte di linguaggi e framework specifici (Eiffel, JContract)

Specifiche logiche

considerazioni

Anche semplici programmi generano specifiche piuttosto complesse.

Definizione e riuso di formule

Spesso associate ad altri formalismi per aumentarne la potenza espressiva. E.g. FSM e formalismi logici il caso delle code.

Specifiche logiche e verifica

Utilizzando FOL viene specificato che **per qualsiasi implementazione** di un programma o di un frammento di programma i **predicati devono essere valutati a vero**

Espressioni logiche possono essere utilizzate nella fase di analisi applicando **deduzione logica**. La proprietà espressa in forma FOL deve poter essere **derivata dalla specifica**. Questo fornisce una prova! (e.g. Il caso della coda).

Questo non era possibile per le specifiche operazionali con le quali è possibile **scoprire comportamento (simulazioni)** ma non fornire prove. Ma ... **theorem proving è indecidibile in FOT**

Specifiche logiche e verifica

Utilizzando FOL viene specificato che **per qualsiasi implementazione** di un programma o di un frammento di programma i **predicati devono essere valutati a vero**

Espressioni logiche possono essere utilizzate nella fase di analisi applicando **deduzione logica**. La proprietà espressa in forma FOL deve poter essere **derivata dalla specifica**. Questo fornisce una prova! (e.g. Il caso della coda).

Questo non era possibile per le specifiche operazionali con le quali è possibile **scoprire comportamento (simulazioni)** ma non fornire prove. Ma ... **theorem proving è indecidibile in FOT**

Sommario

1 Specifiche - generalità

2 Modelli Operazionali

- Data Flow Diagram (DFD)
- Finite State Machines (FSM)
- Petri Nets (PN)

3 Modelli Descrittivi

- Diagrammi Entità-Relazione
- Specifiche logiche
- Specifiche algebriche

Specifiche algebriche

generalità

Formalismo descrittivo basato su definizione ed uso di algebre

- algebre omogenee
- algebre eterogenee

Molti sistemi software possono essere specificati “facilmente” utilizzando algebre eterogenee

Specifiche algebriche

esempio

Si vuole specificare un sistema che gestisce le stringhe. Con le seguenti operazioni:

- creazione di una stringa vuota
- concatenazione di stringhe
- aggiunta di un carattere ad una stringa
- lunghezza di una stringa
- verifica se una stringa è vuota o meno
- verifica di uguaglianza di stringhe

Insiemi?

Specifiche algebriche

esempio

Definizione di algebra (sorts, segnatura, operazioni, vincoli)

La semantica delle operazioni viene data tramite equazioni che definiscono le proprietà fondamentali che devono valere quando le operazioni sono applicate (**assiomi**)

Specifiche algebriche

vincoli

constraints isEmpty, append, new, add, equal, length **so that**
for all [s,s1,s2:String;c:Char]

- isEmpty(new()) = ?
- isEmpty(add(s,c)) = ?
- length(new()) = ?
- length(add(s,c)) = ?
- append(s,new()) = ?
- append(s1,add(s2,c)) = ?
- equal(new(),new()) = ?
- equal(new(),add(s,c)) = ?
- equal(add(s,c),new()) = ?
- equal(add(s1,c),add(s2,c)) = ?

Specifiche algebriche

derivazione di proprietà

Una proprietà è vera se può essere derivata dagli assiomi:

$$\text{append}(\text{new}(), \text{add}(\text{new}(), c)) = \text{add}(\text{new}(), c)$$
$$\text{append}(\text{new}(), s) = s$$

Specifiche algebriche

proprietà non derivabili dagli assiomi

Capita spesso di poter incontrare proprietà che non possono essere derivate degli assiomi. Si consideri ad esempio la proprietà:

$\text{equal}(\text{add}(s,'a'),\text{add}(s,'b')) = \text{false}$

La specifica è incompleta e possono essere attuati tentativi di rendere la specifica “più” completa aggiungendo altri assiomi.

Specifiche algebriche

specifiche modulari

Sistemi complessi possono richiedere di definire molte algebre allo stesso tempo.

Sono previsti operatori di “import” che permettono di includere un'algebra nella definizione di un'altra

Esercizio

Confronto tra i diversi tipi di specifica e dei meccanismi di analisi applicabili:

Obiettivo: Salvare capra e cavoli :-)