Verifica – parte IV

Rif. Ghezzi et al.
6.8-6.9

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Debugging

Individuazione e correzione degli errori

Conseguente a un fallimento

Attività non banale:

Quale errore ha causato il fallimento?

Come correggere l'errore?

La correzione dell'errore ha effetti collaterali (cioè genera altri errori)? (test di regressione)

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Individuazione dell'errore

Ridurre la distanza fra errore e fallimento.

Strategia: rendere visibile lo stato del programma:

esecuzione controllata (debugger)

asserzioni

istruzioni di output

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Debugger

Aiuta nell'individuazione degli errori senza modifiche al codice sorgente

Funzionalità di base:

esecuzione step by step

breakpoint

valutatore di espressioni

watch (break se il valore di una variabile cambia)

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Affidabilità

- Definita come la probabilità che un software funzioni correttamente in un determinato intervallo di tempo
- Negli ultimi anni sono stati concepiti modelli che consentono di stimare e di prevedere l'affidabilità di un sistema software.
- Mutuati da altri ambiti (hardware o ingegneria industriale)

Peculiarità del software

- Il software non è soggetto a corruzione o usura fisica
- Il software non presenta errori transienti e non necessita di rodaggio
- Gli esemplari di un software sono uguali fra loro
- Mancanza di continuità

Quantità legate all'affidabilità

- AF(t), numero medio di fallimenti dopo un tempo t.
  - Misura media su diverse installazioni del sistema
  - Estensione derivabile della funzione
- FI(t), intensità di fallimento
  - Fallimenti per unità di tempo
  - Derivata di AF rispetto a t
- MTTF, mean time to failure
  - Tempo medio fra due fallimenti
  - Reciproco di FI

Tempo

- Calendar time: tempo totale trascorso dall'installazione
  - comprende il tempo in cui il sistema è fermo
- Tempo di esecuzione: tempo di funzionamento del software
  - comprende il tempo in cui la piattaforma è allocata per altri processi
- Tempo di clock: tempo effettivo di esecuzione del software sulla piattaforma.

Modelli di affidabilità

- Ipotizzano una relazione fra l'intensità di errore e il numero medio di fallimenti.
- Con l'ipotesi fatta, si utilizzano strumenti matematici per stimare l'andamento del numero medio di fallimenti nel tempo.
- Il punto critico è la scelta della relazione, basata su considerazioni sulla natura del processo e del prodotto software.

Modello base

- Ipotizza che l'intensità di fallimento decresca di una costante per ogni fallimento

$$FI(AF) = k(1 - AF/AF_\infty)$$

- dove AF_∞ è il numero totale di fallimenti (ignoto) e k è una costante
- Data questa ipotesi, si può determinare AF(t)

Calcolo di AF(t)

$$FI(t) = AF'(t)$$

- Equazione differenziale

$$\frac{dAF}{dt} = k(1 - AF/AF_\infty)$$

$$\frac{dAF}{(1 - AF/AF_\infty)} = kdt$$

$$-AF_\infty \frac{dAF}{(AF - AF_\infty)} = kdt$$

Calcolo di AF(t)

- Integrando entrambi i membri

$$-AF_\infty \int \frac{dy}{(y - AF_\infty)} = k \int du$$

$$-AF_\infty [\log(y - AF_\infty)]_{AF_0}^{AF} = k(t - t_0)$$

- Supponendo $AF_0 = 0$ $t_0 = 0$
- si ottiene

$$\log(AF - AF_\infty) + \log(AF_\infty) = -\frac{k}{AF_\infty} t$$

Calcolo di AF(t)

- Essendo $0 \le AF \le AF_\infty$

$$\log(AF_\infty - AF) - \log AF_\infty = -\frac{k}{AF_\infty}t$$

$$\log\frac{AF_\infty - AF}{AF_\infty} = -\frac{k}{AF_\infty}t$$

- Esponenziale di entrambi i membri:

$$\frac{AF_\infty - AF}{AF_\infty} = e^{-\frac{k}{AF_\infty}t} \qquad AF = AF_\infty\left(1 - e^{-\frac{k}{AF_\infty}t}\right)$$

Calcolo di k

- FI è la derivata di AF rispetto a t:

$$FI = -AF_\infty\left(-\frac{k}{AF_\infty}\right)e^{-\frac{k}{AF_\infty}t}$$

- Sostituendo 0 a t:

$$k = FI_0$$

Modello logaritmico

- Suppone che il decremento di FI per ogni fallimento decresca esponenzialmente:

$$FI = ke^{-\theta AF}$$

- dove θ è un parametro detto decadimento dell'intensità di fallimento

Calcolo di AF(t)

- Equazione differenziale:

$$\frac{dAF}{dt} = ke^{-\theta AF}$$

- Separazione delle variabili:

$$\frac{dAF}{e^{-\theta AF}} = kdt$$

- Integrazione di entrambi i membri:

$$\int \frac{dy}{e^{-\theta y}} = k\int du$$

Calcolo di AF(t)

- Risultato dell'integrazione:

$$\frac{1}{\theta}\left[e^{\theta y}\right]_0^{AF} = \left[ku\right]_0^t$$

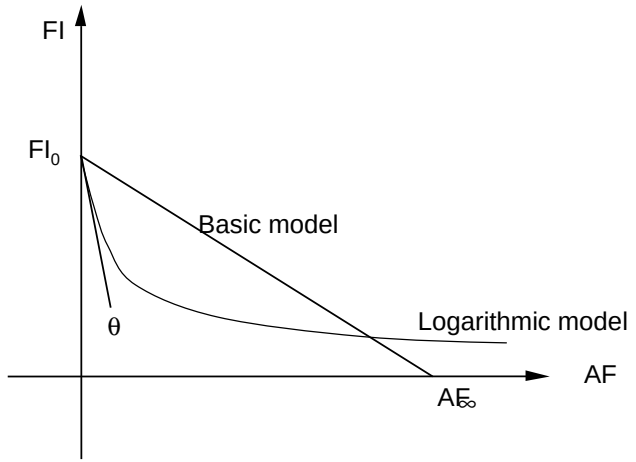
- Cioè

$$e^{\theta AF} - 1 = \theta kt \qquad AF = \frac{1}{\theta}\log\left(1 + \theta kt\right)$$

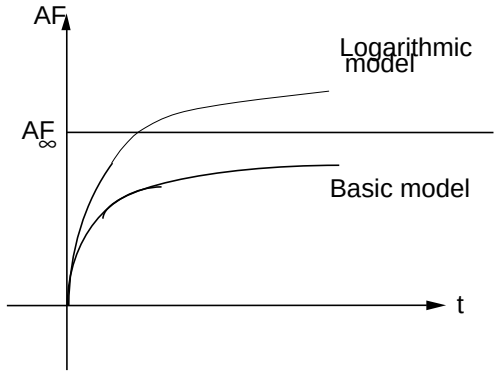
- Calcolo di k

$$FI = \frac{1}{\theta}\frac{1}{1+\theta kt}\theta k \qquad k = FI(0)$$

Confronto fra modelli: FI



Confronto fra modelli: AF



Calcolo dei parametri

- AF_∞ e θ non sono noti a priori
- Possono essere stimati tramite osservazione di AF(t)
- Il risultato permette di stimare l'andamento dell'affidabilità nel tempo
- Modelli diversi per classi diverse di applicazioni.

Verifica di qualità soggettive

- Qualità del software come
  - complessità
  - comprensibilità
  - riusabilità

  sono largamente soggettive
- C'è comunque richiesta di metriche per la misurazione di queste qualità
- Le proposte hanno generalmente applicabilità limitata ad alcune classi di applicazioni.

Software science (Halstead)

- Cerca di dare misure e stime quantitative di qualità soggettive, quali
  - difficoltà
  - livello di astrazione
  - sforzo
- Le misure sono date in termini di quantità oggettive

Quantità

- η_1: numero di operatori unici e distinti
- η_2: numero di operandi unici e distinti
- N_1: numero totale di occorrenze di operatori
- N_2: numero totale di occorrenze di operandi
- η_2^*: numero di operandi concettuali di ingresso/uscita distinti

Esempio

```
begin
      max := 0;
      read(x);
      while x<>0 do
           begin
                if x > max
                then max := x;
                read(x)
           end;
      write(max)
end;
```

Operatori		Operandi	
Begin ... end	2	max	4
:=	2	0	2
;	5	x	5
Read	2		
(...)	3		
While...do	1		
<>	1		
if...then	1		
>	1		
Write	1		

- $\eta_1 = 10$
- $\eta_2 = 3$
- $N_1 = 19$
- $N_2 = 11$
- $\eta_2^* = 3$ (2 ingressi, 1 uscita)

Lunghezza del programma

- Vocabolario del programma
  $\eta = \eta_1 + \eta_2$
- Lunghezza del programma
  $N = N_1 + N_2$
- Stima di N: $N^* = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$
- Calcoli su algoritmi pubblicati indicano un valore dell'errore medio (N* - N) / N inferiore al 10% (nell'esempio, N=30, N*=38).
- Utile per stime se dall'inizio si possono stimare η_1 ed η_2, ad esempio in base a statistiche su applicazioni simili

Stima di N

- Volume di programma: numero di bit necessario a codificare ogni elemento di programma
  $V = N * \log_2 \eta$
- Volume potenziale: quello del programma più sintetico in cui si può codificare l'algoritmo (disponibile come operazione predefinita):
  $N = \eta = 2 + \eta_2^*$
  $V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$

Livello di programma e sforzo

- $L = V^* / V$
  - Tenta di misurare il livello di astrazione di un programma
- $E = V / L = V^2 / V^*$
  - Misura la difficoltà dell'implementazione, manutenzione, comprensione del programma.
  - Risultati sperimentali soddisfacenti: mostrano la dipendenza della complessità dal linguaggio.

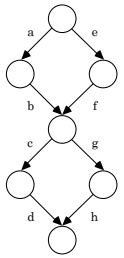
Teoria di McCabe

- Stima la complessità di un programma (per quanto riguarda produzione, comprensione, modifica).
- Basata sulla teoria dei grafi
- Complessità concettuale di un programma (per codifica, correzione, manutenzione) legata alla complessità del suo flusso di controllo

Numero ciclomatico

- Rappresentazione vettoriale dei cammini in un grafo a n archi.
- Ogni cammino è un vettore di n componenti, ognuno uguale al numero di volte che l'arco corrispondente è percorso.
- Numero ciclomatico: numero di cammini linearmente indipendenti.

Esempio



	a	b	c	d	e	f	g	h
P	1	1	1	1	0	0	0	0
Q	1	1	0	0	0	0	1	1
R	0	0	1	1	1	1	0	0
S	0	0	0	0	1	1	1	1

Risultati teorici

- C = e – n + 2p, dove
  - e è il numero di archi
  - n è il numero di nodi
  - p è il numero di componenti connesse del grafo (normalmente una per ogni procedura)
- C = d + 1, dove d è il numero di punti di decisione (a 2 uscite) del programma
  - Un punto di decisione a k uscite è contato come k-1 punti di decisione a 2 uscite (traduzione da costrutto case a costrutto if)

Numero ciclomatico e complessità

- Il numero ciclomatico dà un'idea immediata della complessità del flusso di controllo di un programma.
- Non tiene però conto di altri aspetti, come la complessità delle strutture di dati.
- Sperimentalmente risulta correlato al numero di errori riscontrati.
- Un modulo di un sistema ben progettato dovrebbe avere C fra 3 e 7, e non superare 10 (conferme empiriche).

Approccio Goal Question Metric

- Cerca di integrare e superare le metriche proposte da Halstead, McCabe e altri.
- Presupposti:
  - Qualsiasi metrica deve essere usata per analizzare la qualità, non per valutare le persone
    - In caso contrario, i progettisti cercheranno non di produrre buon software, ma di massimizzare le metriche (es. linee di codice)

Approccio Goal Question Metric

- La valutazione di qualità deve riguardare non solo il prodotto, ma anche il processo.
- Le misure di qualità vanno definite non solo per il prodotto finale, ma anche per tutti i prodotti intermedi.
- Le grandezze da misurare vanno scelte in base agli obiettivi da raggiungere. Le misure e le linee guida sui valori ottimali vanno convalidati in base alle esperienze concrete.

Fase 1: definizione dell'obiettivo (goal)

- Schema generale:
  - analizzare <ambito dello studio>
  - con l'obiettivo di <obiettivo>
  - le <caratteristiche>
  - dal punto di vista di <stakeholder>
  - nel contesto di <il contesto operativo>
- Ambito dello studio: processo, prodotto, fase del processo

Fase 1: definizione dell'obiettivo (goal)

- Obiettivo: caratterizzare, valutare, prevedere, migliorare
- Caratteristiche: costo, correttezza, affidabilità, usabilità
- Stakeholder: utente, progettista, manager, impresa
- Contesto operativo: organizzazione, impresa, gruppo

Fase 1: esempi

- Analizzare il sistema informativo allo scopo di stimare il costo dal punto di vista del manager nel contesto di una software house
- Analizzare la fase di test allo scopo di migliorare l'affidabilità dal punto di vista dell'utente finale nell'ambito di un impianto manifatturiero
- Analizzare la specifica dei requisiti allo scopo di valutare la comprensibilità dal punto di vista del programmatore nel contesto di una software house

Fase 2: definizione delle domande (question)

- Individuare domande la cui risposta ha l'obiettivo di definire il goal
- Esempi:
  - "Esistono malfunzionamenti critici nell'uso del sistema?" oppure "Qual è il MTTF?" (affidabilità)
  - "Qual è lo sforzo necessario per la fase di test?" (costo)

Fase 3: definizione delle metriche (metric)

- Metrica appropriata per ogni domanda:
- Esempi:
  - Criticità di un malfunzionamento: valore da 1 a 10 attribuito dall'utente ("pesato" in base alla competenza?)
  - Copertura del codice sorgente in fase di test: percentuale (righe di codice, istruzioni, decisioni)

GQM nella pratica

- Metodologia ampia, in evoluzione
- Valutazioni informali (qualità soggettive) ma sforzo per arrivare a misure oggettive
- Le decisioni (soprattutto domande e misure) vanno convalidate a posteriori
- Le metriche sono arbitrarie, ma devono rispettare alcune condizioni per essere significative

Condizioni su metriche: esempi

- La metrica di complessità C(P) di un frammento di codice deve essere tale che, se P;Q è la composizione di due frammenti P e Q sia $C(P;Q) \geq C(P)$
- Se D(P) è la misura della dimensione di un programma,
  - $D(P) \geq 0$
  - D(P) = 0 se e solo se P è il programma nullo
  - Se P e Q sono disgiunti (cioè non condividono alcun elemento) D(P;Q) = D(P) + D(Q)

Linee guida

- La definizione di metriche dovrebbe tenere conto delle caratteristiche che influenzano il goal desiderato.
  - Esempio: riusabilità → modularità → alta coesione, basso accoppiamento → rapporto fra metodi pubblici e privati
- Per ogni metrica, dovrebbero essere definite soglie il cui superamento richiede un esame del prodotto o del processo.