Le metriche del software

Università di Ferrara
Corso di Ingegneria del Software
AA 2013/2014

Argomenti trattati

- La misura come strumento scientifico
- Misure, metriche e indicatori
- Metriche del *software*
- Metriche per *software orientato agli oggetti*
- Approfondimenti: accuratezza delle metriche

Misurare: significato

- Un'attività quotidiana
- Grandezze fisiche: massa, lunghezza, velocità …
- Grandezze economiche: prezzi, inflazione, PIL, Mib30, …
- Di tutto di più: gradimento, reddito, caratteristiche fisiche, …
- Misurare per conoscere
- Misurare per giudicare
- L'effetto di semplificazione ed imprecisione

Scienza e misura

- Finalità della misura
- Rendere oggettivi i risultati degli esperimenti
- Oggettività implica ripetibilità, confronto, confidenza
- Avere dati da cui ricavare modelli matematici
- Limiti intrinseci della misura
- È una approssimazione < misure di realtà fisiche
- È una astrazione < misure per valutare o stimare

Misurazione

- Processo che assegna numeri o simboli ad attributi di entità del mondo reale, per descriverle secondo regole definite
- Concetti rilevanti
  - **Entità**: persona, semaforo, programma
  - **Attributo**: età, stato, dimensione
  - **Valore**: 18 anni, rosso, 35 linee

Differenze tra Misura e Metrica

- **Misura**
  - Risultato della misurazione
  - Assegnazione empirica ed oggettiva di un valore (numerico o simbolo) ad un'entità, per caratterizzarne un attributo specifico
- **Metrica, un insieme di regole**
  - Per stabilire le entità da misurare
  - Per definire gli attributi rilevanti
  - Per definire l'unità di misura
  - Per definire una procedura per assegnare numeri e simboli

Indicatori

- Caratteristiche difficilmente misurabili
- Metriche usate per stimare caratteristiche
  - Identificare attributi misurabili
  - Per stimare (con incertezza) caratteristiche non misurabili
- Esempio
  - La dimensione del *software è un attributo misurabile*
  - La manutenibilità è stimata in base alla dimensione (ma non solo!)

Le metriche nella produzione

- Strumenti di valutazione e controllo
- Entità da misurare
  - ***Processi***: *insiemi correlati di procedure astratte*
  - ***Progetti***: *attività concrete legate a tempo e risorse*
  - ***Prodotti***: *beni e servizi in uscita dai progetti*
  - ***Risorse***: *elementi impiegati (e consumati) dal progetto per produrre prodotti*
- Categorie di attributi
- *Attributi **interni** > misurabili rispetto alle entità*
- *Attributi **esterni** > misurabili rispetto all'ambiente*

Le metriche nel software

- Un problema aperto
  - Il *software è difficile da misurare*
  - Il *software presenta aspetti diversi*
  - Le tecnologie cambiano in fretta
  - L'ambiente ha una grande influenza
- Tipi di metriche
  - Del prodotto in sé > quanto è grande
  - Delle sue funzionalità > cosa deve fare
  - Del suo comportamento > cosa e quando succede

Linee di codice sorgente (S)LOC

- La metrica più intuitiva e più usata
- Conteggio dei costrutti
  - Semplificato ed adattato alle funzionalità degli *editor*
  - Usata per derivare informazioni di costo e produttività
- Limiti
  - Dipendente dal linguaggio
  - Dipendente dallo stile di codifica

Analisi di Halstead – 1

- Maurice H. Halstead. *Elements of Software Science,* Elsevier, Amsterdam, 1977

Complessità del codice
- Misura a posteriori o stima durante il progetto
- Controversa
- Basata sul numero e sul tipo dei costrutti
- Basata su elementi di psicologia cognitiva e statistica

- Limiti
  - Adatta solo ai primi linguaggi di programmazione
  - Oggi è statisticamente inefficace

Analisi di Halstead – 2

- Unità di misura
  - n1: numero di operatori distinti usati dal programma
  - n2: numero di operandi distinti usati dal programma
  - N1: numero di occorrenze degli operatori
  - N2: numero di occorrenze degli operandi

- Usate per determinare (tra l'altro)
  - La dimensione (lunghezza) N di un programma
  - N = n1 log n1 + n2 log n2 dove log è in base 2

  - Il volume V di un programma
  - V = N log (n1 + n2) per cui V varia con il variare del
  - linguaggio di programmazione

Halstead assegna un valore (livello) I come costante di linguaggio, ma la successiva ricerca ha mostrato che essa dipende anche dal programmatore

Complessità ciclomatica – 1

- **Proposta da Thomas McCabe nel 1976**
  - Indipendente dal linguaggio di programmazione
- **Complessità del flusso di controllo**
  - Funzione dei possibili cammini indipendenti sul grafo di flusso
  - Rappresentazione astratta del codice
- **Limiti**
  - Fallibilità dimostrata (ciononostante usata)
  - Costosa da applicare prima di scrivere il codice

Complessità ciclomatica – 2

- **Il valore determinato viene confrontato con soglie prefissate**
  - Ad esempio:
    - 1–10 Complessità bassa
    - 21–50 Complessità elevata
    - >50 Complessità non accettabile
- **Corrisponde all'esatto numero di casi di prova necessari per verificare ogni possibile esito di ogni decisione dell'unità**
- **È preferibile calcolarla con l'ausilio di strumenti automatici**

Complessità ciclomatica – 3

- **Definizione algebrica**

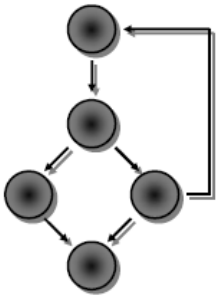
v(G) = e – n + p	numero di percorsi lineari in G
e	numero degli archi (flusso)
n	numero dei nodi (espressioni o comandi)
p	numero delle componenti connesse da ogni arco (=2)

- **Esempio, una sequenza, v(G) = 3 – 4 + 2 = 1**

Complessità ciclomatica – 4

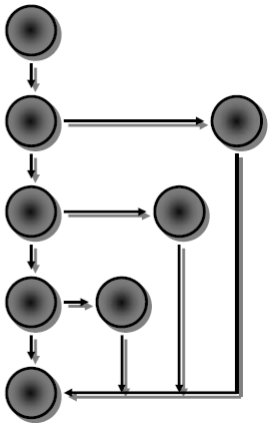
- **Errore in difetto**
  - La complessità reale è maggiore di quella misurata
- **Esempio, v(G) = 6 – 5 + 2 = 3**
  - Ma codice sorgente offuscato!

```
int _INT = 1 ; int  INT_ =
0;  int  Int_ ( int _int )
{ if (_int == INT_) return
_INT ; else   return  _int
* Int_ ( _int - _INT ) ; }
```

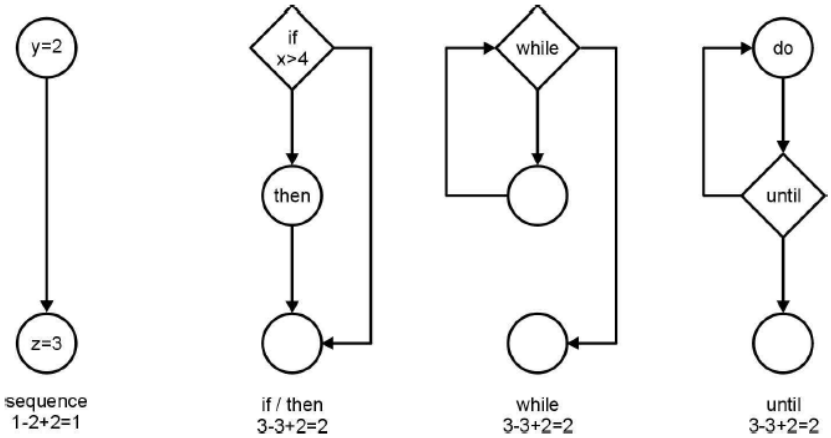


Complessità ciclomatica – 5

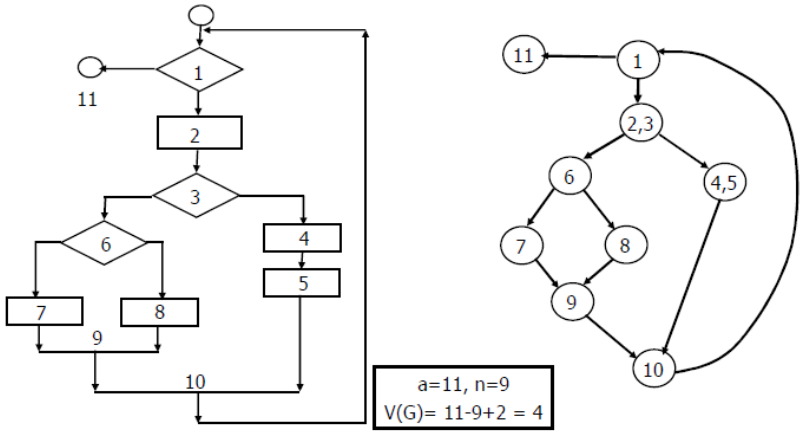
- **Errore in eccesso**
  - La complessità reale è <u>inferiore</u> a quella misurata
- **Esempio, v(G) = 10 – 2 + 2 = 4**
  - `switch` visto come una sequenza di `if` annidati
  - v(G) = C + 1
  - C numero dei casi (= 3)



Complessità Ciclomatica – ES1



Complessità ciclomatica – ES2



Function Point (FP) – 1

- Una proposta di A. J. Albrecht, 1977
  - Non misura il *software, ma le sue entità logico-funzionali*
  - Misura di dimensione del progetto e di produttività
  - Indipendente dal linguaggio di programmazione
  - *International Function Point User Group (IFPUG)*
- Conteggio dei punti funzione
  - Funzionalità differenziate per categorie
  - Funzionalità pesate in base agli attributi del prodotto
- Limiti
  - Misurazioni diverse possono dare risultati leggermente diversi
  - Risultati ottimali per sistemi gestionali

Function Point – 2

Componenti misurate	Definizione
External Input (EI)	Processo elementare a seguito del quale dati entrano nella componente
External Output (EO)	Processo elementare a seguito del quale dati escono dalla componente
External Inquiry (EQ)	Processo elementare per il quale specifici dati interni sono richiesti alla componente e da essa emessi
Internal Logical File (ILF)	Gruppo di dati correlati, interni alla componente, alimentati tramite EI
External Interface File (EIF)	Gruppo di dati correlati, esterni alla componente, usati come riferimento

Function point – 3

- ***Unadjusted* FP (valore grezzo)**
  - Ciascuna componente viene associata ad uno specifico grado di complessità
  - Ogni componente pesa quanto il suo grado di complessità
- ***Adjusted* FP (valore raffinato)**
  - 14 fattori d'influenza (IF), ciascuno con peso da 0.00 a 0.05
  - AFP = UFP × (0.65 + ∑ IF)

Componente	Complessità			Totale
	Bassa	Media	Alta	
EI	× 3	× 4	× 6	
EO	× 4	× 5	× 7	
EQ	× 3	× 4	× 6	
ILF	× 7	× 10	× 15	
EIF	× 5	× 7	× 10	

Evoluzioni del Function Point –1

Feature Points

- I feature points sono un adattamento dei function points, a cui viene aggiunta un altro parametro di misurazione: gli algoritmi!
- Un Algoritmo e definito come un problema computazionale delimitato, incluso per un determinato programma per computer. L'inversione di una matrice, la decifrazione di una sequenza di bit ed il trattamento di un interruzione sono tutti esempi di algoritmi.

Evoluzioni del Function Point – 2

Object Points

Gli object points sono una metrica basata sulle funzioni, ma, a differenza dei function points, essi si incentrano sui tool di ausilio che permettono di creare velocemente schermate e report. I parametri di configurazione che prende in esame sono i seguenti:

- **Schermate**: Tutto cio che deve essere visualizzato sullo schermo
- **Report**: I vari report che devono essere generati
- **Componenti 3GL**: Tutto cio che deve essere costruito "a mano"

CO.CO.MO – Constructive Cost Model

Metrica definita da B. Boehm con l'obiettivo di descrivere il *costo di produzione e* definisce tre modelli:

- *Basic COCOMO:* questo modello calcola lo sforzo e il costo per lo sviluppo di un software come funzione della dimensione del programma espressa in linee di codice stimate
- *Intermediate COCOMO:* oltre alla dimensione del programma si introducono nella funzione alcuni indicatori che includono valutazioni del prodotto, dell'hardware disponibile, del personale e degli attributi del progetto
- *Advanced COCOMO:* incorpora le caratteristiche della precedente descrizione introducendo anche gli indicatori per ogni fase del processo di sviluppo (analisi, design, schematizzazione, ...)

Standard ISO per misure funzionali

- ISO/IEC 14143 – *Functional size measurement*
  - Definire la terminologia del settore
  - Definire i criteri per valutare le metriche funzionali
  - Definire i criteri per accreditare i professionisti che le usano
- **Concetti definiti**
  - Accuratezza di una misura funzionale
  - Accuratezza di una metrica funzionale
  - Ripetibilità e riproducibilità di una metrica funzionale
  - Soglia di sensibilità di una metrica funzionale
  - Applicabilità ad un dominio funzionale

Metriche per il mondo OO

- **Occorrono metriche dedicate**
  - Le metriche tradizionali non sono accurate
  - Complessità funzionale e complessità strutturale
  - Non linearità del codice (la misura SLOC non funziona!)
  - Uso di strutture e funzioni complesse (McCabe non funziona come misura assoluta!)
- **Metriche per i metodi**
- **Metriche per le classi**
- **Metriche per i sistemi**

Metrica per i metodi

- **Un metodo è sostanzialmente procedurale**
  - Si può usare McCabe come metrica di base
  - Corretta per tenere conto di interfacce e variabili locali
- **Complessità di un metodo**

$MC = w_1 MIC + w_2 MLVC + w_3 MCoC$

MIC	complessità dell'interfaccia
MLVC	complessità delle variabili locali
MCoC	complessità ciclomatica del codice
w_i	pesi determinati statisticamente

Metrica per le classi

- **Complessità di una classe**

$CC = w_3\ CCL + w_4\ CCI$	
CCI	complessità dell'I/F della classe
$CCL = w_5\ ECCL + w_6\ ICCL$	complessità locale
$ECCL = \sum_j MIC_j$	complessità locale esterna
$ICCL = w_7\ CACL + w_8\ CMCL$	complessità locale interna
$CACL = \sum_h CA_h$	complessità degli attributi locali
$CMCL = \sum_k MC_k$	complessità dei metodi locali
CA	costante o CC di classe attributo
w_i	pesi determinati statisticamente

- **Proprietà della metrica per le classi**
  - Considera solo la parte locale della classe
  - Considera la complessità dell'uso delle classi negli attributi

Metrica per i sistemi

- **Complessità del sistema**

  $SC = \sum_n CC_n$

- **Metriche utilizzabili come fattori d'influenza**

NC	Numero totale di classi
NRC	Numero di classi radice
NLC	Numero di classi foglia

Accuratezza delle metriche

- **Precisione**
  - In termini di esattezza e ripetibilità:
    SLOC e McCabe sì, FP no
- **Aderenza alla realtà**
  - Se la misura corrisponde all'evidenza sperimentale
  - Se le metriche sono usate come indicatori non in termini assoluti

La qualità del software

Università di Ferrara
Corso di Ingegneria del Software
AA 2013/2014

La Qualità del Software

- **Percepita (in uso)**
  Esprime l'efficacia ed efficienza, con cui il software serve le esigenze dell'utente, ma anche l'usabilità del prodotto.
- **Interna (intrinseca)**
  Esprime la misura in cui il codice software possiede una serie di attributi, indipendentemente dall'ambiente di utilizzo (analisi statica del codice sorgente).
- **Esterna**
  Esprime le prestazioni del software nel suo ambiente di utilizzo (comportamento del codice in esecuzione).

ISO IEC/9126

La norma ISO/IEC 9126:2001 definisce sei caratteristiche principali che costituiscono la qualità di un prodotto software

1 – Funzionalità

Capacità di fornire funzioni tali da soddisfare, in determinate condizioni, requisiti funzionali espliciti o impliciti (il software fa ciò per cui è stato sviluppato). Sottocaratteristiche:

- **Adeguatezza**: presenza di funzioni appropriate per compiti specifici
- **Accuratezza**: capacità di fornire risultati od effetti in accordo con i requisiti
- **Interoperabilità**: capacità di interagire con altri sistemi.
- **Sicurezza**: capacità di evitare accessi non autorizzati a programmi e dati

2 – Affidabilità

Capacità di mantenere le prestazioni stabilite nelle condizioni e nei tempi stabiliti. Sottocaratteristiche:

- **Maturità (robustezza):** capacità di evitare fermi della applicazione a seguito di errori nel software
- **Tolleranza errori:** capacità di mantenere livelli di prestazioni predeterminati in caso di fallimenti o di violazione delle regole di interfacciamento
- **Recuperabilità:** capacità di ripristinare livelli di prestazione predeterminati e di recuperare i dati a seguito di errori

3 – Usabilità

Capacità del sw di essere compreso, appreso, usato con soddisfazione dall'utente in determinate condizioni d'uso. Sottocaratteristiche:

- **Comprensibilità**: impegno richiesto agli utenti per capire il funzionamento del software e la sua applicabilità.
- **Apprendibilità**: impegno richiesto agli utenti per imparare a usare il software.
- **Operabilità**: impegno richiesto agli utenti per usare il software.
- **Attrattività**: capacità del software di essere piacevole per l'utente che ne fa uso.

4 – Efficienza

Rapporto fra prestazioni e quantità di risorse utilizzate, in condizioni normali di funzionamento. Sottocaratteristiche:

- **Comportamento rispetto al tempo**: tempi di risposta e di elaborazione richiesti per eseguire le funzioni richieste in determinate condizioni.
- **Uso di risorse**: quantità e tipo di risorse usate per seguire le funzioni richieste in determinate condizioni.

5 – Manutenibilità

Capacità del software di essere modificato con un impegno contenuto. Sottocaratteristiche:

- **Analizzabilità**: impegno richiesto per diagnosticare carenze o cause di fallimento, o per identificare parti da modificare.
- **Modificabilità**: impegno richiesto per modificare, rimuovere errori o sostituire componenti.
- **Stabilità**: capacità di ridurre il rischio di comportamenti inaspettati a seguito di modifiche.
- **Provabilità**: impegno richiesto per validare le modifiche apportate al software.

6 – Portabilità

Facilità con cui il software può essere trasferito da un ambiente operativo ad un altro. Sottocaratteristiche:

- **Adattabilità**: capacità adattiva a nuovi ambienti operativi applicando le sole azioni previste per questo motivo da parte del software stesso.
- **Installabilità**: impegno richiesto per installare il software in un particolare ambiente.
- **Coesistenza**: capacità di coesistenza con altri software nel medesimo ambiente, condividendo risorse.
- **Sostituibilità**: capacità di essere utilizzato al posto di un altro software.

Livelli di qualità

- La qualità è un livello di possesso di determinati attributi che soddisfa delle esigenze.
- Il livello di qualità dei vari attributi dipende da una serie di fattori quali: la classe di rischio del software che si deve sviluppare, la tipologia di software, i requisiti del Cliente, il contesto d'uso dell'utilizzo del software.

Livelli di qualità – esempio

Caratteristica	Sottocaratteristica	rilevanza
Funzionalità	Adeguatezza	1
	Accuratezza	2
	Interoperabilità	2
	Sicurezza	1
Affidabilità	robustezza	1
	Tolleranza errori	1
	Recuperabilità	1
Usabilità	Comprensibilità	2
	Apprendibilità	2
	Operabilità	1
	Attrattività	3
Efficienza	Comportamento rispetto al tempo	1
	Uso di risorse	1
Manutenibilità	Analizzabilità	1
	Modificabilità	2
	Stabilità	1
	Provabilità	1
Portabilità	Adattabilità	3
	Installabilità	3
	Coesistenza	3
	Sostituibilità	3