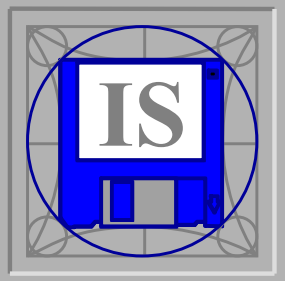
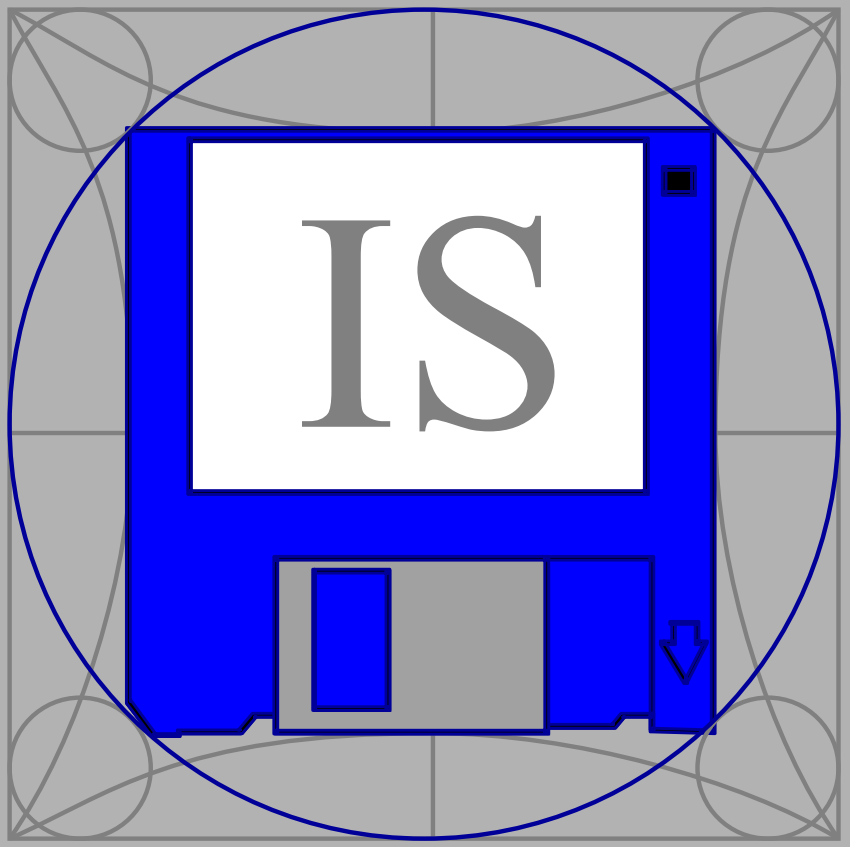


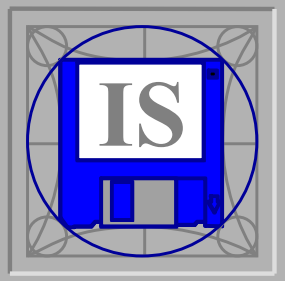
Qualità del *software*



Ingegneria del Software

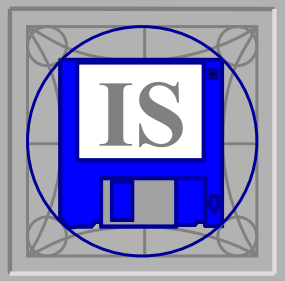
V. Ambriola, G.A. Cignoni,
C. Montangero, L. Semini

Aggiornamenti di: T. Vardanega (UniPD)



Qualità: intuizione iniziale

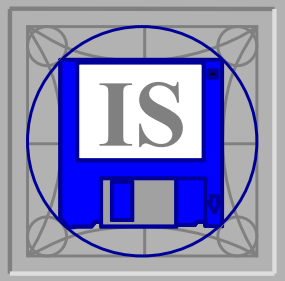
- Il concetto di qualità correla con quello di valutazione
- Si valuta per fare confronti
- O per determinare il grado di conformità alle attese
- Con destinatari diversi e punti di vista diversi
  - Chi fa
  - Chi usa
  - Chi valuta come terza parte
- Qui parliamo di qualità di prodotto (*software*)



Glossario ragionato – 1/4

Qualità:
Insieme delle caratteristiche di un'entità, che ne determinano la capacità di soddisfare esigenze sia espresse che implicite
(ISO 8402:1994, glossario dei termini, confluito in ISO 9000:2005)

- **Visioni della qualità**
  - **Intrinseca: conformità ai requisiti, idoneità all'uso**
  - **Relativa: soddisfazione del cliente**
  - **Quantitativa: misurazione oggettiva, per confronto**
- **In carico al Sistema Qualità del fornitore**

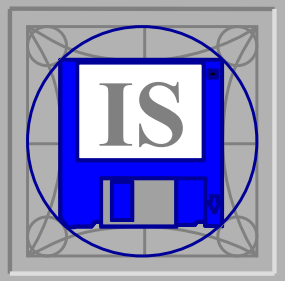


Glossario ragionato – 2/4

Sistema Qualità:
Struttura organizzativa, responsabilità, procedure, risorse, atte al perseguimento della qualità

(ISO 8402:1994 → ISO 9000:2005)

- **Si riassume in tre elementi**
  - **Piano della Qualità**
  - **Controllo di Qualità**
  - **Miglioramento continuo (parte dell'argomento T8)**
- **Il nostro Piano di Qualifica li comprende tutti**

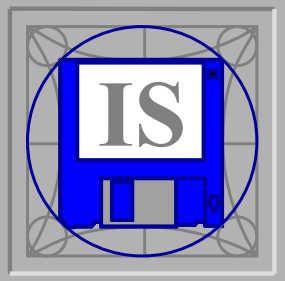


Glossario ragionato – 3/4

Piano della Qualità:
Le attività del Sistema Qualità mirate a fissare gli obiettivi di qualità, insieme con i processi e le risorse necessarie per conseguirli

(ISO 9000)

- **Visione orizzontale, trasversale all'intera organizzazione**
  - Fissare le politiche aziendali per il perseguimento della qualità
- **Visione verticale, specifica di prodotto / servizio**
  - Fissare gli obiettivi di qualità del singolo progetto
- **Basate sull'adozione di uno specifico *way of working***
  - Grado di conformità riflesso nel corrispondente cruscotto di controllo



Glossario ragionato – 4/4

Controllo di Qualità:
Le attività del Sistema Qualità pianificate e attuate per assicurare che il prodotto soddisfi le attese

(ISO 9000)

- **Prevenire è meglio che curare …**
  - **Assicurare conformità passo-passo invece che solo a fine corsa**
  - **Attuare il *way of working***
  - **Controllarne gli effetti tramite il cruscotto di controllo (modo non invasivo sulle attività)**
- **Questo è *Quality Assurance* (accertamento di qualità)**

I principi del Sistema Qualità

Seven Quality Management Principles

1. CUSTOMER FOCUS
Meet and exceed customer expectations

2. LEADERSHIP
Provide purpose, direction and engagement.

3. ENGAGEMENT OF PEOPLE
Recognition, empowerment and enhancement of skills and knowledge.

4. PROCESS APPROACH
Understand processes to optimize performance.

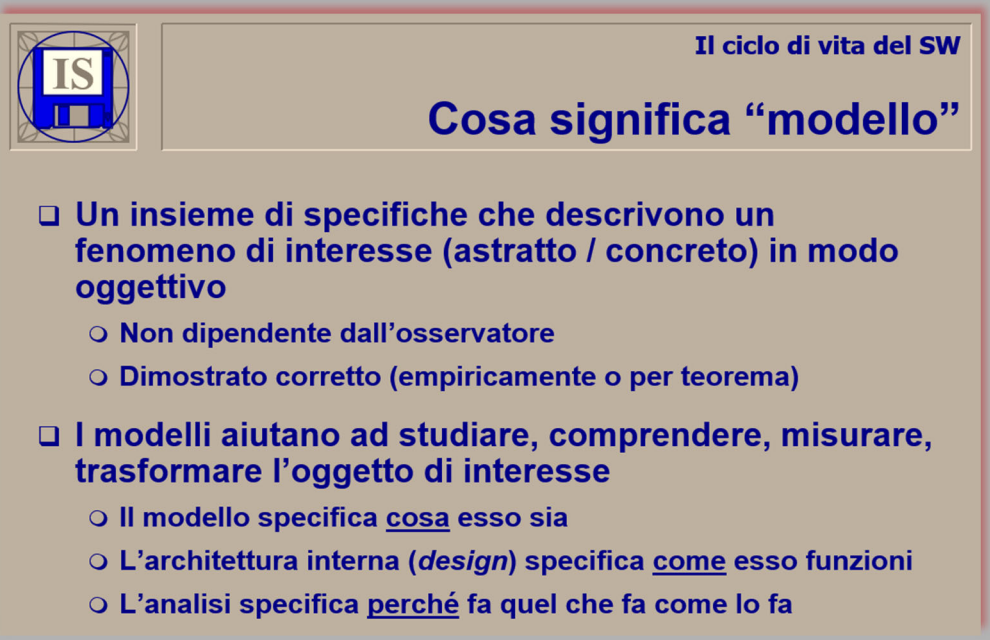
5. IMPROVEMENT
To maintain current performance and to create new opportunities.

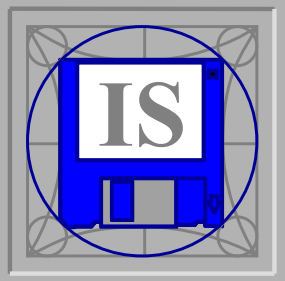
6. EVIDENCE-BASED DECISION MAKING
Facts, evidences and data analysis for decision making.

7. RELATIONSHIP MANAGEMENT
Manage relationship with interested parties to optimize performance.

Modelli della qualità SW – 1/3

- Servono per uniformare punti di vista diversi
- Lato sviluppo: prospettiva del progetto
- Lato uso: prospettiva utente
- Lato direzione: costi/benefici del *way of working*
- Servono per favorire valutazione oggettiva





Modelli della qualità SW – 2/3

- Cosa significa "qualità" in un prodotto SW
  - **ISO/IEC 9126:2001** ***SWE Product Quality***
    3 categorie, 7 caratteristiche principali,
    31 sotto-caratteristiche
- Metriche per la <u>valutazione quantitativa</u> della qualità
  - **ISO/IEC 14598:1999** ***SW Product Evaluation***
    3 prospettive (sviluppatore, committente, valutatore terzo)
- Oggi queste due dimensioni sono unificate
  - **ISO/IEC 25000:2005** ***SQuaRE: Systems and software Quality Requirements and Evaluation***

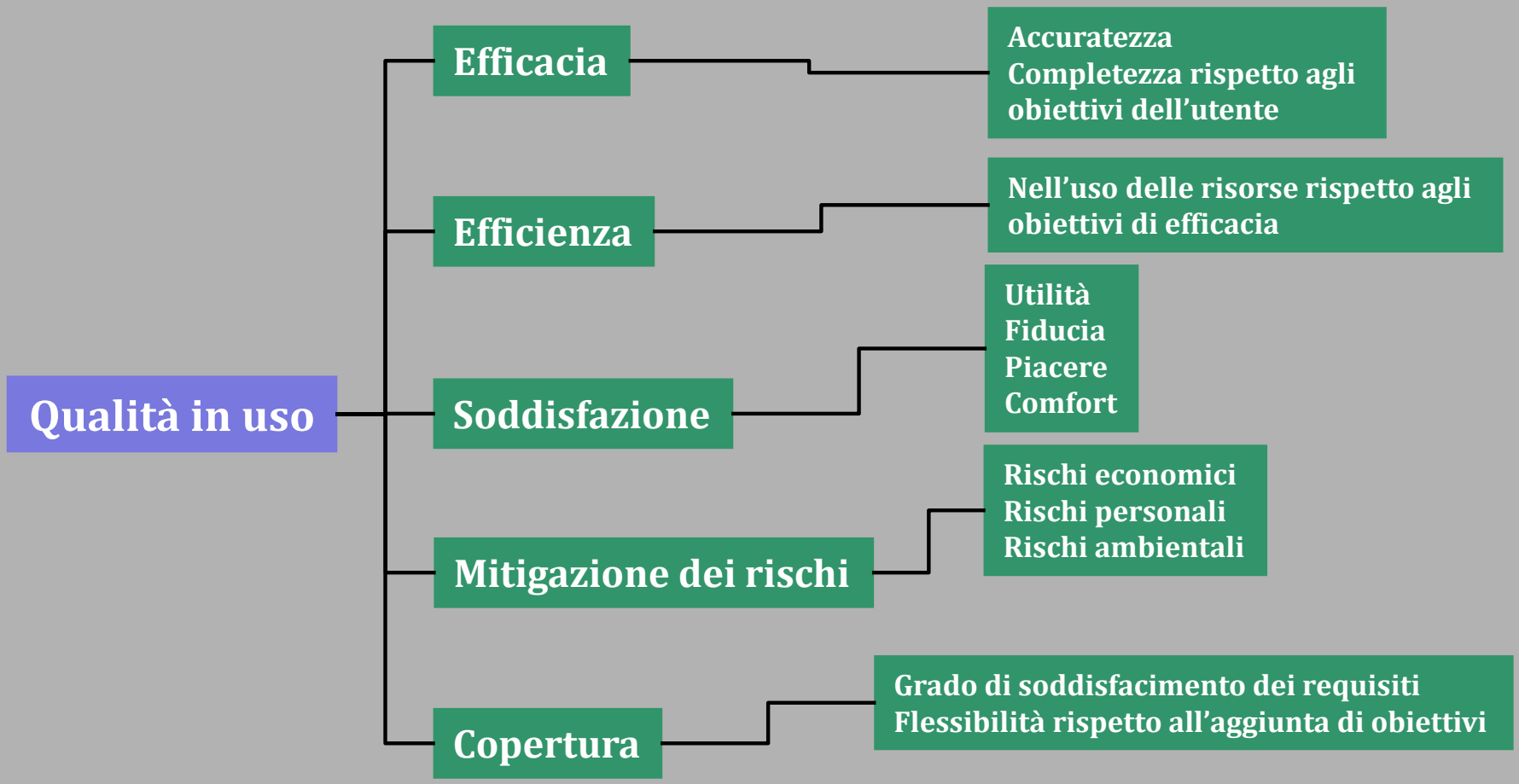
Modelli della qualità SW – 3/3

- 25010:2011 *Quality model*
  - Cosa significa «qualità SW»
- 25020:2019 *Quality measurement framework*
  - Come si misura la qualità SW
- 25030:2007 *Quality requirements*
  - Come si specificano i requisiti di qualità SW
- 25040:2011 *Quality evaluation*
  - Come si conduce la valutazione della qualità SW

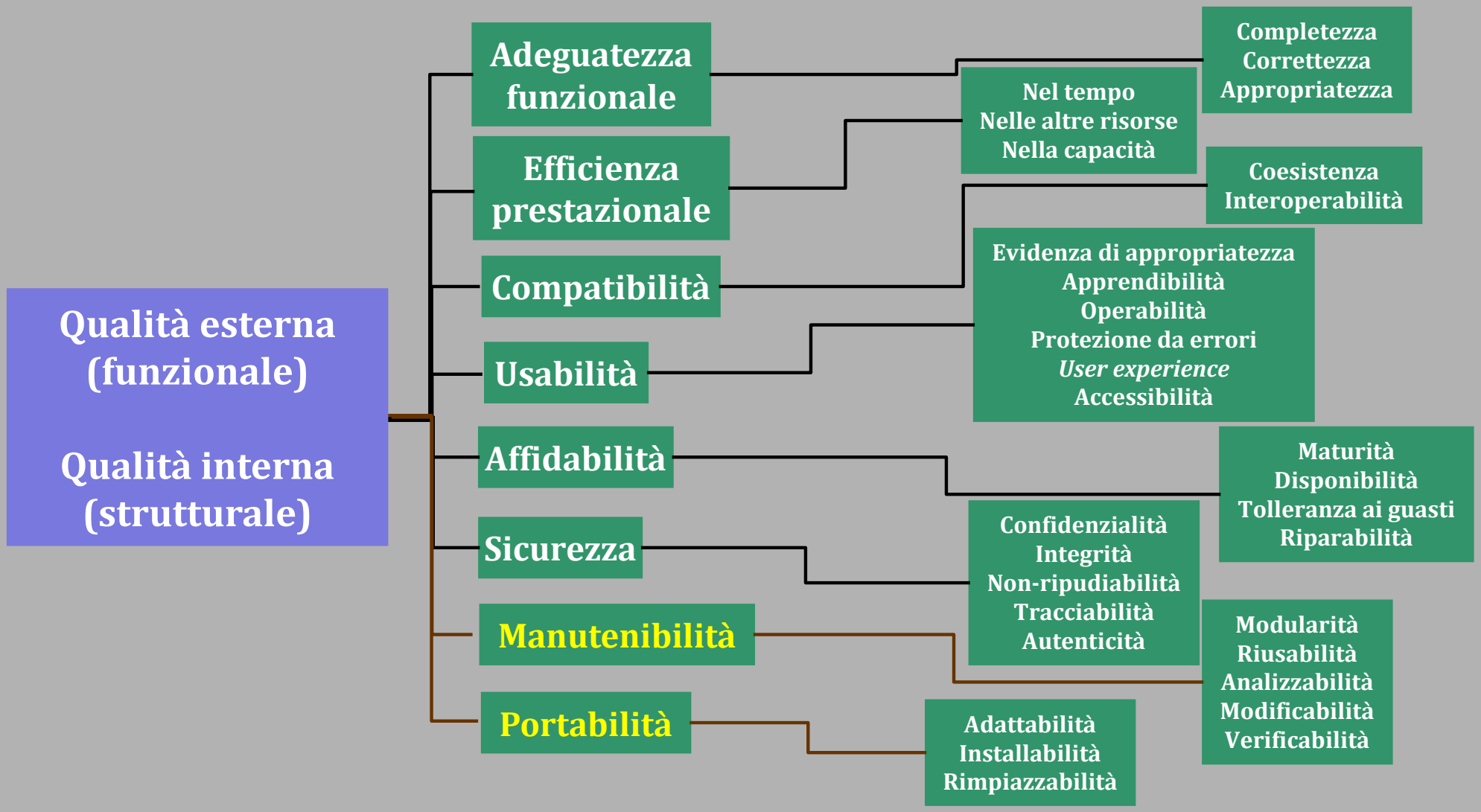
Misurazione quantitativa:
Il processo con cui assegnare simboli o numeri ad attributi di una entità, secondo regole definite

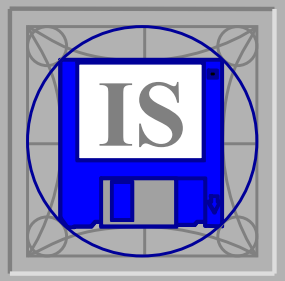
N. Fenton, *Software metrics, a rigorous approach*, 1997

Cosa significa qualità SW – 1/2



Cosa significa qualità SW – 2/2





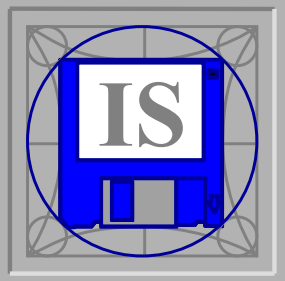
Software metrics

- ***Any type of measurement that relates to a software system, process or documentation***

Entity	Metric
Program	SLOC
Effort	Person/days
Text	Gunning's Fog index

Fog index = [(average # words / sentence) + (# words of 3 syllables or more)] * 0.4

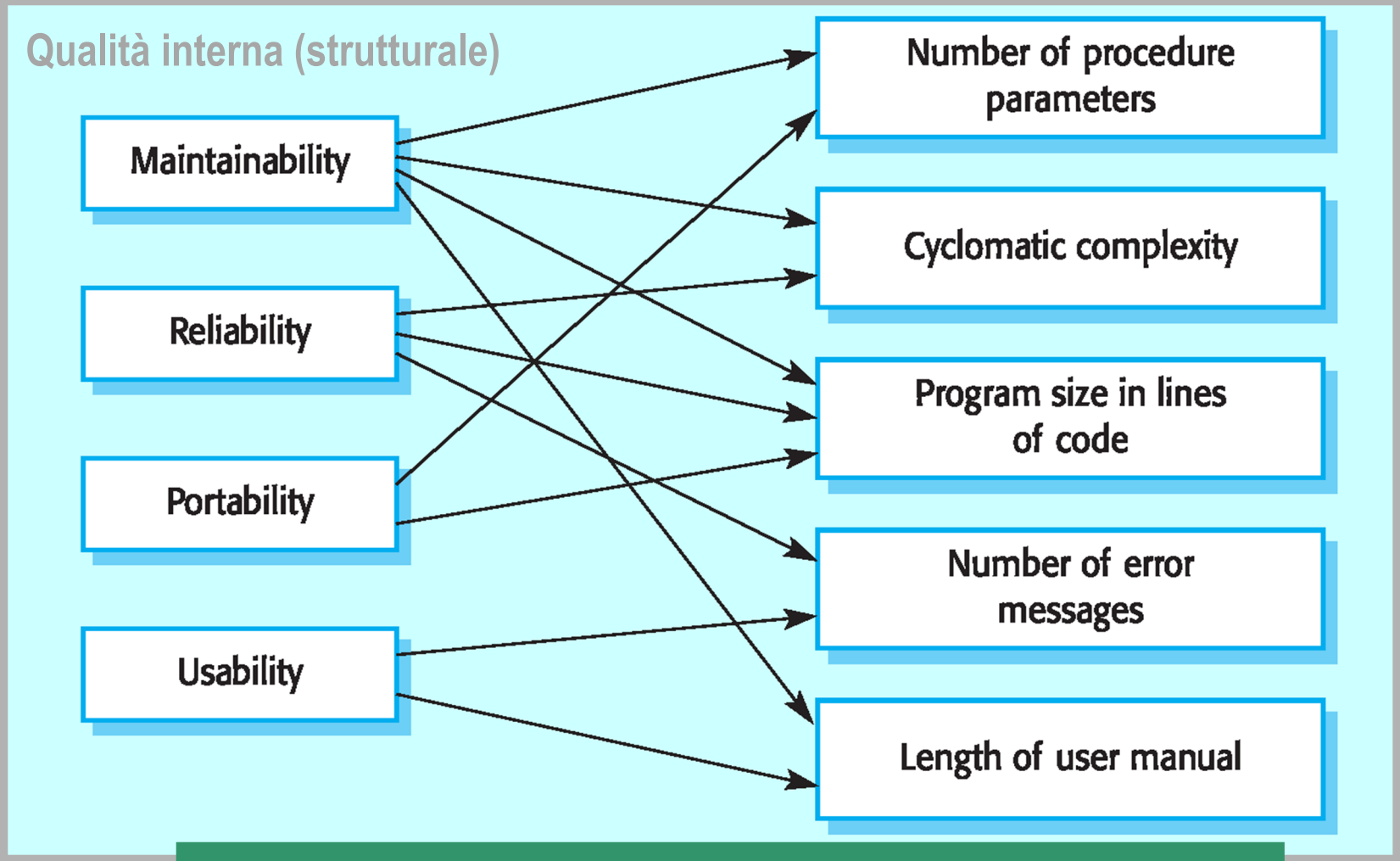
- ***Allow quantifying product and process attributes***
- ***May be used to predict product attributes or to control the software process***
- ***Product metrics can be used for general predictions or to identify anomalous components***



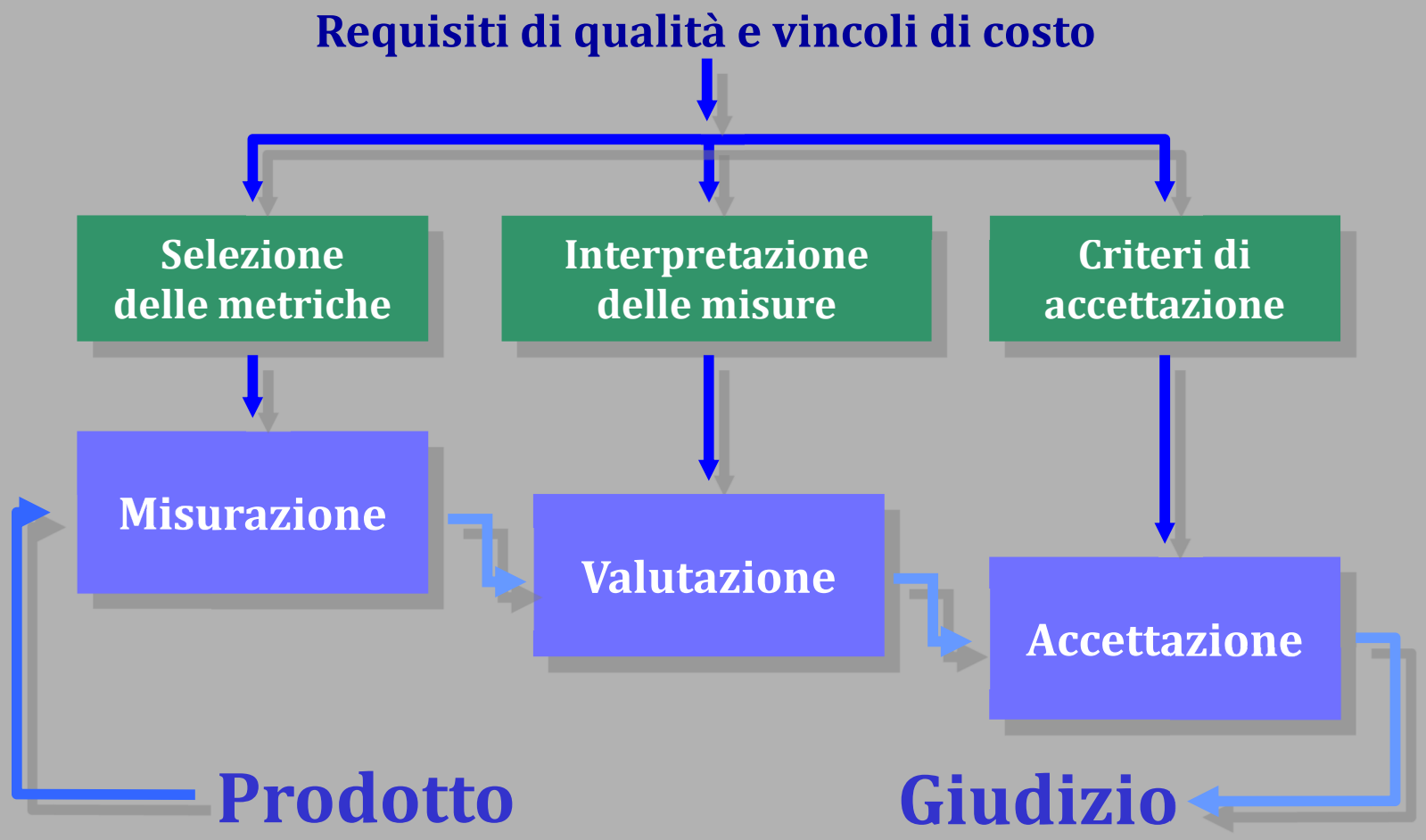
Assumptions on metrics

- *A software property or attribute can be measured*
- *A relationship exists between what we can measure and what we want to know*
- *We only know how to measure internal attributes*
  - *Product quality*
- *But we are often more interested in external attributes*
  - *Quality in use*
- *It may be difficult to relate what can be measured to desirable external quality attributes*

Fattori di influenza misurabili

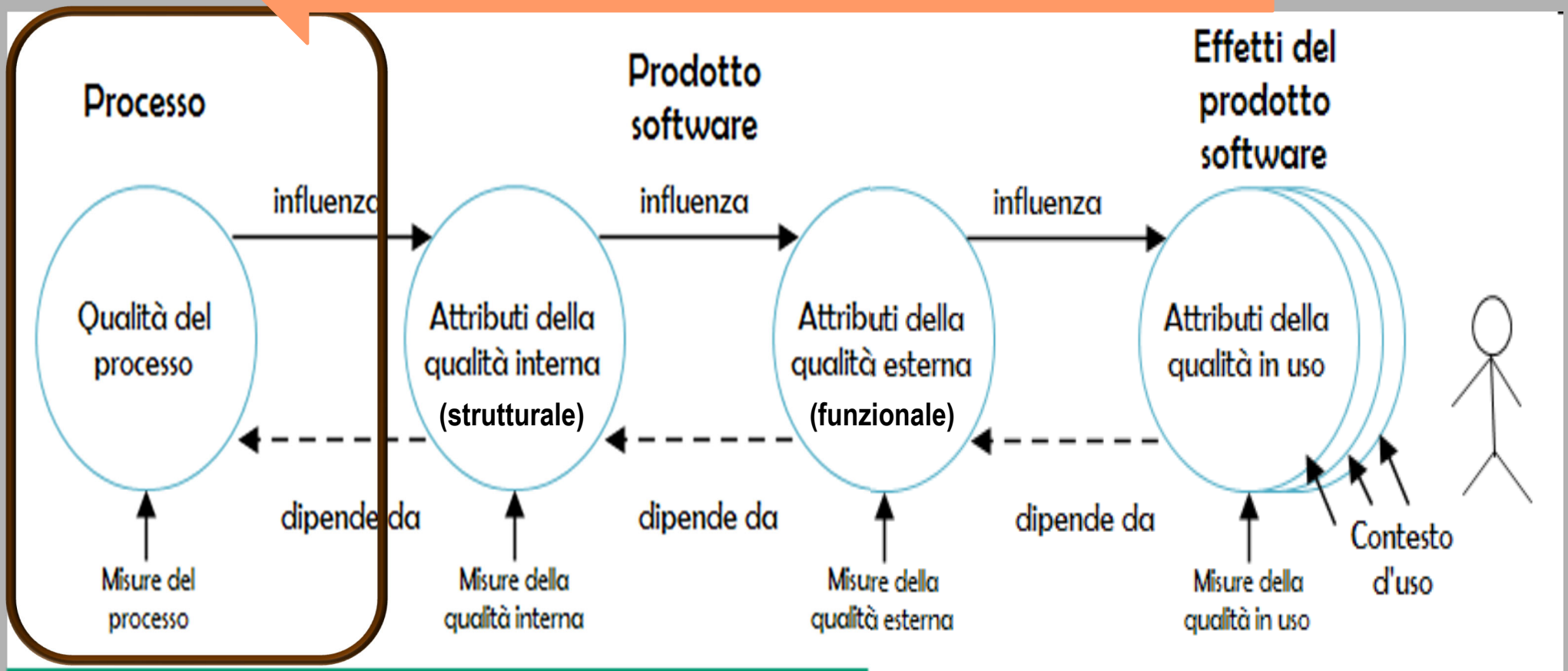


Il processo di valutazione



La qualità nel ciclo di vita del SW

Verso le vere cause



Di questo parleremo nella prossima lezione

Riferimenti

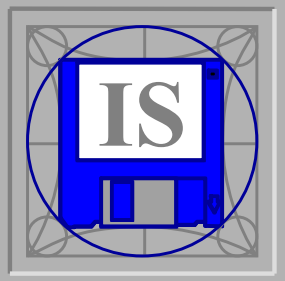
- ISO 9000:2000, Quality Management Systems – Fundamentals and vocabulary
- ISO/IEC 9126:2001, Information Technology – Software product quality – Part 1: Quality model
- ISO/IEC 14598:2001, Information Technology – Software Product Evaluation
- The ISO/IEC 25000 Series of Standards, https://iso25000.com

Appendice 1

QUALITÀ DI PROGETTAZIONE

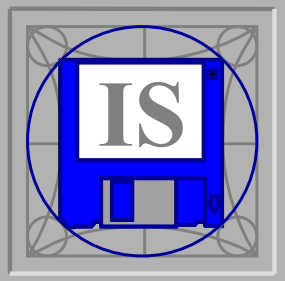
Premesse

- Non tutti i problemi hanno una (buona) soluzione
  - Conviene essere prudenti
- A ogni scelta progettuale devono corrispondere
  - Obiettivi (della scelta)
  - Vincoli (nella scelta)
  - Alternative (alla scelta)
  - Come la soluzione risponde al problema
- Fattibilità e verificabilità sono qualità cardine della progettazione



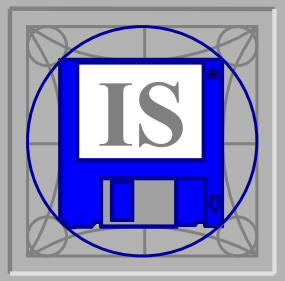
Qualità da ricercare

- La decomposizione modulare deve individuare componenti architetturali dotate di
  - Minimo accoppiamento
  - Massima coesione funzionale (autosufficienza)
- L'incapsulazione (*information hiding*) permette di nascondere il dettaglio realizzativo
  - Solo l'interfaccia è pubblica
  - Il dettaglio realizzativo è noto solo all'interno



Controllare accoppiamento e coesione

- L'accoppiamento è indicatore dell'intensità di relazione tra parti distinte
  - La modifica di una comporta modifiche nell'altra
  - Forte accoppiamento indica cattiva modularità
- La coesione è indicatore dell'intensità di relazione all'interno di una singola parte
  - Forte coesione indica buona modularità

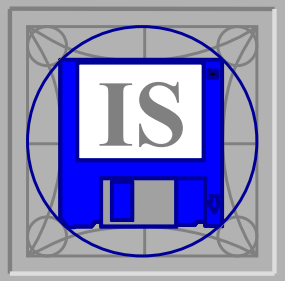


Ricercare integrità concettuale

- Facilmente riconoscibile nelle architetture fisiche
  - Suggerisce adesione a uno stile uniforme
  - Coerente in tutte le parti del sistema e nelle loro interazioni
- Bilancia ricchezza funzionale con semplicità d'uso
- Desiderabile in ogni architettura di sistema
  - Anche nel *software*
- Richiede osservanza e vigilanza
  - Facilita parallelismo nella realizzazione

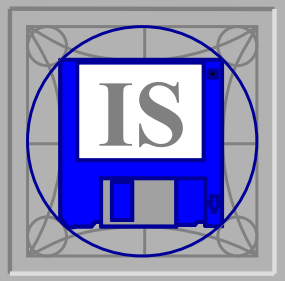
Avvertenze – 1/3

- L'**astrazione** omette informazione per poter applicare operazioni simili a entità diverse
  - Ciò che è caratteristico dell'intera gerarchia è fissato in radice
  - Ciò che differenzia si aggiunge per specializzazione allontanandosi dalla radice
- L' astrazione è usabile solo tramite concretizzazione
  - Per parametrizzazione
    (p.es.: da *template* a entità concreta in C++)
  - Per specializzazione
    (p.es.: da interfaccia a classe in Java e C++)
  - Per valorizzazione
    (p.es.: da classe a oggetto tramite costruttore in OOP)



Avvertenze – 2/3

- La concorrenza di buona qualità aiuta a decomporre il sistema in più entità autonome garantendo
  - Efficienza di esecuzione
  - Atomicità di azione
  - Consistenza e integrità dei dati condivisi
  - Semantica precisa di comunicazione e serializzazione
  - Predicibilità di ordinamento temporale
- La distribuzione di buona qualità ripartisce il sistema in programmi disseminati su nodi distinti garantendo
  - Bilanciamento di carico
  - Frugalità nelle comunicazioni (buon grado di autonomia)

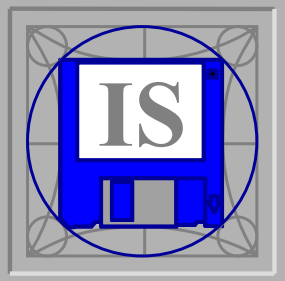


Avvertenze – 3/3

- Eventi ed errori critici emergono più facilmente in presenza di concorrenza o distribuzione
  - Per evitarli servono alcune accortezze
- In relazione al flusso dei dati
  - Saper determinare quando un certo dato è disponibile
- In relazione al flusso di controllo
  - Saper determinare l'ingresso in un particolare stato
- In relazione al trascorrere del tempo
  - Saper determinare l'arrivo di un certo istante temporale
- Mai fare assunzioni ottimistiche!

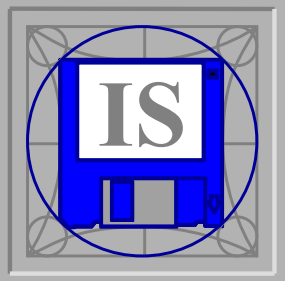
Appendice 2

DALLA PROGETTAZIONE ALLA CODIFICA



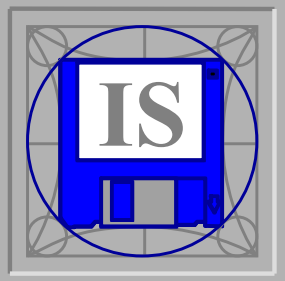
Consigli: *enforce intentions*

- Rendere chiaro – nel codice – il confine tra esterno e interno dei moduli
- Decidere chiaramente e codificare coerentemente ciò che può essere specializzato
  - Rendere il resto immodificabile (`final`, `const`, ...)
- Proteggere tutto ciò che non deve essere visto e acceduto dall'esterno
  - `Private`, `protected`, ...
- Decidere quali classi possono produrre istanze e quali no
  - Usare il *pattern* Singleton per le classi a istanza singola



Consigli: *defensive programming*

- Programmare esplicitamente il trattamento dei possibili errori
- Nei dati in ingresso
  - Verificarne la legalità prima di usarli
- Nella logica funzionale
  - Preservare proprietà (invarianti, pre/post-condizioni,…)
- Definire la strategia di trattamento degli errori (*error handling*) è compito della progettazione



Gestire gli errori nei dati in ingresso

- Possibili tecniche di trattamento
  - Attendere fino all'arrivo di un valore legale
  - Assegnare un valore predefinito (*default*)
  - Usare un valore precedente
  - Registrare l'errore in un *log* persistente
  - Sollevare una eccezione gestita
  - Abbandonare il programma