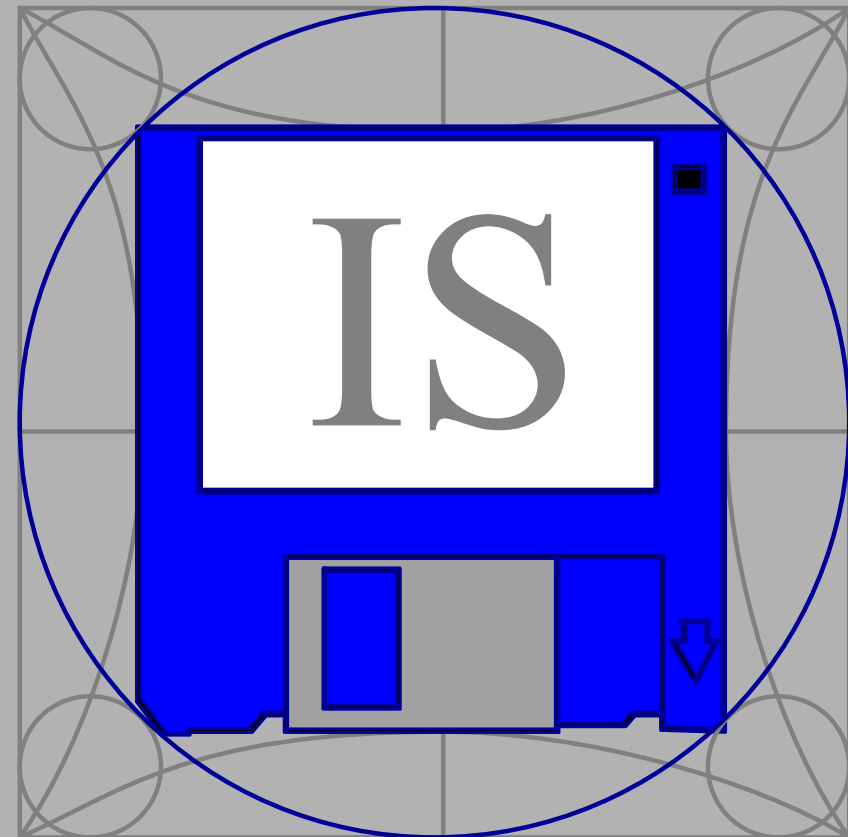


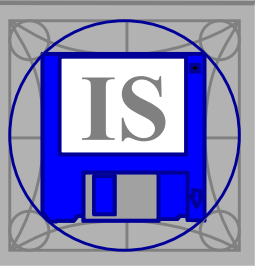
Qualità di processo

Ingegneria del Software

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

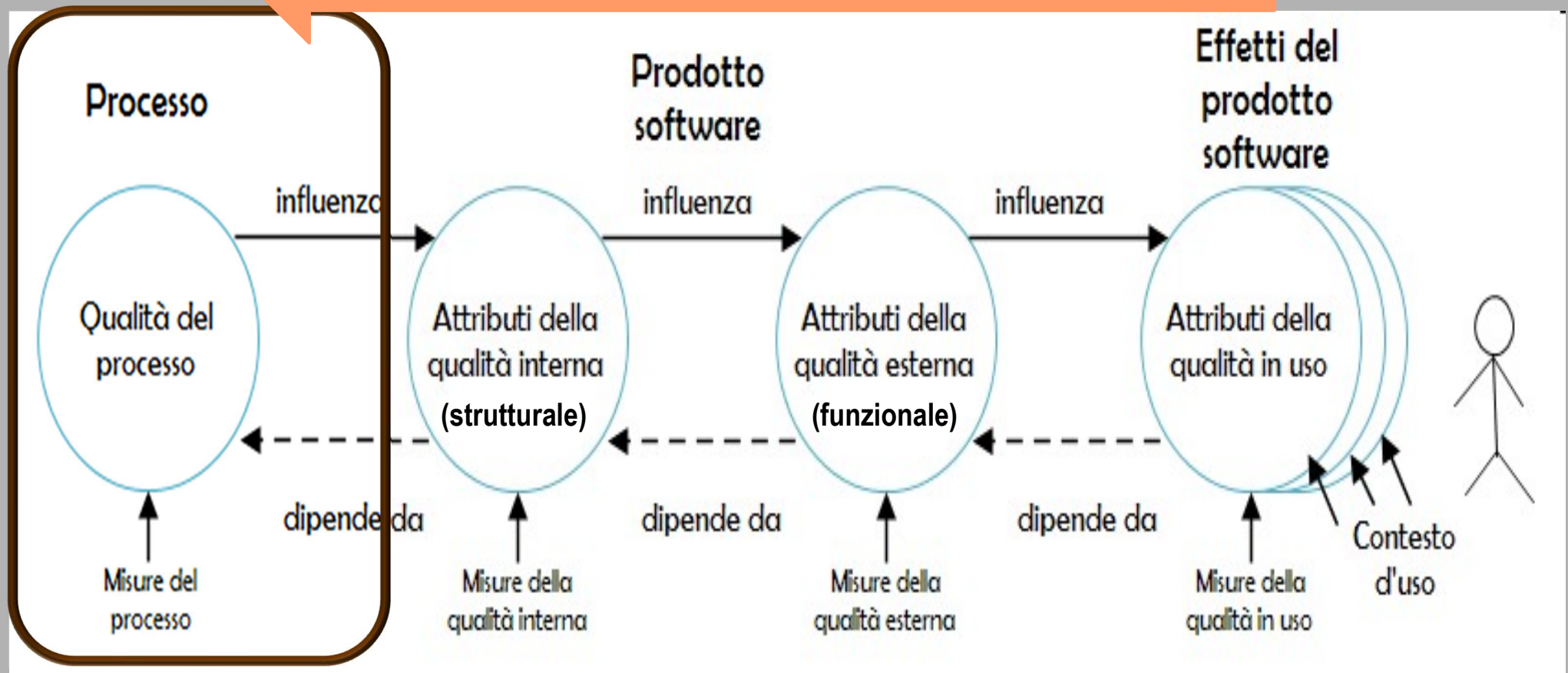
Aggiornamenti: T. Vardanega (UniPD)

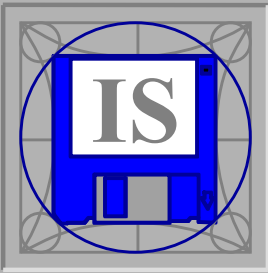




Visione d'insieme

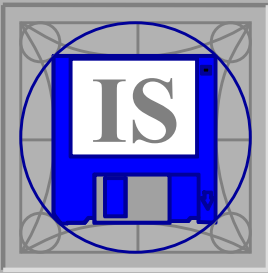
Verso le vere cause



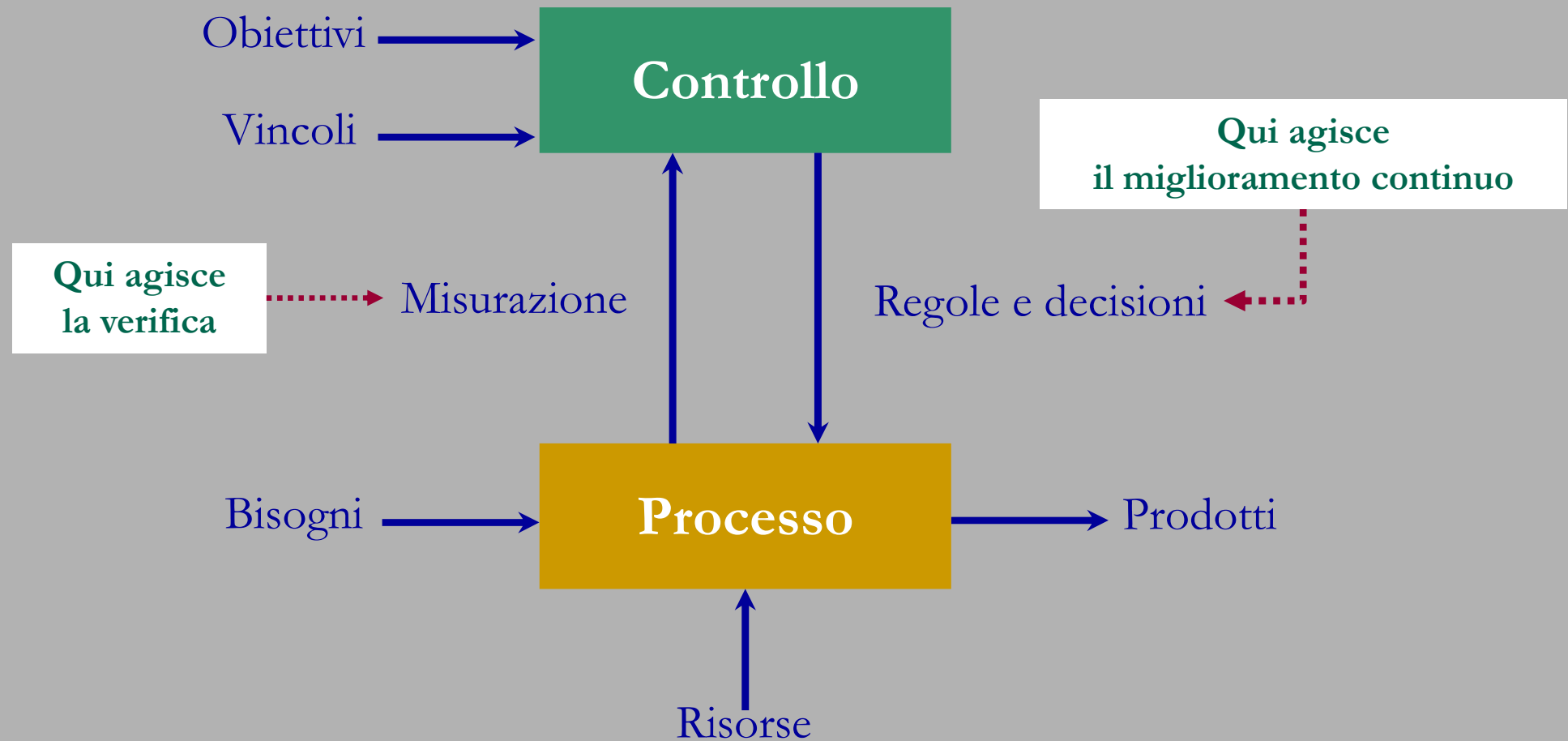


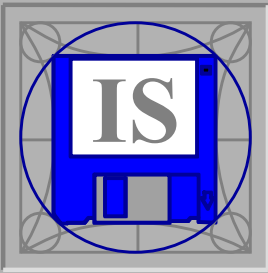
Dal prodotto al processo

- ❑ **Da tubi sporchi non esce acqua pulita**
- ❑ **La qualità di processo è esigenza primaria, che richiede**
 - **Adozione sistematica piuttosto che occasionale**
 - **Verifica costante, preventiva prima che reattiva**
 - **Valutazione riproducibile e quindi automatizzata**
 - **Disposizione costante al miglioramento**



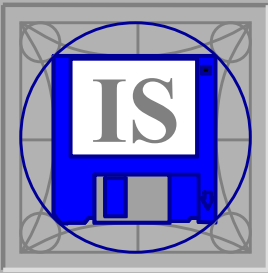
Modello concettuale di processo





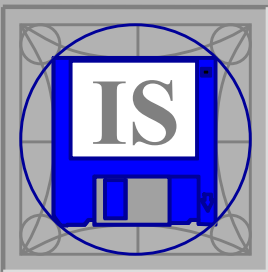
Per perseguire qualità di processo

- ❑ **Prima definire il processo (parte del *way of working*)**
 - Per poterlo applicare coerentemente
 - Per poterne valutare gli effetti in modo ragionevole
- ❑ **Poi controllare il processo al fine di migliorarlo**
 - In efficacia: prodotti conformi alle attese
 - In efficienza: minori costi a pari qualità di prodotto
 - In esperienza: apprendere dall'esperienza (anche di altri)
- ❑ **Servono buone metriche e buoni strumenti di valutazione**

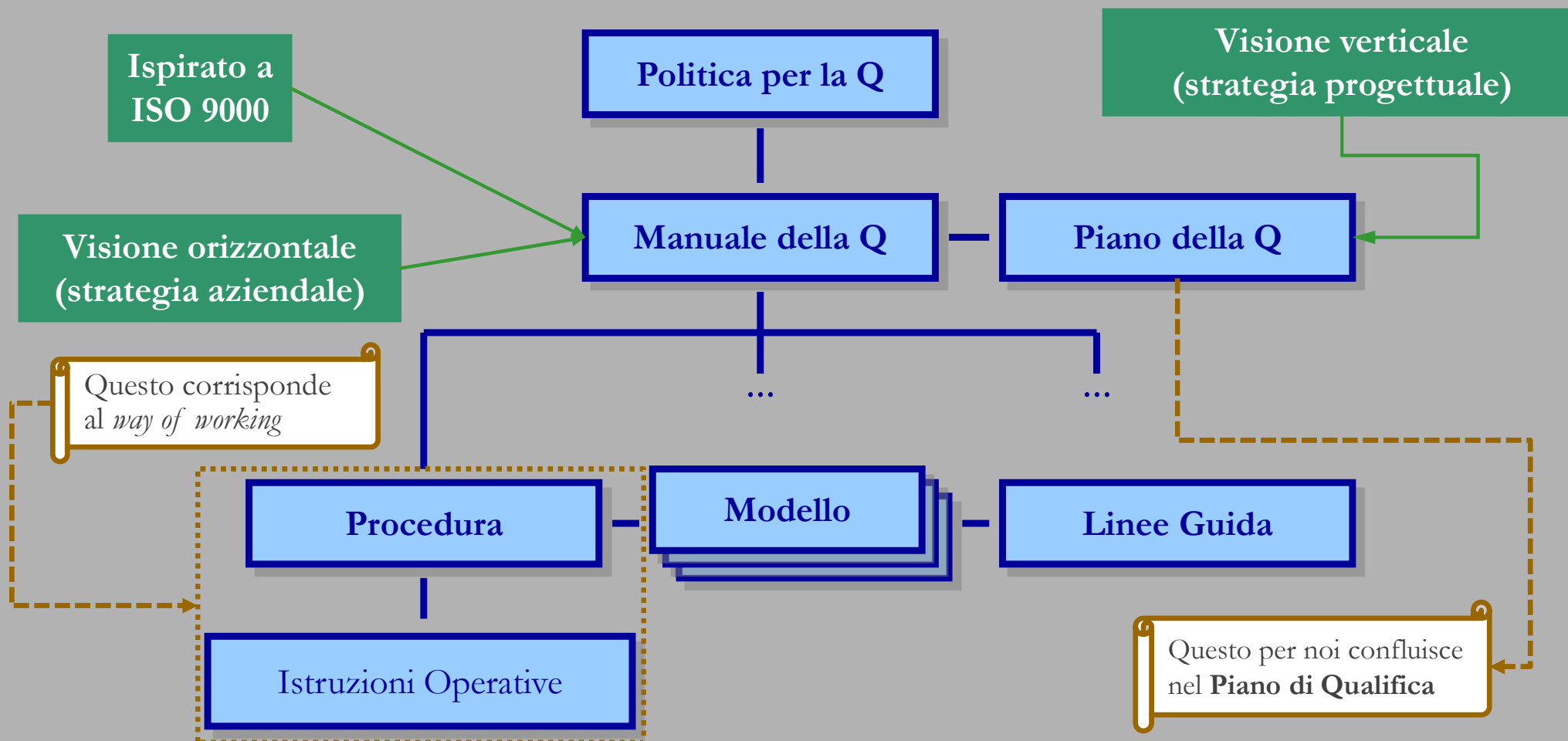


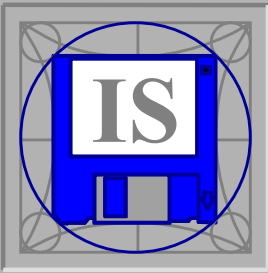
La famiglia delle norme ISO 9000

- ❑ **ISO 9000:2015** (fondamenti e glossario)
 - Modello di qualità neutro rispetto al dominio
- ❑ **ISO 9001:2015** (sistema qualità – requisiti)
 - La visione ISO 9000 calata nei sistemi produttivi
 - **ISO/IEC/IEEE 90003:2018** (ISO 9001:2015 applicato a prodotti SW)
- ❑ **ISO 9004:2018** (qualità organizzativa - autovalutazione)



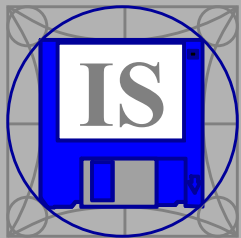
Documentazione del Sistema Qualità





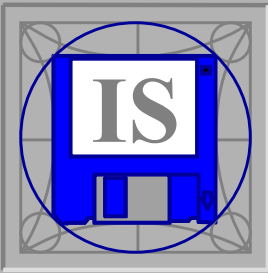
Valutazione della qualità di processo

- ❑ **SW Process Assessment & Improvement (SPY)**
 - Valutazione della maturità dei processi e auto-miglioramento
- ❑ **CMM (*Capability Maturity Model*, 1987)**
 - Dal SEI @ CMU al DoD per la valutazione dei fornitori
 - Modello di valutazione delle organizzazioni
 - Poi esteso al *service management* come **CMMI**
- ❑ **SPICE (*Software Process Improvement Capability dEtermination*, 1992)**
 - Per armonizzare SPY con ISO/IEC 12207 e ISO 9001
 - Poi confluito in **ISO/IEC 330xx: 2015 (*Process assessment*)**



L'idea base del modello SPY

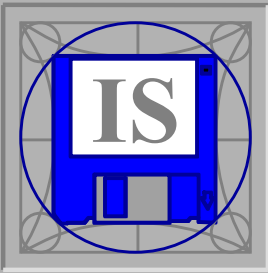




Il passo successivo: CMMI

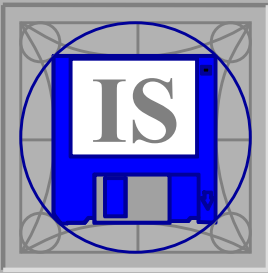
- ❑ **CAPABILITY**: misura l'adeguatezza (efficienza ed efficacia) di un singolo processo per gli scopi a esso assegnati
- ❑ **MATURITY**: misura quanto bene l'organizzazione è governata dal suo insieme di processi
 - Il *bottom* di *capability* dei processi valutati
- ❑ **MODEL**: insieme di criteri di valutazione (in scala assoluta)
- ❑ **INTEGRATION**: architettura di integrazione delle diverse discipline (system, HW, SW) e tipologie di attività delle organizzazioni
 - Sviluppo di prodotti e servizi (CMMI-DEV)
 - Gestione ed erogazione di servizi (CMMI-SVC)
 - Approvvigionamento di prodotti e servizi (CMMI-ACQ)



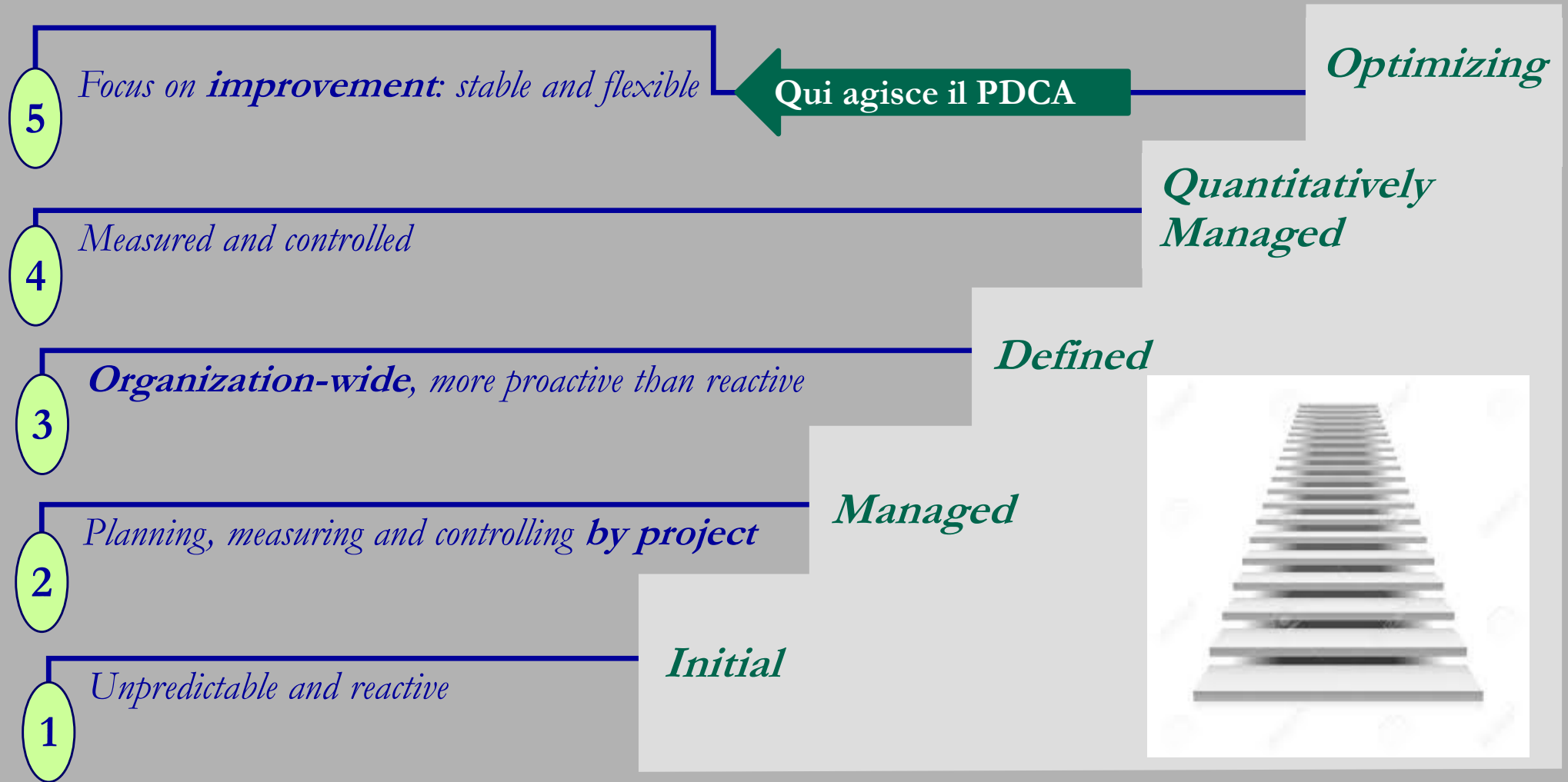


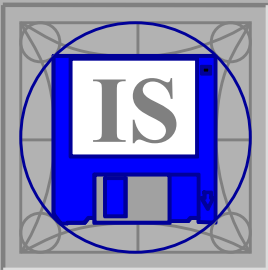
L'alto e il basso ...

- ❑ Un processo a basso livello di *capability*
 - Ha effetti imprevedibili, che dipendono da chi lo attua
 - Viene interpretato e attuato in modo opportunistico
 - Porta a compromessi tra qualità e consegna
- ❑ Un processo ad alto livello di *capability*
 - È seguito da tutti in modo disciplinato, sistematico e quantificabile
- ❑ L'intelligenza dei processi di una organizzazione si chiama *governance*
 - Sapere il perché delle proprie scelte, per efficacia, efficienza, e relazione con le *best practice*
 - Visione sul futuro












I 5 livelli di maturità

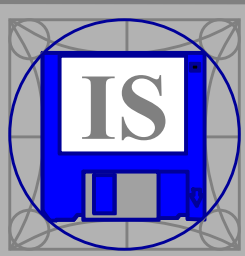




Un esempio per analogia

Orientarsi in terreno sconosciuto

5	Focus on improvement : ho anche informazioni dinamiche sulle congestioni (ottimizzare il percorso scegliendolo a seconda della situazione)	 
4	Measured and controlled : la cartina stradale è arricchita di indicazioni numeriche precise sulle distanze (gestire il viaggio quantitativamente)	 
3	Organization-wide : dispongo di una cartina stradale (corrispondente alla mappa dei processi condivisa a livello di organizzazione)	
2	By project : la persona cui chiedo potrebbe fornirmi indicazioni precise con riferimenti (sapendo mentre avanzo se sono sulla strada giusta), ma anche no	 
1	Unpredictable and reactive : chiedo a qualcuno, che mi fornisce indicazioni approssimative (magari arrivo; più probabilmente mi perdo)	 



ISO/IEC 33020:2019

Table 1 — Process capability level ratings

Scale	Process attributes	Rating
Level 1 Performed	Process Performance	
Level 2 Managed	Process Performance Performance Management Work Product Management	
Level 3 Established	Process Performance Performance Management Work Product Management Process Definition Process Deployment	
Level 4 Predictable	Process Performance Performance Management Work Product Management Process Definition Process Deployment Quantitative Analysis Quantitative Control	N not achieved $(0 \leq x \leq 15\%)$ P partially achieved $(15 < x \leq 50\%)$ L largely achieved $(50 < x \leq 85\%)$ F fully achieved $(85 < x \leq 100\%)$
Level 5 Innovating	Process Performance Performance Management Work Product Management Process Definition Process Deployment Quantitative Analysis Quantitative Control Process Innovation Process Innovation Implementation	

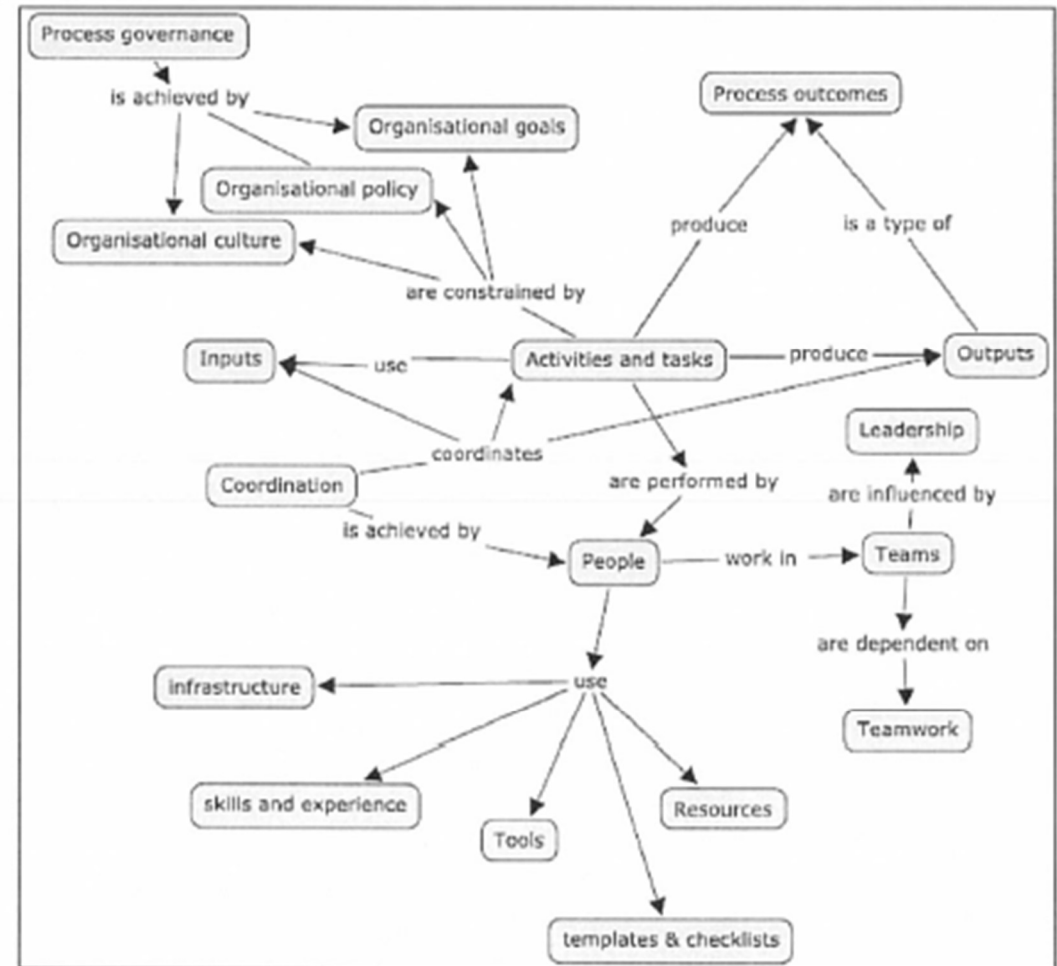
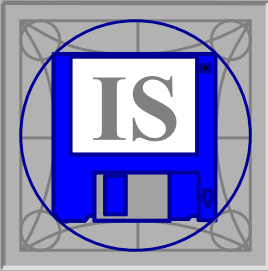
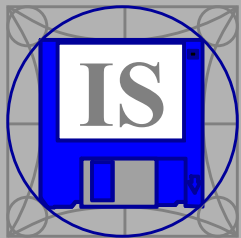


Figure B.1 — A process performance conceptual model



Why software fails

- **IEEE Spectrum (2 September 2005)**
<http://spectrum.ieee.org/computing/software/why-software-fails>
- ***As of Jan-2005, nearly 2000 government and commercial organizations [in the USA] voluntarily reported their CMM levels***
- ***53% at level 1 / 2***
- ***30% at level 3***
- ***17% at level 4 / 5***

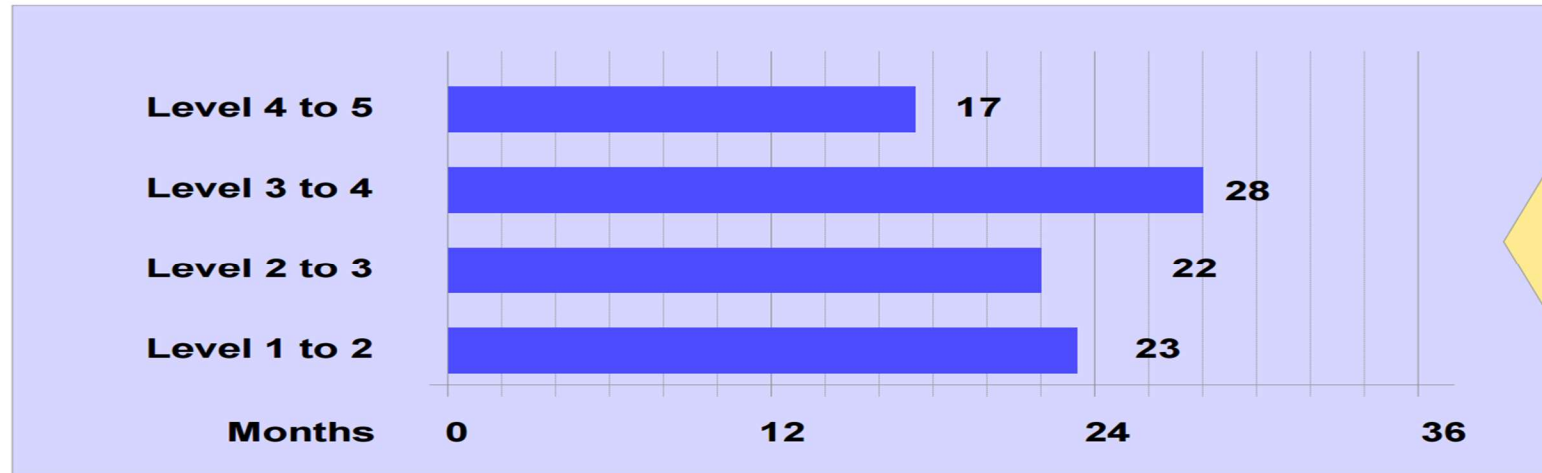


Costs and benefits

Benefit

Productivity growth (per year)	35 %
Increase of early defect detection (per year)	22 %
Reduction of time-to-market of a product (per year)	19 %
Reduction of field defects (per year)	39 %
Return on Investment	5.0

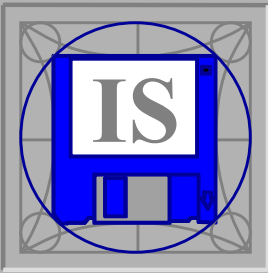
From: "Benefits of CMM-Based Software Process Improvement", Software Engineering Institute
Average of 13 organizations, using SW-CMM



A significant reduction of that time can be achieved by using existing experience and assets.

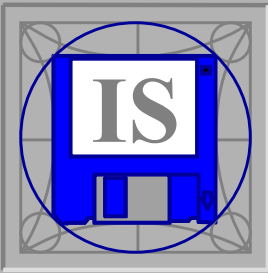
Average time used to reach the next maturity level in organizations, that have started their software process improvement in 1992 or later.

From: Software Engineering Institute, Process Maturity Profile of the SW Community, August 2002

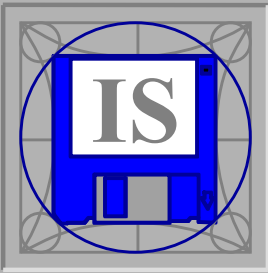


Esempio di valutazione SPICE

- ❑ **TOPS: *Towards organized software processes in SMEs* – FP4 (ESPRIT) 27977: 1998-2000**
 - Promuovere l'adozione di strumenti per il controllo della qualità nelle aziende della produzione SW
 - Formazione, valutazione dei processi
- ❑ **Valutazioni offerte come servizio**
 - Assaggio dei metodi SPY
 - Strumento di indagine
 - Strumento di confronto quantitativo (*benchmark*)



- ❑ **36 aziende localizzate nel centro Italia**
- ❑ **Per lo più di piccole dimensioni**
 - 21 (58%) con fatturato annuo < 1.000.000 €
 - 17 (46%) con < 10 dipendenti
- ❑ **Visione limitata della qualità**
 - Pochi S[G]Q certificati ISO 9001 (7,21%)
 - Crescita come obiettivo primario (21,57%)
 - Qualità solo come risposta a clienti o alla concorrenza (28,78%)



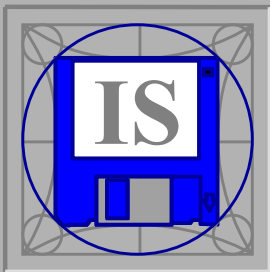
□ Processi valutati

- ENG.1.2 *Analisi dei requisiti*
- ENG.1.6 *Prove del software*
- SUP.6 *Joint review*

□ Rilevanti al rapporto con il committente

□ Miglioramento

- Valutazione non formale



<< 3

Media su tutte

5	N	N	N	N
4	N	N	N	N
3	P	P	P	P
2	L	P	P	P
1	L	P	L	L
	ENG.1.2	ENG.1.6	SUP.6	Media

non adeguato parzialmente

N

P

Solo le migliori

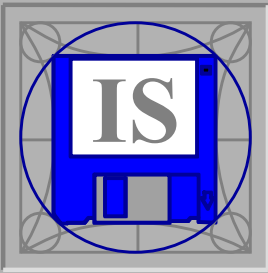
≤ 3

N	N	N	N
P	N	N	N
L	C	L	L
C	C	L	C
C	C	L	C
ENG.1.2	ENG.1.6	SUP.6	Media

largamente completamente

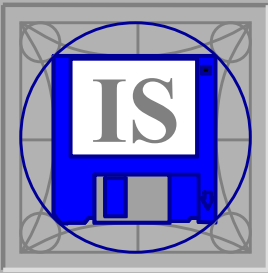
L

C



Appendice: qualità di progettazione

- ❑ Non tutti i problemi hanno una (buona) soluzione
- ❑ Per ogni scelta progettuale serve fissare con la massima chiarezza possibile
 - Obiettivi (della scelta)
 - Vincoli (nella scelta)
 - Alternative (alla scelta)
 - Come la soluzione corrisponde al problema
- ❑ Fattibilità e verificabilità sono qualità cardine della progettazione



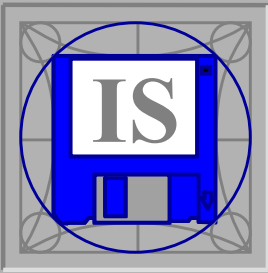
Buone tecniche di progettazione – 1/3

❑ Decomposizione modulare

- Una buona decomposizione architettuale identifica componenti tra loro indipendenti
- Minimo accoppiamento
- Autosufficienza (coesione funzionale)

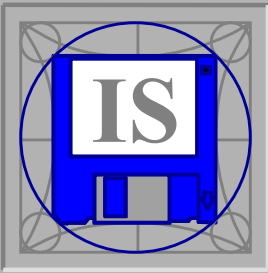
❑ Incapsulazione (*information hiding*)

- Nascondere il dettaglio realizzativo
- Solo l'interfaccia è pubblica
- Il dettaglio è noto solo all'interno



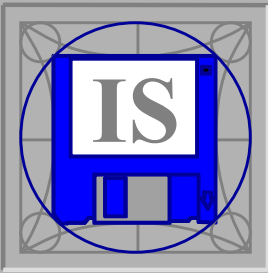
Buone tecniche di progettazione – 2/3

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



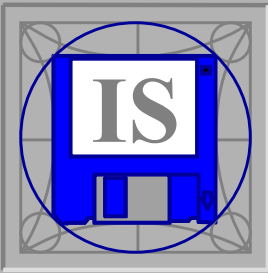
Buone tecniche di progettazione – 3/3

- ❑ 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
- ❑ A ogni astrazione corrisponde una 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)



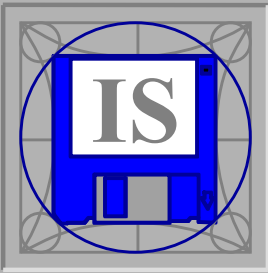
Problematiche critiche – 1/2

- ❑ 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 indipendenza



Problematiche critiche – 2/2

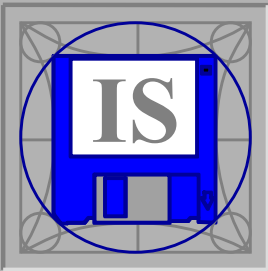
- ❑ Eventi ed errori in presenza di concorrenza o distribuzione
- ❑ In relazione al flusso dei dati
 - Saper determinare quando un certo dato è disponibile
- ❑ In relazione al flusso di controllo
 - Saper determinare l'ingresso in un particolare stato (dell'intero sistema o di una sua parte)
- ❑ In relazione al trascorrere del tempo
 - Saper determinare l'arrivo di certo istante temporale
- ❑ Mai fare assunzioni ottimistiche!



Ricerca integrità concettuale

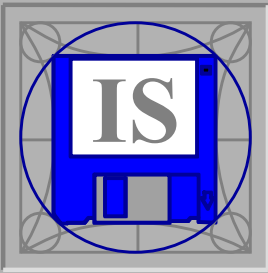
- ❑ **Facilmente riconoscibile nelle architetture fisiche**
 - Suggerisce adesione a uno stile uniforme
 - Coerentemente in tutte le parti del sistema e nelle loro interazioni
- ❑ **Bilancia ricchezza funzionale con semplicità d'uso**
- ❑ **Desiderabile in ogni architettura di sistema**
 - Anche nel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





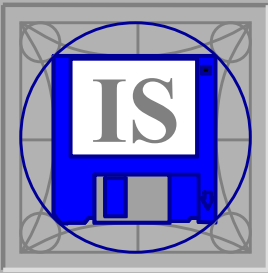
Consigli: *enforce intentions*

- ❑ Nel passaggio da progettazione a codifica
- ❑ Rendere chiaro il confine tra esterno e interno dei moduli
- ❑ Decidere chiaramente e codificare coerentemente ciò che può essere specializzato
 - Rendere il resto imm modificabile (`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: fissare proprietà (invarianti, pre- e 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