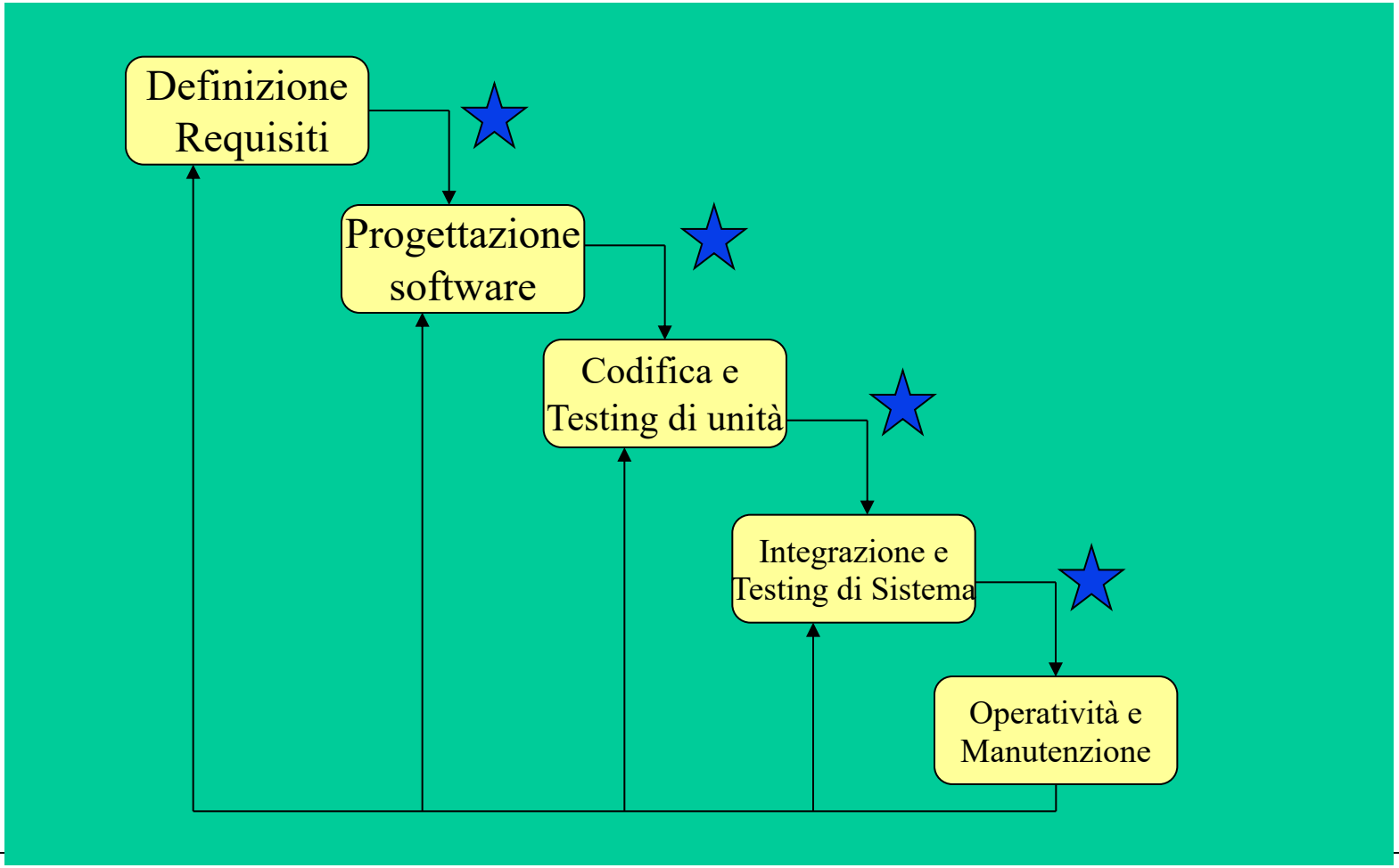
Manutenzione del software

Generalità
Leggi dell'evoluzione del software
Classi di manutenzione
Modelli di processo per la manutenzione
Tecniche per ringiovanire il software
Legacy systems

Generalità

- La manutenzione del software è il processo di modifica di un prodotto software dopo il suo rilascio in esercizio
  - per eliminare malfunzionamenti
  - migliorare prestazioni o altri attributi di qualità
  - adattarlo a modifiche dell'ambiente operativo
- Il termine manutenzione del software (o evoluzione del software) include spesso anche
  - estensioni delle funzionalità originarie per soddisfare nuovi bisogni degli utenti
- 70 % del budget software è speso in manutenzione
- 75% del personale software svolge attività di manutenzione

La Manutenzione nel Waterfall model



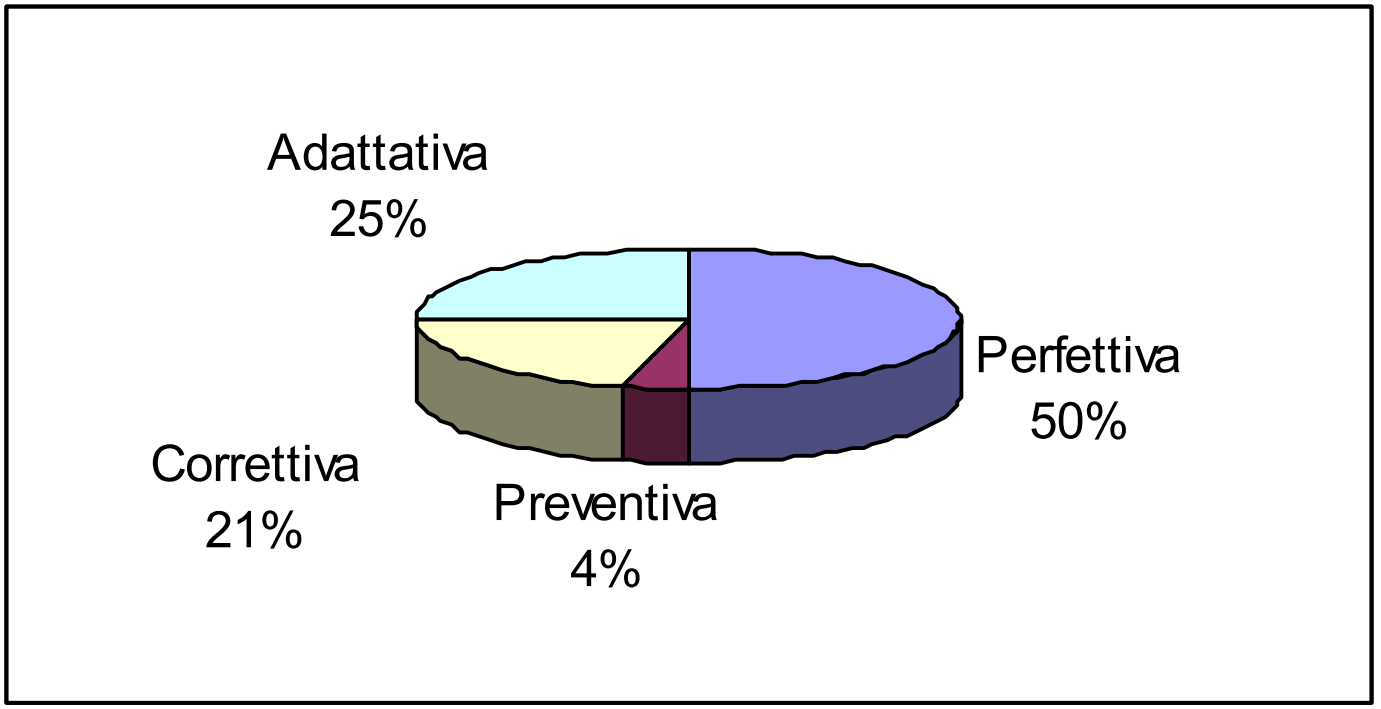
Leggi dell'evoluzione del software

- Basate sull'osservazione di un S.O. IBM su 4 cicli di versioni maggiori
- Lehman e Belady, 1976
- **I legge: Cambiamento continuo**
  - un programma utilizzato in un ambiente del mondo reale deve cambiare, oppure diventa progressivamente meno utile in quell'ambiente
- **II legge: Entropia crescente**
  - Man mano che un programma cambia, la sua struttura degrada e la dimensione cresce, con il risultato di una complessità crescente
  - Risorse addizionali sono richieste per preservarne la semantica e semplificarne la struttura

Classi di manutenzione

- Manutenzione correttiva
  - Modifiche per correggere difetti
- Manutenzione adattativa
  - Modifiche per adattare il software a cambiamenti dell'ambiente operativo (hardware, software di base, interfacce, organizzazione, legislazione, ecc.)
- Manutenzione perfettiva
  - Estensione dei requisiti funzionali, o migliorie di requisiti non funzionali in risposta a richieste dell'utente
- Manutenzione preventiva
  - Modifiche che rendono più semplici le correzioni, gli adattamenti e le migliorie

Distribuzione dello sforzo di manutenzione



Problemi della manutenzione

- In gran parte dipendono dalla mancanza di controllo e disciplina nelle fasi di analisi e progetto del CVS
- Alcuni fattori tecnici:
  - difficoltà nel comprendere un programma scritto da altri
  - mancanza di *documentazione* completa/ consistente
  - software non progettato per modifiche future
  - difficoltà nel tradurre una richiesta di modifica di funzionamento del sistema in una modifica del software
  - valutazione dell'impatto di ciascuna modifica sull'intero sistema
  - la *gestione della configurazione* del software
  - la necessità di ritestare il sistema dopo le modifiche

Modelli di processo per la manutenzione del software

Modello di riparazione veloce (**Quick-fix model**)

- modifiche al codice in termini di patches (‘pezze’)
- veloce ed economico sul breve termine
- degradazione delle struttura
- documentazione modificata a posteriori

Modello di miglioramento iterativo (**iterative-enhancement model**)

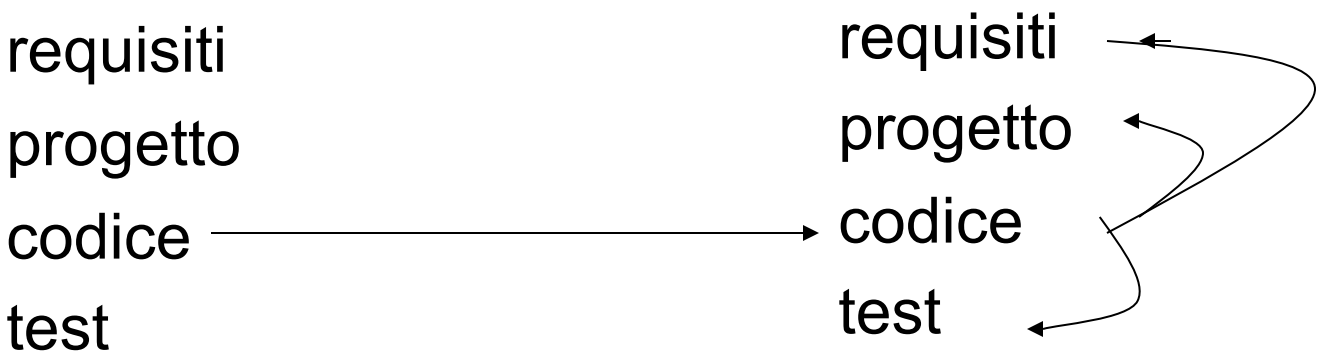
- valutazione preventiva dell’impatto della modifica
- decisione se lavorare su componenti esistenti o sviluppare nuove componenti
- preserva la struttura
- lento e costoso sul breve termine
- documentazione modificata in anticipo

Modello di riparazione veloce (quick- fix model)

Vecchio sistema

requisiti
progetto
codice
test

Nuovo sistema



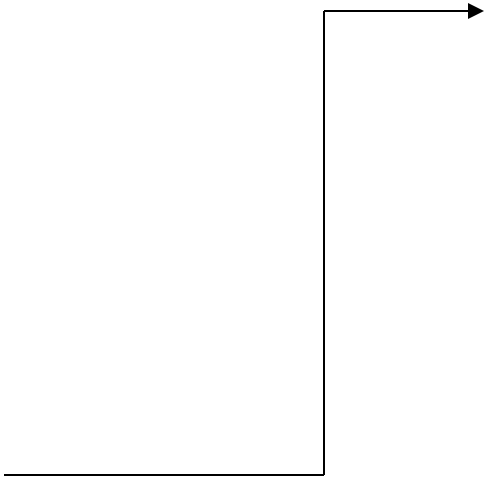
Prima modifico il codice e poi (se ho tempo…) aggiorno la documentazione…

Modello di miglioramento iterativo (iterative-enhancement)

Vecchio sistema

requisiti
progetto
codice
test

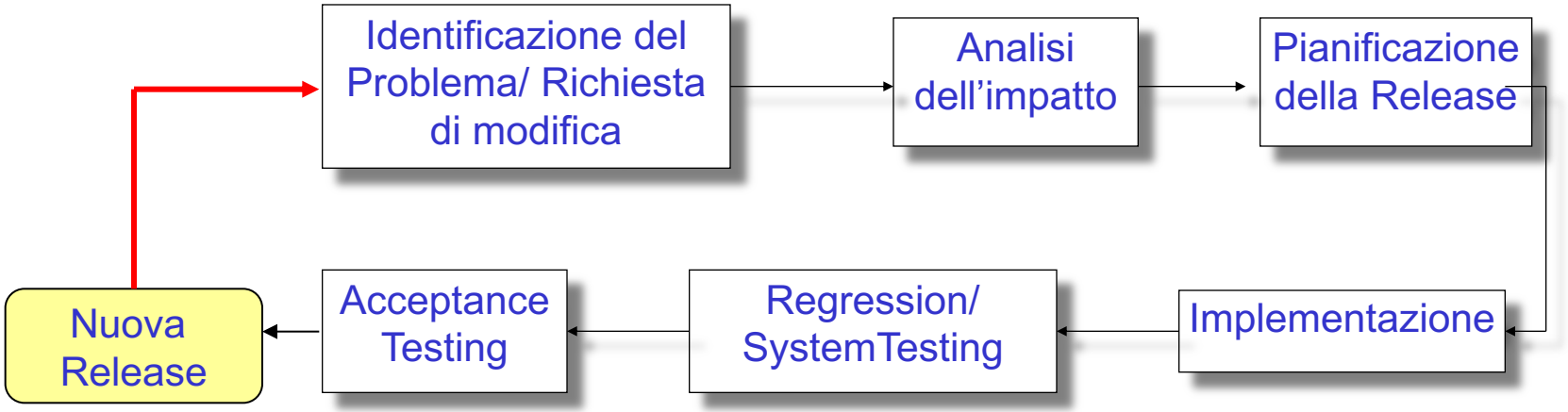
Analizza
Richiesta di modifica



Nuovo sistema

requisiti
progetto
codice
test

Il processo di manutenzione secondo lo Standard IEEE 1219, 1993



Problemi della manutenzione d'urgenza

- In molti casi si ricorre comunque alla manutenzione "d'urgenza"…
  - Se un difetto serio deve essere riparato;
  - Se modifiche dell'ambiente causano effetti collaterali imprevisti;
  - Se è necessario l'adeguamento urgente a seguito di un cambiamento delle regole di business
- In questo caso si finisce con l'anticipare il più possibile l' implementazione del cambiamento sul codice
  - Con tale approccio, la qualità complessiva del software si ridurrà.
  - Utile pianificare interventi di aggiornamento della documentazione e/o di miglioramento della qualità del software.

Gli interventi per ‘Ringiovanire’ il software

- Sono finalizzati a migliorare la manutenibilità di un software ormai deteriorato dagli interventi di manutenzione subiti.
- Diversi tipi di intervento possibili:

- Ridocumentazione
- Restructuring (o Refactoring)
- Reengineering
- Reverse Engineering

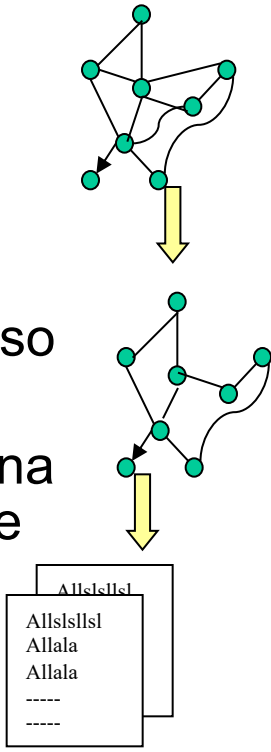
Ridocumentazione

- Consiste in una **analisi statica** del codice sorgente (attraverso appositi strumenti) al fine di produrre documentazione del sistema. Si analizzano:
  - usi delle variabili, chiamate fra componenti, path del flusso di controllo, dimensioni dei componenti, parametri di chiamate, .. Per capire cosa fa il codice e come lo fa.

- Gli output di una attività di ridocumentazione possono essere:
  - grafi delle chiamate,tabelle delle interfacce delle funzioni, dizionari dati, diagrammi del data-flow o control-flow, pseudo-codice, cross-reference fra componenti o variabili
  - Tali output si possono usare per verificare se il software ha bisogno di ristrutturazione.

Restructuring

- Attività che trasforma il codice esistente in codice equivalente dal punto di vista funzionale, ma migliorato dal punto di vista della sua qualità.
- In genere si esegue in tre passi:
  1. Analisi statica del codice per ottenerne una rappresentazione interna (es. call graph, control-flow graph…)
  2. Semplificazione della rappresentazione interna attraverso tecniche di trasformazione automatiche.
  3. La nuova rappresentazione viene usata per generare una versione strutturata (migliorata) ed equivalente al codice originario.



Refactoring

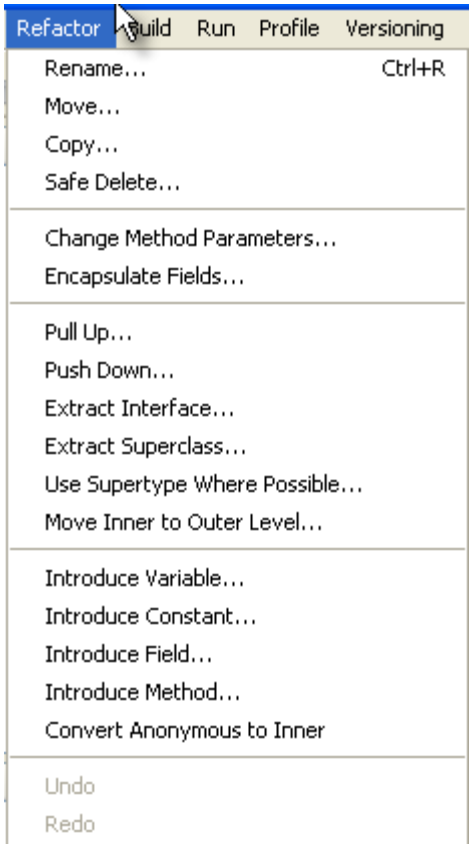
- Anche i sistemi object-oriented sono soggetti al deteriorarmento e diventano legacy!
- Il refactoring è un insieme di tecniche usabili per migliorare il codice ed il design di sistemi object-oriented.
- Martin Fowler è autore del libro “Refactoring: Improving the Design of Existing Code” (1999)dove presenta più di 70 pattern per eseguire il refactoring.
- Tali tecniche cercano di eliminare i cosiddetti *Bad Smells* dal codice.

Esempi di Bad Smells nel codice

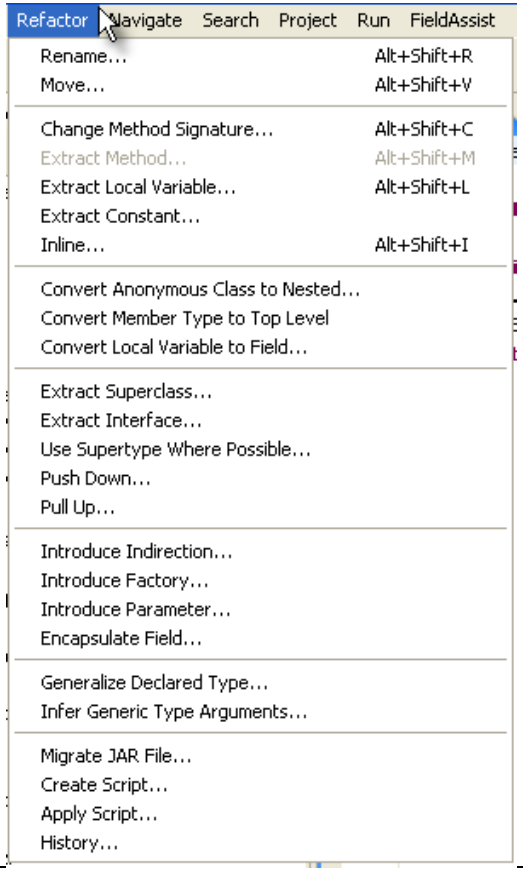
Problema (Bad Smell)	**Pattern di Refactoring applicabile**
Codice duplicato	*Extract Method*
Metodi troppo lunghi	*Extract Method*
Classi troppo grandi	*Extract Class, Extract Sub-Class, Extract Interface*
Lunghe liste di parametri di metodi	*Replace parameter with Method*
Cambiamenti divergenti in una classe (una classe è soggetta a cambiamenti per tanti motivi)	*Extract Class*
…	…

Esempi di Tool per il Refactoring

Net Beans Refactoring



Eclipse Refactoring



Reengineering

- It is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form [*Chikofsky and Cross*]

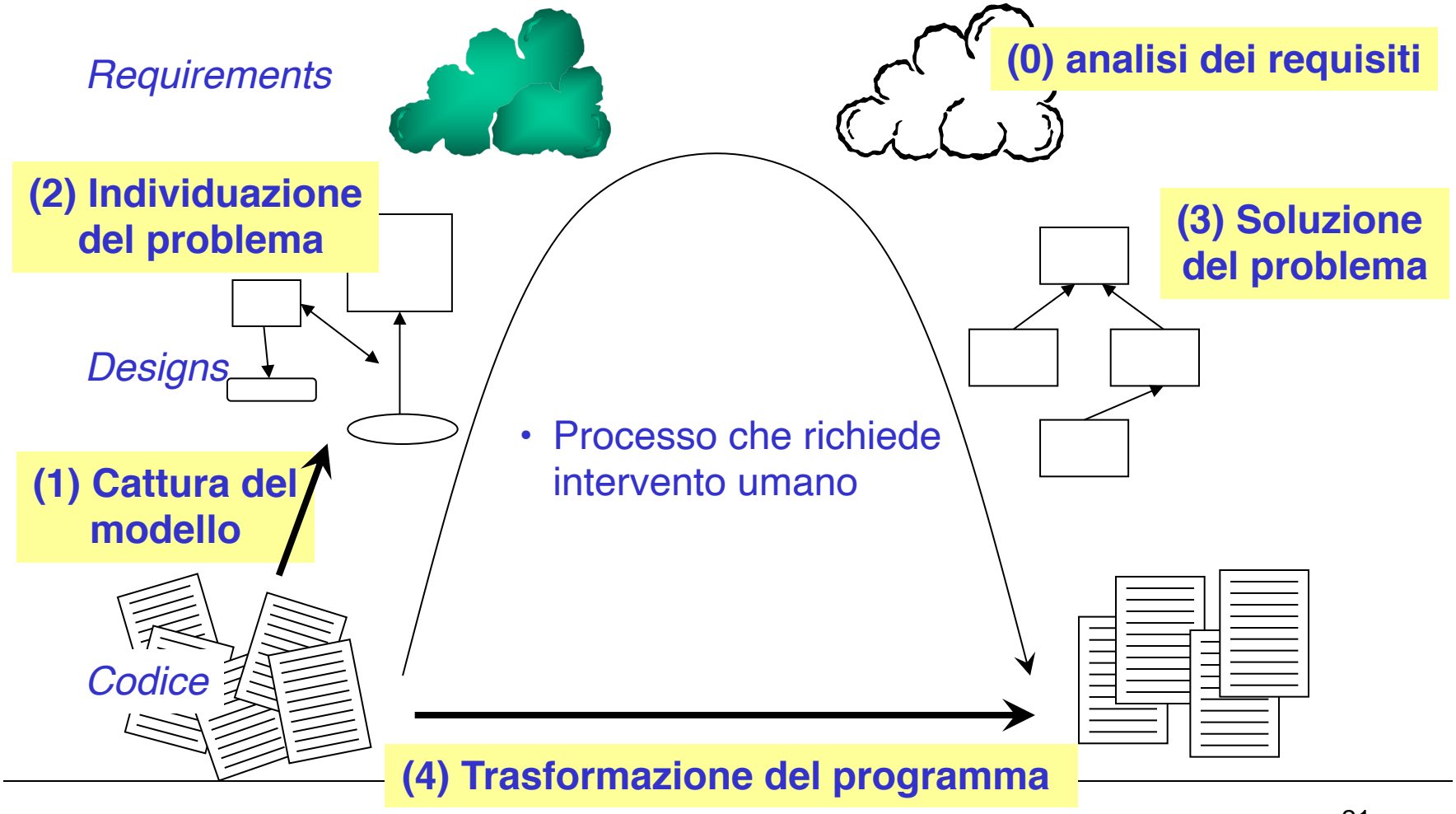
- La reingegnerizzazione (reengineering) è un'attività di re-implementazione di un sistema software svolta per migliorare la manutenibilità di un software esistente.

- Essa può comprendere:
  - Ridocumentazione, Ristrutturazione e riscrittura di parte del software (o anche di tutto) senza modificare l'insieme di funzionalità che esso realizza

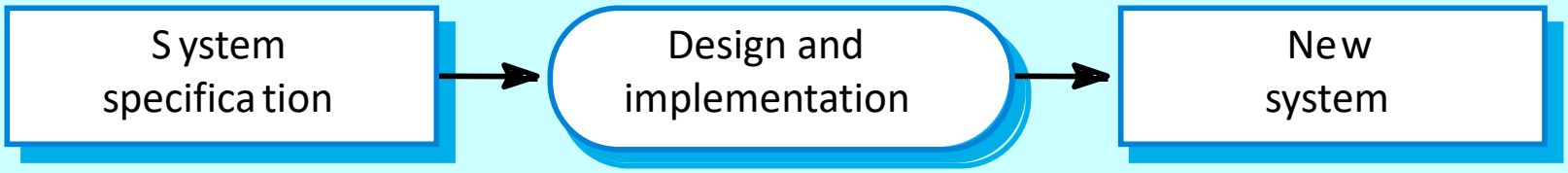
Obiettivi del Reengineering

- *Modularizzazione del sistema*
  - Suddivisione di un sistema monolitico in parti da riusare separatamente
- *Miglioramento delle Performance*
  - Migliorare le prestazioni di un sistema esistente
- *Migrazione (o Porting) verso altre Piattaforme*
  - Necessità di localizzare I componenti dipendenti dalla piattaforma
- *Estrazione del progetto*
  - Per migliorare maintainability, portability, etc.
- Usare una *nuova Tecnologia*
  - quali nuove caratteristiche di un linguaggio, standards, librerie, etc.

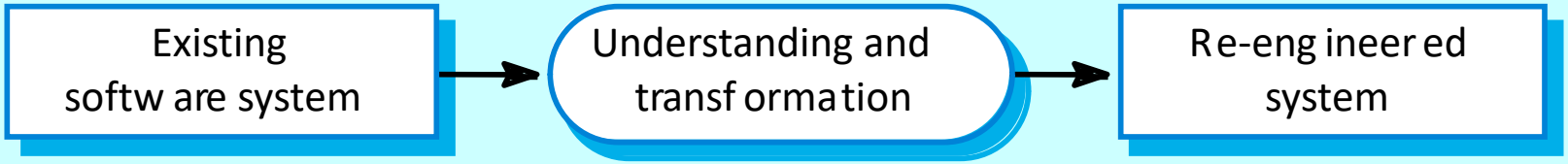
Il ciclo di vita del Reengineering



Forward engineering e reengineering



Forward engineering



Software re-engineering

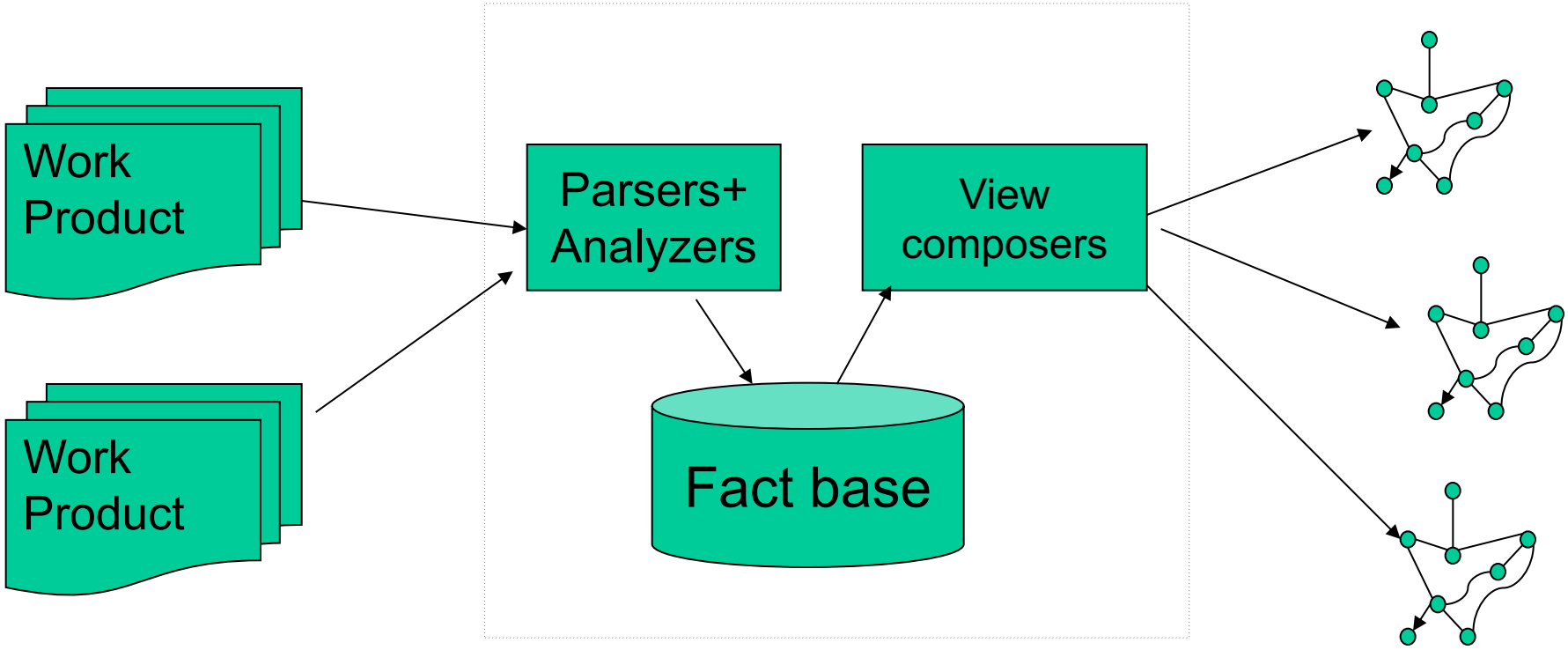
Reverse Engineering

- È un'attività che consente di ottenere specifiche e informazioni sul design di un sistema a partire dal suo codice, attraverso processi di estrazione ed astrazione di informazioni.

- La definizione di **Chikofsky and Cross**, 1990 [1]:
- *Analyzing a subject system*
  - *to identify the system's components and their inter-relationships and*
  - *to create representations of the system in another form or at a higher level of abstraction*
- *A two-steps process*
  - *information extraction*
  - *view abstraction*

[1] **Chikofsky and Cross,** Reverse engineering and design recovery: A taxonomy. *IEEE Software*,1990

What is reverse engineering



Fasi di Estrazione ed Astrazione

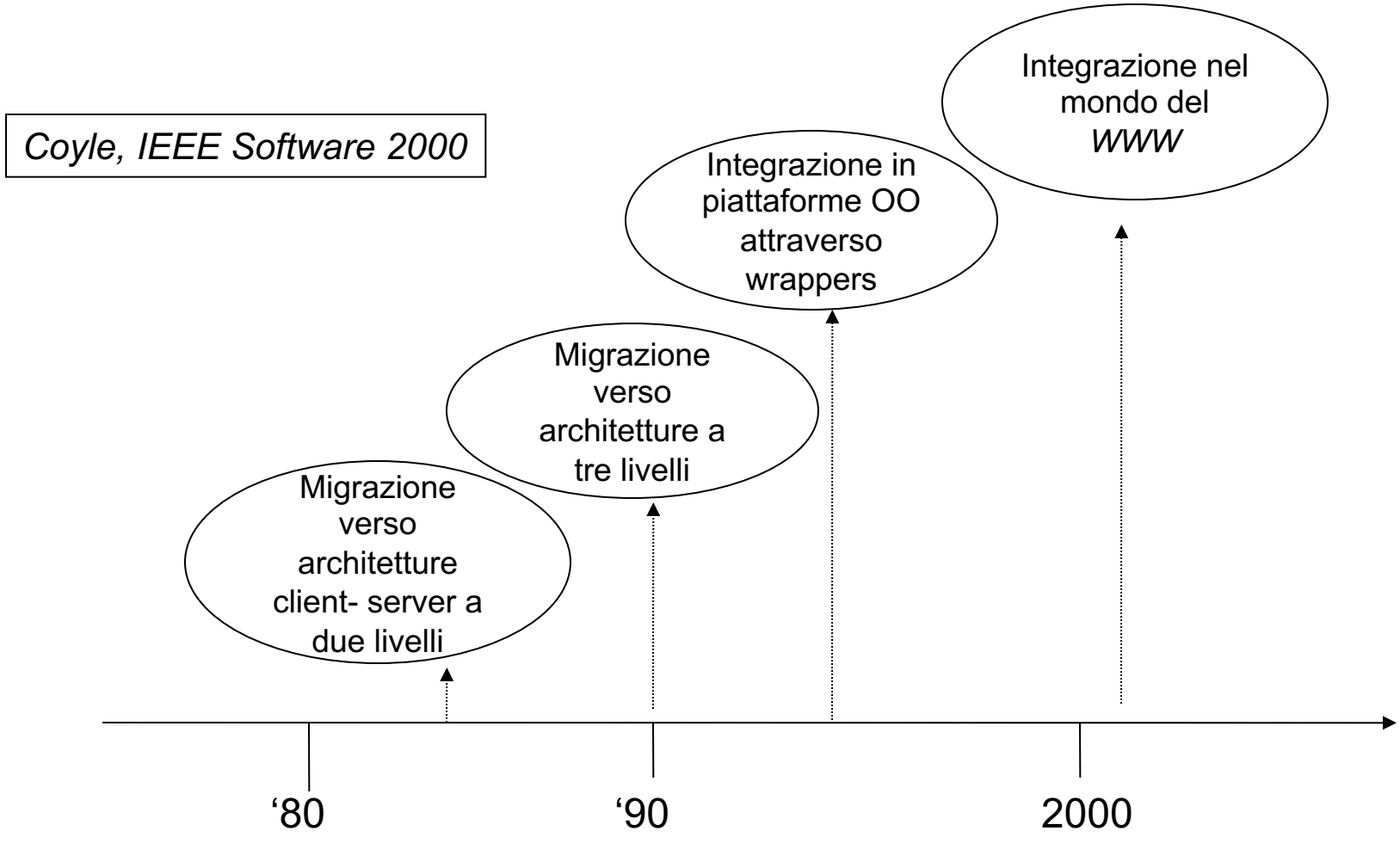
- Estrazione
  - Analisi del codice o di altri artifatti software, allo scopo di ottenere informazioni relative al sistema analizzato.
  - Particolarmente utili sono quelli strumenti in grado di estrarre informazioni da un codice sorgente qualsiasi, nota che sia la grammatica del linguaggio di programmazione (ad esempio JavaCC)
- Astrazione
  - Si esaminano le informazioni estratte e si cercano di astrarre diagrammi, o viste, ad un più alto livello di astrazione (es.: diagrammi di progetto, architetturali, del dominio dei dati)
  - I processi di astrazione non sono completamente automatizzabili poichè necessitano di conoscenza ed esperienza umana

Il problema dei Legacy Systems

- Un sistema legacy (“ereditato”) é spesso vecchio (10 anni o più di vita);
- É di grandi dimensioni (centinania di migliaia di linee di codice)
- É scritto in assembly o in un linguaggio di vecchia generazione
- É stato probabilmente sviluppato prima che si diffondessero I moderni principi dell’ingegneria del software
- La manutenzione è stata svolta in modo da seguire le modifiche nei requisiti, aumentando così l’entropia (il disordine) del sistema
- La manutenzione risulta ormai difficile e costosa
- Realizza funzionalità cruciali e irrinunciabili per l’organizzazione
- Contiene anni di esperienza accumulata nell’ambito del dominio specifico del problema

Esempio di Sistema Legacy…

La gestione dei Sistemi Legacy: due decenni di strategie



Il problema

Un sistema legacy obbliga il management a cercare soluzioni e strategie di manutenzione vincenti!

Svariate alternative disponibili:

- Eliminazione del sistema e sviluppo di un nuovo sistema
- Recupero dei componenti più preziosi del sistema, sostituendo i restanti con prodotti preconfezionati (COTS)
- Eliminazione dei componenti ormai inutili, del codice obsoleto e dei dati “morti”
- Congelamento del sistema *as is*, e riutilizzo mediante tecniche di wrapping
- Manutenzione Ordinaria
- Manutenzione Preventiva (re-documentation, restructuring o reengineering)
- Migrazione …

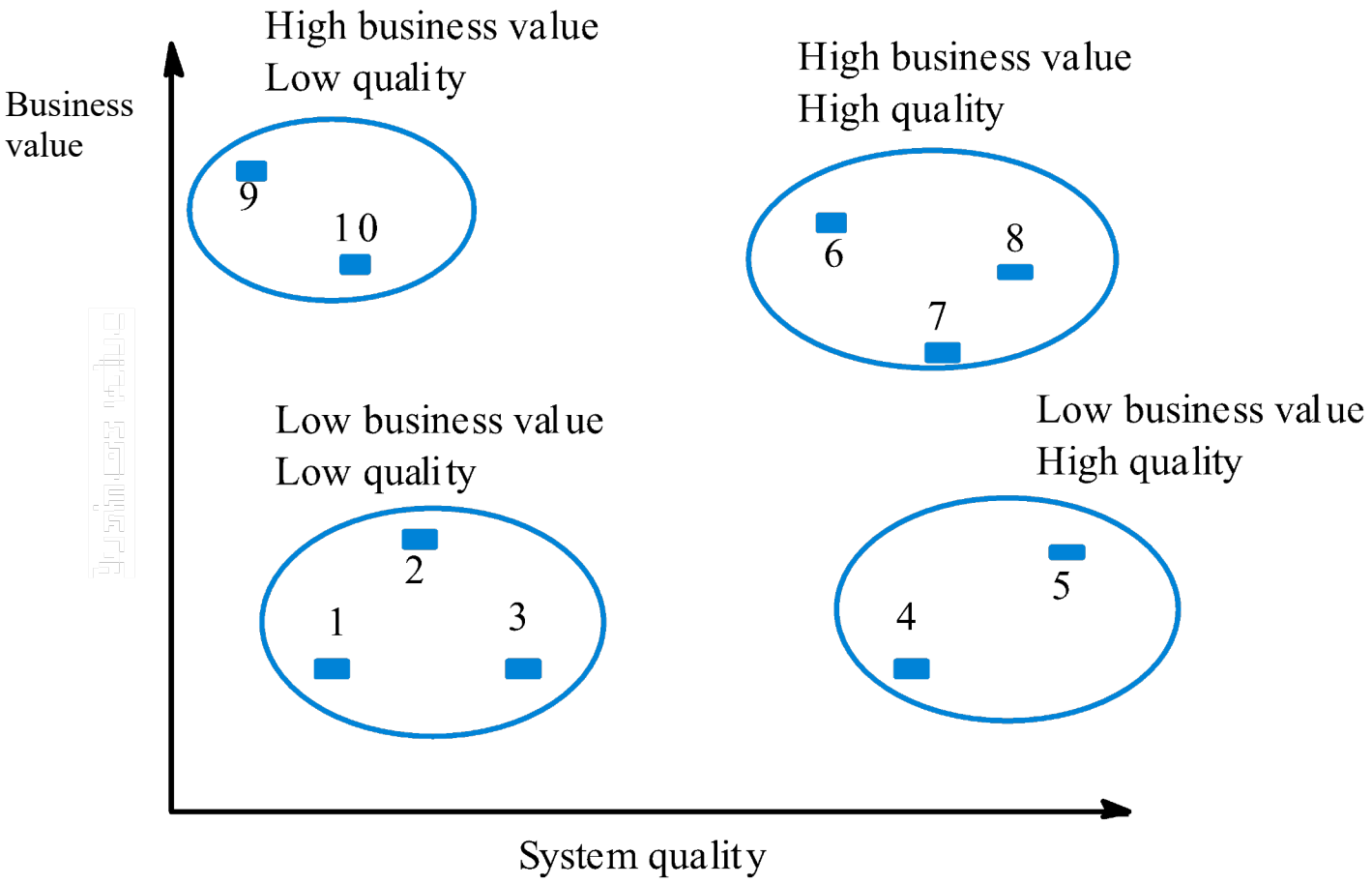
Come decidere?

- Occorre valutare il sistema sia da una prospettiva Economica che Tecnica.

- Prospettiva Economica (Business Value):
  - L'azienda ha realmente bisogno del sistema?

- Prospettiva Tecnica:
  - Qual è la qualità del software applicativo, e del suo ambiente di utilizzo (sia software che hardware)?

System quality and business value Analysis



Legacy system categories

1. **Low quality, low business value**
   - Dovrebbe essere abbandonato
2. **Low-quality, high-business value**
   - Realizza funzionalità importanti ma è costoso mantenerlo. Dovrebbe essere reingegnerizzato in modo da rendere le future (necessarie) operazioni di manutenzione più agevoli ed efficaci (cioè finire nel quarto caso)
3. **High-quality, low-business value**
   - Si può decidere sia di abbandonarlo (in quanto poco importante), sia di rimpiazzarlo con COTS (se realizza qualcosa di generale, indipendente dal dominio specifico) oppure mantenerlo (dato che i costi di manutenzione saranno limitati)
4. **High-quality, high business value**
   - Su di esso si eseguono le operazioni di manutenzione
     - In questo modo, però, la qualità diventerà via via più bassa, fino a finire nel secondo caso

Valutazione del valore economico

- L'Assessment del Valore di Business dovrebbe prendere in considerazione diversi punti di vista *per valutare* quanto è importante Il processo aziendale per il raggiungimento degli obiettivi aziendali.
  - Utenti del sistema, clienti, Line managers, IT managers, Senior managers.
- É necessario condurre interviste e raccogliere I risultati su:
  - Utilizzo del sistema
  - Processi aziendali supportati
  - Fidatezza del sistema
  - Gli output prodotti

Valutazione della qualità tecnica

- Richiede due tipi diversi di assessment:

- *Assessment dell'ambiente (environment)*
  - Quanto è efficace l'ambiente operativo del sistema e quanto costa mantenerlo ?
- *Application assessment*
  - Qual è la qualità dell'applicazione software?

Fattori per la Valutazione dell'ambiente

Factor	Questions
Supplier stability	Is the supplier is still in existence? Is the supplier financially stable and likely to continue in existence? If the supplier is no longer in business, does someone else maintain the systems?
Failure rate	Does the hardware have a high rate of reported failures? Does the support software crash and force system restarts?
Age	How old is the hardware and software? The older the hardware and support software, the more obsolete it will be. It may still function correctly but there could be significant economic and business benefits to moving to more modern systems.
Performance	Is the performance of the system adequate? Do performance problems have a significant effect on system users?

Fattori per la Valutazione dell'ambiente

Support requirements	What local support is required by the hardware and software? If there are high costs associated with this support, it may be worth considering system replacement.
Maintenance costs	What are the costs of hardware maintenance and support software licences? Older hardware may have higher maintenance costs than modern systems. Support software may have high annual licensing costs.
Interoperability	Are there problems interfacing the system to other systems? Can compilers etc. be used with current versions of the operating system? Is hardware emulation required?

Fattori per la valutazione della qualità tecnica

Factor	Questions
Understandability	How difficult is it to understand the source code of the current system? How complex are the control structures that are used? Do variables have meaningful names that reflect their function?
Documentation	What system documentation is available? Is the documentation complete, consistent and up-to-date?
Data	Is there an explicit data model for the system? To what extent is data duplicated in different files? Is the data used by the system up-to-date and consistent?
Performance	Is the performance of the application adequate? Do performance problems have a significant effect on system users?

Un esempio di modello di Qualità

Caratteristica	Attributo	Metriche
Business Value	Economic value	Market value, Profitability Index, Internal Rate of Return
	Data Value	# mission critical archives, #application-dependent archives
	Utility	Business function coverage rate, actual usage frequency, customer satisfaction metrics
	Specialization	# highly specialized functions, #generic functions
Technical Value	Maintainability	LOC, FP, Control Flow Knots, Cyclomatic Complexity, Dead code rate
	Decomposability	Architecture modularity, # modules with separation of concerns
	Deterioration	Backlog increase, defect rate increase, response-time increase, maintenance time per request increase
	Obsolescence	System age, operating system version, hardware version, technical support
		availability