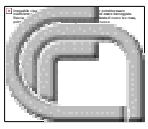
Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Strategie e tecniche innovative per l'interfacciamento di prodotti Legacy

P. Storniolo

Rapporto Tecnico N.:
RT-ICAR-PA-12-06 **Dicembre 2012**



Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Strategie e tecniche innovative per l'interfacciamento di prodotti Legacy

P. Storniolo[1]

Rapporto Tecnico N.: **RT-ICAR-PA-12-06**

Data: **Dicembre 2012**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Viale delle Scienze edificio 11, 90128 Palermo.

Sommario

Introduzione

Nell’ambito dello studio e dell’implementazione di un prototipo di soluzione documentale digitale di nuova generazione che possa superare gli attuali limiti operativi delle soluzioni disponibili sul mercato, si ritiene fondamentale prendere in considerazione i sistemi già presenti in ambito aziendale.

Nel corso degli anni, gli ambienti IT sono diventati più complessi e più eterogenei grazie alla diverse esigenze dei clienti e alla rapida innovazione nel settore. Molte aziende richiedono l'integrazione con i sistemi presenti per mantenere in attività i processi operativi. Per risolvere questo problema, molti produttori di software offrono la possibilità di interoperabilità in fase di progettazione di nuovi sistemi.

L'approccio generalmente utilizzato per l'interoperabilità aumenta il valore delle soluzioni IT aiutando i clienti a integrare i dati presenti in azienda e la logica di business in nuove applicazioni o processi di business senza la necessità di "eliminare e sostituire" il codice o effettuare costose riscritture delle applicazioni.

La maggior parte delle aziende informatizzate hanno un ambiente con molti sistemi legacy, applicazioni, processi e fonti di dati. Il mantenimento dei sistemi legacy è una delle difficili sfide che le moderne le imprese si trovano ad affrontare oggi.

Il mercato offre una varietà di soluzioni a questo problema sempre più comune di modernizzazione dei sistemi legacy. Tuttavia, la comprensione del punti di forza e di debolezza di ciascuna tecnica di modernizzazione è fondamentale per selezionare la soluzione corretta e per il successo globale dello sforzo di rinnovamento tecnologico. Questo lavoro vuole fornire indicazioni utili al fine di poter indicare metodi per l’integrazione dei sistemi legacy nelle nuove applicazioni.

Sistemi Legacy

Definizione

Secondo il *FOLDOC (Free On-Line Dictionary Of Computing)* [(1)] si definisce Legacy System "*A computer system or application program which continues to be used because of the cost of replacing or redesigning it and often despite its poor competitiveness and compatibility with modern equivalents*" (Un sistema di calcolo o un programma applicativo che continua ad essere utilizzato a causa del costo di sostituzione o di riprogettazione e spesso nonostante la sua scarsa competitività e compatibilità con equivalenti moderni).

Le caratteristiche peculiari che possono riscontrarsi in un Sistema Legacy sono:

- È un sistema fondamentale per l'operatività dell'organizzazione; inoltre spesso è fortemente utilizzato (migliaia di transazioni al giorno) essendo *mission-critical* e dovendo quindi rimanere operativo 24 ore su 24;
- L'organizzazione ha molto investito nel corso degli anni, e quindi non può essere semplicemente accantonato;
- È generalmente composto da centinaia o addirittura milioni di linee di codice distribuite su molti programmi;
- Il nucleo principale spesso è stato progettato e realizzato con tecniche e tecnologie vecchie di parecchi anni;
- Utilizza, quando presente, un DBMS obsoleto spesso non relazionale;
- È scritto con linguaggi di vecchia generazione (COBOL, PL1, Assembler …);
- L'interfaccia utente è a caratteri;
- Le varie parti che lo compongono sono prevalentemente *stand-alone*, ciascuna progettata e realizzata indipendentemente dal resto, definendo un'integrazione prevalentemente "verticale";
- Generalmente manca di adeguata documentazione e, ove presente, risulta difficile da comprendere poiché spesso non aggiornata con le varie inevitabili modifiche apportate nel tempo;
- Implementa anche procedure non formalizzate dell'organizzazione.

Non tutte le caratteristiche indicate sono presenti solo su applicazioni e sistemi sviluppati negli scorsi decenni. In alcune situazioni si reputano sistemi legacy anche

implementazioni recenti che però non seguono i canoni e le tecniche di progettazione e di sviluppo proprie dell'ingegneria del software. In alcuni ambienti si può giungere ad affermazioni "estreme" come "ogni applicazione non Java è legacy". Chiaramente non è sufficiente l'utilizzo di un particolare linguaggio di programmazione (Java) o di un particolare paradigma (OO, SOA …) per poter affermare che un sistema possa ritenersi non legacy.

Problematiche

Le maggiori criticità dei sistemi legacy non sono relative al loro funzionamento corrente, ma spesso legate alle difficoltà che si incontrano qualora si debba intervenire per un loro adeguamento a nuove e mutate esigenze dovute ai cambiamenti dell'organizzazione o al rilascio di nuove funzionalità orientate all'integrazione con nuovi servizi.

Nella maggior parte dei casi non utilizzano una interfaccia grafica e non sono progettati per lavorare attraverso architetture di rete che utilizzano il protocollo TCP/IP.

Non sono pensati per essere integrati con altri applicativi aziendali, quindi difficilmente posso essere utilizzati in cooperazione.

Classificazione

La maggior parte dei sistemi legacy è sviluppata in ambiente mainframe; la principale distinzione fra essi, soprattutto nell'ottica di un possibile trattamento, si basa sulle modalità con cui sono organizzati i servizi di presentazione, di logica applicativa e di accesso ai dati.

In genere è possibile distinguere tali sistemi come:

1. **Altamente Decomponibili**: sono strutturati e decomposti nelle tre parti
   - Accesso ai dati
   - Logica Applicativa
   - Interfaccia Utente

   Risultano agevolmente trattabili grazie alla modularità della loro struttura.

2. **Data Decomponibili**: solo la parte di accesso ai dati è separata, la logica applicativa e l'interfaccia utente sono fuse e difficilmente distinguibili.
   La manutenzione su tali sistemi è realizzabile riutilizzando, ove possibile, la componente di accesso ai dati.
3. **Program Decomponibili**: la parte di accesso ai dati e la logica applicativa sono un blocco monolitico, è separata l'interfaccia utente. Anche in questo caso, sotto opportune condizioni la gestione risulta affrontabile. In questa categoria generalmente ricadono la maggior parte dei sistemi legacy.
4. **Monolitici**: è il caso peggiore. Non vi è separazione tra le componenti del sistema. Il sistema è chiuso. La gestione risulta complessa e generalmente si utilizza un approccio di tipo "black-box".

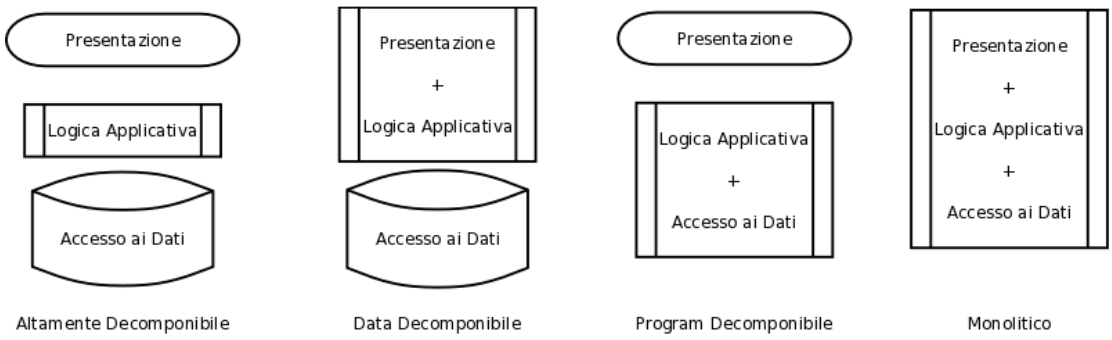


Fig.1 - Le differenti tipologie di Sistemi Legacy

Trattamento

Qualunque sistema, sia esso legacy o no, sia esso su mainframe o su hardware midrange (PC, workstation, server) viene sottoposto, nel corso del suo ciclo di vita, ad una serie di attività di valutazione, manutenzione e trattamento che ne comportano una generale evoluzione.

La sola attività operativa associata al cambiamento e all'evoluzione di un sistema, fino a qualche tempo fa, era la manutenzione. Recentemente ha acquisito notevole importanza quello che viene ormai comunemente definito il trattamento. Tale attività può richiedere una approfondita analisi e comprensione del sistema, affrontata con tecniche di software reengineering con approccio tipo white-box, o limitata allo studio del solo sotto-sistema di presentazione, operando quindi sempre con tecniche di software reengineering ma di tipo black-box.

L’approccio white-box coinvolge necessariamente tecniche di reverse engineering che consentano una comprensione approfondita delle componenti del sistema e dovrà poi attuare le necessarie riconversioni.
L’approccio black-box, viceversa, utilizza l’analisi delle interfacce esterne e attua in definitiva una politica di incapsulamento (wrapping) del sistema originario.

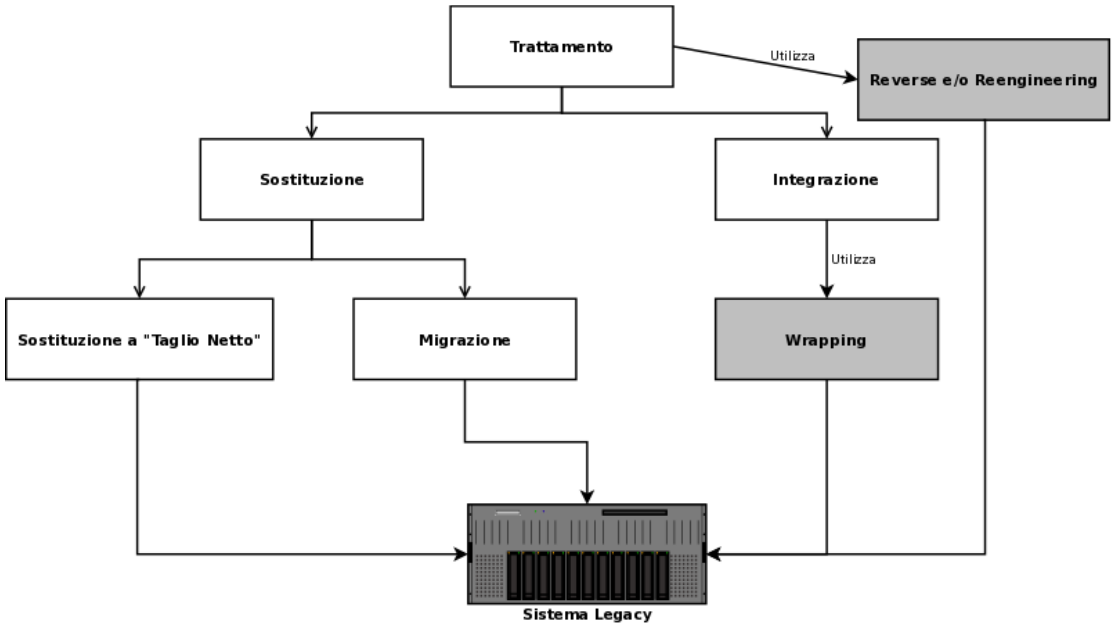


Fig.2 - Possibili Trattamenti di un Sistema Legacy

Manutenzione

Malgrado il miglioramento dei processi produttivi del software ed in generale dei sistemi informativi, la manutenzione, elemento strategico, rappresenta oltre il 65-70% della spesa per i sistemi informatici. Se si considera, inoltre, la possibilità di effettuare manutenzione non strutturata e a volte caotica, ciò porta ad un rapido degrado di un sistema che lo rende rapidamente legacy.
La manutenzione comporta una serie di problematiche legate al fatto che spesso è estremamente complicato, se non addirittura impossibile, comprendere il software realizzato da altri. Inoltre, generalmente l’autore è restio a collaborare alla risoluzione dei problemi. Nel caso, poi, di sistemi molto complessi e datati, la documentazione è spesso inesistente o fortemente inadeguata. Se a questo si aggiunge che i criteri di progettazione del software (quando applicati) non tengono conto della possibile modificabilità, ci si rende conto che affrontare la manutenzione è praticamente impossibile (spesso anche agli stessi autori del software!).

In letteratura si distinguono vari tipi di manutenzione:

a) **Migliorativa**: interviene su gli elementi che servono a soddisfare i requisiti non funzionali del sistema;
b) **Adeguativa**: utilizzata per attualizzare il sistema a mutate esigenze sia a livello infrastrutturale che tecnologico;
c) **Correttiva**: utilizzata per rimuovere difetti del sistema (bug);
d) **Evolutiva**: migliora il sistema soprattutto dal punto di vista dei requisiti funzionali.

In alcune circostanze si parla anche di manutenzione "Preventiva", indicando con tale termine una serie di interventi periodici o scaturiti da particolari circostanze, atti ad evitare possibili malfunzionamenti del sistema.

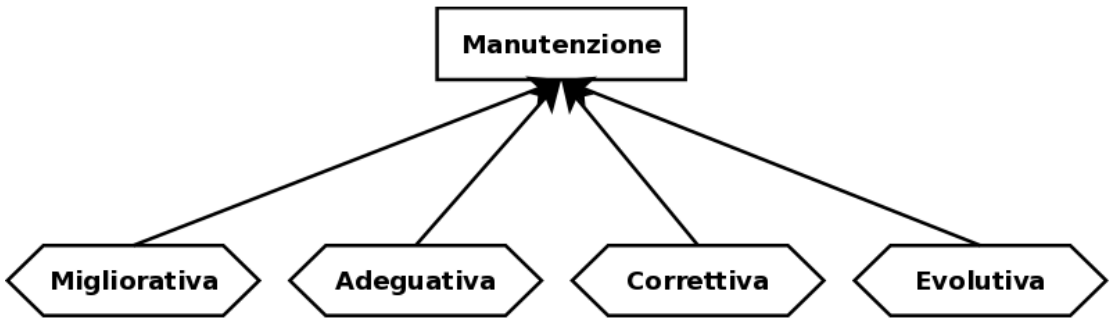


Fig.3 - Tipologie di Manutenzione

Reverse Engineering

Per molti anni l'ingegneria del software ha investito soprattutto nella ricerca di nuovi e più efficienti processi di sviluppo del software, per aumentare la qualità delle applicazioni e per ridurre significativamente il cosiddetto *time-to-market*. Il ciclo dei prodotti si è ridotto a tal punto (principalmente nel cosiddetto "mercato di massa") che la manutenzione è stata spesso soppiantata dal rilascio di nuove versioni.

Le grandi organizzazioni non possono però sostituire i loro sistemi senza una adeguata analisi. Uno degli obiettivi del reverse engineering è proprio quello di recuperare le informazioni, la conoscenza, tipicamente insita nei sistemi legacy di cui si è persa traccia documentata. Questa azione di recupero risulta fondamentale soprattutto quando sul vecchio sistema devono operare progettisti e sviluppatori che non avevano precedentemente partecipato al suo sviluppo. Attraverso le tecniche

proprie del reverse engineering si possono ottenere adeguate e nuove rappresentazioni del sistema originale.
Attraverso l'applicazione del processo di reverse engineering è anche possibile migliorare sia la manutenzione che il riuso delle componenti, identificando e documentando gli elementi che definisco il sistema.
Bisogna evidenziare che tali le attività comportano investimenti spesso ingenti tali da rendere poco conveniente l'attuazione del processo stesso.

Migrazione

Si parla di migrazione quando si attua il passaggio da un sistema esistente, detto sorgente, a uno nuovo, detto obiettivo. Nello specifico, il sistema di partenza è un sistema legacy ed il secondo è un sistema sviluppato secondo nuovi paradigmi e moderne tecnologie. La migrazione è un processo estremamente rischioso e comporta particolari accorgimenti. Molti progetti di migrazione intrapresi a metà degli anni 90 sono falliti a causa della carenza nella fase di analisi e reverse engineering. Una causa diffusa del fallimento della migrazione è, inoltre, lo scarso supporto da parte del management aziendale, che vede spesso la migrazione come un evento traumatico da evitare.
Si possono perseguire, in genere, tre diversi tipi di migrazione:

1) **Migrazione parziale**: si trasferisce solo la parte del sistema che da maggiori problemi in termini di prestazioni e/o di manutenzione;
2) **Migrazione totale**: tutto il sistema viene trasferito;
3) **Mainframe unplug**: si tratta di sostituire le vecchie piattaforme mainframe con nuove e generalmente non direttamente compatibili piattaforme. Le applicazioni che costituisco il sistema vengono ricompilate nel nuovo ambiente (si parla in questi casi di sliding o rehosting). Questo approccio è ovviamente utile nei casi in cui le componenti del sistema siano ben progettate per cui è vantaggioso mantenerle.

Spesso un considerevole miglioramento per il business dell'organizzazione si può ottenere mediante la modifica dell'interfaccia utente. Generalmente a caratteri (tipica dei vecchi terminali IBM 3270), è possibile renderla grafica o anche accessibile via Web da un browser mediante strumenti middleware (spesso proprietari) con cui si può effettuare la migrazione; si parla in questi casi di screen scraper realizzati sovente

mediante linguaggi di scripting. Ovviamente tali interventi, applicabili anche a sistemi legacy monolitici, soffrono di problemi legati alle prestazioni.
Altro aspetto della migrazione di un sistema legacy è quello del trasferimento dei dati da un formato/piattaforma ad un altro. Tale passaggio comporta la soluzione di alcuni problemi:

- **Conversione**: passaggio da un db non relazionale o da un file system ad un db relazionale;
- **Trasformazione**: creazione di viste o aggregazioni di informazioni;
- **Spostamento**: trasferimento fisico delle informazioni da un supporto ad un altro;
- **Allocazione**: da uno storage centralizzato ad uno di rete (SAN).

Generalmente, affrontare il problema della migrazione dei dati è più semplice rispetto a quello dell'applicazione, ciò comporta quindi minor sforzo sia in termini di investimento che di risorse.
La problematica della migrazione dei dati, comunque, non interessa solo i sistemi legacy, ma si presenta spesso anche in settori quali ad esempio il DataWarehousing.

Tecniche d'Integrazione

Ci sono sostanzialmente due approcci per il riutilizzo di sistemi legacy: reingegnerizzazione e integrazione. Con il termine reingegnerizzazione si intende generalmente la ristrutturazione di un sistema. La ristrutturazione di un sistema legacy richiede che il codice del sistema sia ben documentato e/o che possa essere analizzato e trasformato da un processo automatico [(2)]. La ristrutturazione di un sistema è lenta. L'integrazione è più veloce e, in definitiva, più economica della reingegnerizzazione. Per integrare un sistema legacy, si deve definire il ruolo di ogni sottosistema, definire le interfacce per ogni sottosistema, e costruire un oggetto wrapper per ciascun sottosistema.
Una strategia di integrazione può essere intrusiva o non intrusiva. Una strategia di integrazione che richiede la conoscenza dei meccanismi interni di un sistema legacy si chiama integrazione intrusiva (white-box), mentre una strategia di integrazione che richiede la sola conoscenza esterna delle interfacce di un sistema legacy si chiama non-intrusiva (black-box). Ci sono due approcci principali per l'integrazione di sistemi legacy: integrazione delle applicazioni e integrazione dei dati.

La filosofia alla base dell'integrazione delle applicazioni si fonda sulla considerazione che esse contengano la logica di business dell'impresa. La soluzione consiste nel mantenimento di tale logica di business operando un'estensione delle interfacce dell'applicazione per consentire l'interazione con gli altri elementi del sistema.

Un rilevante intervento di integrazione è quello che comporta la modernizzazione dell'interfaccia utente (User Interface - UI) che è la parte più visibile di un sistema. Modernizzare l'interfaccia utente migliora l'usabilità del sistema e ciò è generalmente molto apprezzato dagli utenti finali. Una comune tecnica per l'ammodernamento dell'interfaccia utente è quella dello screen scraping, che consiste nel wrapping delle vecchie interfacce, basati su testo con nuove interfacce grafiche [(3)].

Altro approccio, sempre nell'ambito dell'integrazione delle applicazioni, è quello usato nella cosiddetta integrazione point-to-point. Tale intervento si basa sulla generazione di opportuni canali di comunicazione sviluppati fra ciascuna coppia di applicazioni. Tale soluzione è costosa, perché il numero di interfacce necessarie cresce esponenzialmente. Con *n* applicazioni, potrebbero essere richieste *n*(n-1)* interfacce, in quanto per ogni applicazione potrebbe essere necessario definire un'interfaccia con ciascun altra. L'impatto dei cambiamenti rispetto le esigenze di comunicazione per l'aggiunta di una nuova applicazione è significativo. La manutenzione chiaramente è un problema legato sostanzialmente al numero di nodi che costituiscono il sistema complessivo.

Si può ridurre tale complessità, riconducendola ad un incremento lineare, attraverso l'utilizzo di middleware message-oriented basato sul Common Object Request Broker Architecture (CORBA - http://www.omg.org/spec/CORBA/Current/) [(4)]. La soluzione richiede l'interfacciamento di ogni applicazione al "bus dei messaggi" tramite un adattatore. Ogni applicazione ha, quindi, una sola interfaccia di programmazione, il bus dei messaggi. Le applicazioni comunicano attraverso l'invio di messaggi al bus, che li inoltra. Un meccanismo di sottoscrizione delle code consente agli abbonati di ricevere solo i messaggi cui sono interessati. Il middleware prodotto può anche fornire servizi a valore aggiunto quali la consegna garantita, la consegna certificata, la messaggistica transazionale ed eventuali messaggi di trasformazione (usando broker).

La gestione delle informazioni di un sistema informativo aziendale è il risultato del funzionamento di diversi database aziendali, di varie applicazioni e sempre più spesso di integrazione e interoperabilità dei sistemi legacy, ottenute generalmente attraverso fusioni e acquisizioni. Questi sistemi legacy producono dati strutturati o semi-

strutturati che aggiungono alle grandi quantità di dati che una società genera ogni giorno. Questi dati devono spesso essere condivisi tra sistemi eterogenei all'interno della stessa azienda e, infine, oltre le sue stesse mura. La trasformazione dei dati comunicati è necessaria per consentire alle aziende di integrare strettamente i loro sistemi in una infrastruttura coerente senza modificare significativamente le loro applicazioni.

Generiche architetture tecnologiche sono state sviluppate per consentire una più agevole integrazione dei sistemi informativi aziendali. In particolare, si possono utilizzare specifiche piattaforme tecnologiche per l'integrazione dei sistemi legacy.

Una piattaforma sempre più adottata è la Java J2EE Connector Architecture. Essa definisce un'architettura standard per il collegamento, mediante l'uso della piattaforma J2EE, di sistemi eterogenei, implementando un insieme di meccanismi scalabili, sicuri e transazionali, detti appunto connettori, che consentono l'integrazione dei sistemi informativi aziendali (SIA).

Altra interessante piattaforma, proposta dal Object Mamagement Group (OGM), è la cosidetta Model Driven Architecture (MDA) [http://www.omg.org/mda/].

MDA rappresenta un nuovo approccio che fornisce alle aziende gli strumenti necessari per integrare molte tecnologie middleware diverse (ad esempio CORBA, EJB, XML, SOAP e .NET). Fornisce, inoltre, un'architettura che assicura la portabilità, l'interoperabilità cross-platform, l'indipendenza dalla piattaforma, la specificità di dominio e un'elevata produttività.

Conclusioni

Ci sono diversi approcci alla modernizzazione del patrimonio aziendale incluso il reengineering (white-box) e l'incapsulamento (black-box). Prima di iniziare qualsiasi sforzo di modernizzazione di sistemi legacy, ogni possibile opzione deve essere valutata dal punto di vista sia economico che strategico. Bisogna soprattutto considerare anche le strategie e gli interventi volti ad assicurare un successo a lungo termine. Gli attuali sistemi sono fonte potenziale di futuri problemi di legacy. Per eliminare i problemi ereditati dal passato nel futuro, i sistemi dovrebbero essere pensati e costruiti seguendo le moderne tecniche ed i principi propri dell'ingegneria del software o, più in generale, dell'ingegneria dei sistemi informativi.

Bibliografia

1. **Howe, Denis.** FOLDOC. [Online] http://www.foldoc.org/.
2. *Program Transformation Systems.* **Steinbrüggen, Ralf and Partsch, Helmuth.** s.l. : ACM Computing Surveys, 1983.
3. *Web-Enabling Legacy Data When Resources Are Tight.* **Carr, David F.** s.l. : Internet World, 1998.
4. **Hoque, Reaz.** *CORBA for Real Programmers.* s.l. : Elsevier Science & Technology, 1999.