CONVENZIONE INTERBANCARIA
PER I PROBLEMI DELL'AUTOMAZIONE
(CIPA)

GRUPPO DI LAVORO

INTEGRAZIONE FRA TECNOLOGIE *WEB-BASED* E APPLICAZIONI TRADIZIONALI

DICEMBRE 2002

In relazione alle indicazioni contenute nel Piano delle attività della CIPA per il periodo 1.1.2002 – 30.6.2003, il gruppo di lavoro "Integrazione fra tecnologie *Web-based* e applicazioni tradizionali" rassegna il proprio rapporto.

La Segreteria della Convenzione desidera ringraziare i componenti del gruppo di lavoro di seguito indicati per la collaborazione prestata e il contributo fornito nello svolgimento delle attività del gruppo:

Stefano	FABRIZI	Banca d'Italia (Coordinatore)
Pier Luigi	POLENTINI	Banca d'Italia
Enrico	EBERSPACHER	ABI
Paolo	POZZA	Unicredit Servizi Informativi
Franco	MANZINI	Credito Emiliano
Giulio	SOMMARIVA	Banca Popolare di Lodi
Liliana	CARENA	SanPaolo – Imi
Daniele	INSARAUTO	SanPaolo – Imi
Roberto	ZANACCHI	Banco di Brescia

IL SEGRETARIO

(A. M. Contessa)

INDICE

Sintesi

Molti sistemi mission critical sono stati "scritti" diversi anni fa, prevalentemente in linguaggio COBOL. Secondo recenti stime, negli Stati Uniti esistono più di 100 miliardi di righe di codice di questo tipo, e sicuramente non sono destinate a scomparire in tempi brevi.

I maggiori analisti di mercato concordano sul fatto che, nel medio periodo, una percentuale significativa delle nuove funzionalità di un'applicazione sarà sviluppata utilizzando ancora il linguaggio COBOL e che quasi tutte le applicazioni "rilasciate in produzione" includeranno funzionalità appartenenti a procedure consolidate (legacy system).

La realtà è che i sistemi legacy, in una forma o in un'altra, forniscono ancora la maggior parte delle funzioni di elaborazione di transazioni. Pertanto, la strategia migliore per adeguare le soluzioni tradizionali alle nuove tecnologie, ivi comprese quelle WEB, è l'integrazione.

Dietro questa semplice assunzione si nasconde però per le aziende una serie di problematiche che devono essere attentamente valutate. In questo studio si è cercato di prospettare gli elementi fondamentali che sono alla base dell'integrazione con l'architettura WEB, fornendo, ove possibile, indicazioni e linee guida.

1. Innanzitutto bisogna tenere conto che un sistema informativo tradizionale non rappresenta qualcosa di obsoleto di cui disfarsi. È un elemento su cui l'azienda fa affidamento poiché spesso è alla base del *core business*. La possibilità di farlo convivere più o meno facilmente con la tecnologia *WEB* dipende dalla sua struttura che riflette le modalità con cui il sistema è stato progettato. Un sistema con completo disaccoppiamento delle componenti di logica applicativa, accesso ai dati e *presentation* ha modalità di integrazione con il mondo *WEB* semplificate rispetto a un sistema monolitico in cui non è possibile alcuna separazione fra tali componenti.

2. Una volta classificati i propri sistemi *legacy*, occorre attentamente valutare il ritorno che l'integrazione con il *WEB* può dare all'azienda. Sebbene l'adozione di una nuova tecnologia sia generalmente dettata da molteplici ed eterogenee variabili, le stesse possono essere ricondotte per lo più a due fattori: presenza sul mercato e miglioramento dell'efficienza aziendale. Se da un lato, infatti, il *WEB* ha valenza esterna permettendo di differenziare i canali distributivi, introdurre nuovi prodotti o servizi, incrementare la potenziale clientela, dal lato interno fornisce opportunità di ottimizzazione e di riduzione dei costi. Ad esempio, tramite *Intranet*, si può rendere più efficiente il *workflow* dei processi aziendali o, da un punto di vista tecnico, si possono semplificare le attività di installazione e gestione del software delle postazione *client*, attraverso l'accesso centralizzato tramite *browser*, diminuendo il *Total Cost of Ownership*.

Accertata l'esistenza di buone ragioni per connettere i sistemi informativi aziendali al *WEB*, la fase di progettazione della soluzione deve tenere conto di tre aspetti fondamentali:

- la selezione delle metodologie e degli strumenti;
- la valutazione dei rischi;
- la stima dei costi.

3. Circa il primo punto bisogna considerare che, sebbene il tema dell'*Enterprise Application Integration* (EAI) sia di grande attualità, soltanto il 15% delle aziende europee lo ha concretamente

affrontato; le banche sono tra gli operatori più attivi. Ciò significa che sul tema dell'integrazione è stata maturata ancora poca esperienza e, pertanto, non esistono approcci metodologici, tecnici e strumentali consolidati.

D'altro canto è possibile classificare le principali modalità di approccio al tema dell'integrazione individuando sostanzialmente tre opzioni:

- ***Webification***: ridisegnare la sola interfaccia di un'applicazione per consentire l'utilizzo del canale *WEB*, lasciando la logica di presentazione inalterata. Sono comunque possibili miglioramenti "estetici" per favorire, ad esempio, l'utilizzo dell'applicazione anche da parte di utenti non esperti;
- ***On-Line Transaction Integration***: consolidare la *presentation* di più applicazioni in un'unica interfaccia in modo da poter dare supporto a un processo di *business* transazionale e interattivo;
- ***Composite Application/Services***: creare nuove applicazioni, indipendenti dalla piattaforma e riutilizzabili, integrandole anche con componenti esistenti.

Da un punto di vista tecnologico, invece, la necessità, imposta dal mercato, di corredare qualsiasi prodotto di EAI delle funzionalità di integrazione con il *WEB* ha portato a una sovrapposizione fra le classi di strumenti. In altri termini, uno stesso livello di integrazione con il *WEB* può essere ottenuto attraverso numerosi prodotti di EAI, appartenenti però a categorie di costo completamente differenti.

4. L'utilizzo del *WEB* nell'ambito dei sistemi informativi aziendali fornisce molte opportunità ma presenta anche dei rischi.

Innanzitutto la scelta di procedere per integrazione e non per sostituzione, se da un lato permette di salvaguardare gli investimenti, dall'altro lato aumenta la complessità delle architetture.

Rischi più specifici riguardano:

- la scelta dei *partner* tecnologici;
- la sicurezza;
- il degrado delle prestazioni;
- la mancanza di standard consolidati.

Riguardo al primo punto, il problema risiede nella vasta platea di *vendor* di prodotti di integrazione che sono al momento presenti sul mercato. Molto spesso si tratta di fornitori di piccole dimensioni che hanno un numero limitato di risorse da dedicare al supporto dei clienti. Alcune aziende si stanno espandendo in modo rapido, altre sono concentrate su mercati verticali.

È opportuno, pertanto, privilegiare fornitori che possano garantire assistenza, conoscenza e presenza nel medio periodo.

Altro elemento di rischio è costituito dal potenziale indebolimento della sicurezza. Gli attacchi che preoccupano di più sono quelli che intervengono sul *workflow* di interazione con l'utente (c.d. "attacchi semantici").

Questi attacchi non mirano a colpire né gli aspetti fisici dei sistemi (es. apparati di memorizzazione) né, tantomeno, il software (es. algoritmi di crittografia), bensì modificano il corretto flusso delle informazioni (es. elusione di alcuni punti di controllo). In questo caso l'unica difesa è quella che si realizza sul piano della progettazione, prevedendo che i sistemi informatici

"prendano decisioni" sulla base di dati credibili. Un esempio è l'utilizzo di informazioni ridondanti e provenienti da sorgenti o canali differenti.

Circa la possibilità di degrado delle prestazioni, il fattore chiave risiede nella complessità dei collegamenti. Le transazioni effettuate via *WEB* seguono percorsi più lunghi e tortuosi rispetto a quelle realizzate con architetture tradizionali; invece, più intenso è il colloquio tra *client* e *server*, più breve dovrebbe essere il percorso dei dati.

A questi problemi strutturali si affianca la difficoltà di *tuning* dei sistemi. Il tema della verifica delle prestazioni degli ambienti *legacy* è stato da tempo affrontato e il mercato può ritenersi in una certa misura maturo, sia dal punto di vista degli strumenti a disposizione, sia del *know-how* delle aziende. Individuare, in un sistema *legacy,* la componente responsabile di prestazioni degradate è, perciò, ragionevolmente semplice. Le cose si complicano in un'architettura *WEB-legacy* a causa, fondamentalmente, della mancanza di strumenti che siano in grado di operare sull'intera architettura, piuttosto che su sezioni della stessa.

Un ultimo fattore di rischio è rappresentato dalla mancanza di standard affermati. Molto spesso i problemi di integrazione sono risolti attraverso percorsi e soluzioni proprietarie. Alcune statistiche a livello europeo dimostrano che il settore bancario, stante anche il problema delle numerose fusioni, non è tra quelli che ricerca maggiormente approcci standard. È però bene rammentare che la via del "fai da te" è estremamente rischiosa.

5. La selezione dei prodotti e dei *vendor* migliori, i presidi per la sicurezza, il mantenimento dei livelli di servizio, vanno comunque affiancate da riscontri di tipo economico. L'integrazione ha un costo, a volte anche molto significativo, tenuto conto:

- degli impatti organizzativi derivanti dai nuovi paradigmi di sviluppo delle applicazioni;
- dell'acquisto di specifici prodotti software per l'integrazione;
- delle modifiche da apportare ai sistemi *legacy*.

Il livello più alto di integrazione dei sistemi *legacy* con la tecnologia *WEB* è quello che consente di creare nuove applicazioni, indipendenti dalla piattaforma e riutilizzabili, integrandole anche con componenti esistenti (*Composite Application/Services*).

A tal fine occorre passare da una logica di applicazioni separate e funzionalmente indipendenti a un sistema informativo aziendale inteso come insieme di servizi cooperanti. Conseguenza di questo orientamento è che nel medio periodo (4-5 anni) le figure professionali necessarie per lo sviluppo delle applicazioni dovranno essere formate sui processi piuttosto che sui linguaggi di programmazione. L'obiettivo è infatti quello di approdare verso un'architettura orientata ai servizi (*Service Oriented Architecture* – SOA), che consenta di realizzare nuove applicazioni o *business service* riutilizzando per lo più componenti già pronte.

Tutto ciò comporterà costi di riconversione di alcune professionalità la cui entità dipende dalle singole realtà aziendali.

Circa l'acquisto di specifici prodotti software*,* è importante notare che un progetto di integrazione non particolarmente complesso può comportare costi di sole licenze d'uso che possono raggiungere facilmente i 500.000 euro, a cui vanno aggiunti gli eventuali costi per l'hardware e quelli, successivi, per la manutenzione.

Di fatto i *tool* di integrazione sono ancora molto costosi e alla portata di poche aziende. La sfida su cui si confronteranno nel medio periodo i maggiori *vendor* sarà proprio quella di fornire soluzioni alla portata della media impresa. È pertanto opportuno che l'acquisto di strumenti di integrazione sia basato su analisi approfondite.

Uno dei fattori che influenza in modo determinante il costo di un progetto di integrazione è legato alla dimensione delle modifiche sui sistemi preesistenti che si rendono necessarie. Infatti, a seconda della struttura del sistema *legacy* sul quale si deve operare, si può andare da interventi praticamente nulli (caso di sistemi altamente decomponibili) fino ad attività di dimensioni confrontabili al rifacimento stesso del sistema (caso dei sistemi monolitici).

Oltre alla struttura del *legacy*, una variabile fondamentale è la tipologia di integrazione che si vuole realizzare *(Webification, On-Line Transaction Integration, Composite Application/Services*), che può far variare di molto l'ordine di grandezza dei costi.

Tali costi possono essere compresi fra gli 80.000 euro di un semplice progetto per l'unificazione dell'interfaccia di accesso a dei sistemi mainframe (*On-Line Transaction Integration*), realizzata tramite tecnologia *WEB*, e i 10.000.000 di euro di un'iniziativa mirata a rimodellare completamente il sistema informativo aziendale nell'ottica di un'architettura basata su servizi condivisi e indipendenti dalla piattaforma (*Composite Application/Services*).

6. Lo studio fornisce inoltre alcune linee guida per il governo del processo di integrazione, con riferimento alle varie fasi (attività di studio, analisi e realizzazione del *WEB to legacy integration*).

7. Una specifica sezione è dedicata all'esame dei modelli di sviluppo applicativo adottati e di alcune tendenze che al momento sembrano particolarmente promettenti. In questo ambito, è stato fatto specifico riferimento ai *WEB services,* che dovrebbero facilitare l'integrazione permettendo di riutilizzare il software già prodotto, e alla *Model Driven Architecture* (MDA), che rappresenta il tentativo, da parte dell'*Object Management Group*, di realizzare un modello di sviluppo delle applicazioni orientato all'interoperabilità e all'integrazione.

8. In conclusione, viene prospettata una linea guida organizzativa all'integrazione che, mediando fra l'impulso radicale di creare un'architettura applicativa completamente nuova e uno mirato a conservare il più possibile i sistemi *legacy*, consenta di cogliere sia gli obiettivi strategici, sia quelli tattici, trovando di volta in volta la soluzione ottimale nell'ambito di un modello logico di riferimento (c.d. *Enterprise Nervous System*) delle connessioni fra i vari sistemi.

1. Definizione di *legacy system*

Il termine inglese *legacy* è utilizzato per indicare qualcosa di valore ottenuto attraverso un'eredità. Associato a un sistema informativo, pertanto, il termine *legacy* ne evidenzia le caratteristiche di "valore aziendale" e di "provenienza dal passato".

A un sistema *legacy* sono associate una o più delle seguenti peculiarità [Bat 00]:

è fondamentale per l'operatività dell'organizzazione;
è stato oggetto, nel tempo, di significativi investimenti;
il nucleo iniziale è stato progettato secondo vecchie concezioni;
le applicazioni che lo compongono sono prevalentemente *stand-alone*, ognuna progettata e realizzata indipendentemente dalle altre;
è scritto con linguaggi tradizionali (COBOL, PL1, etc.);
non è ben documentato.

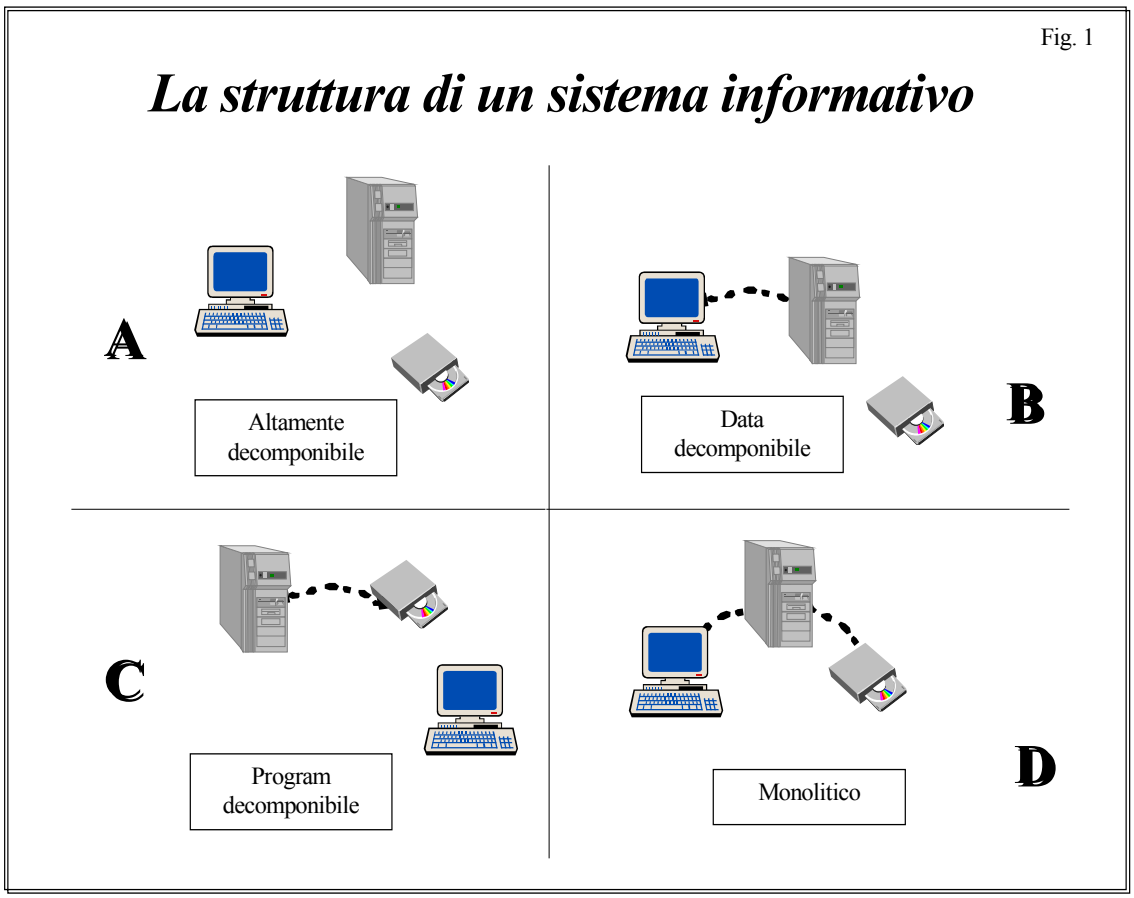
Nell'ambito dei sistemi informativi, però, il termine *legacy* non sta a indicare qualcosa di vecchio. Tuttora infatti si continuano a progettare e realizzare, oltre che a mantenere, sistemi con tecnologie tradizionali.

Un *legacy system* non è, pertanto, un sistema in via di smantellamento ma, al contrario, uno strumento su cui l'azienda fa affidamento anche per il futuro. Quindi, la convivenza fra questi sistemi e quelli realizzati con nuovi paradigmi e tecnologie è un dato imprescindibile.

Un'altra caratterizzazione dei sistemi *legacy*, molto importante ai fini delle possibilità di integrazione, è quella che fa riferimento agli aspetti "strutturali" dei sistemi stessi [Bat 00]. Più in particolare possono essere definite quattro tipologie di sistemi (cfr. Fig. 1):

- **Altamente decomponibili (A);** sistemi, cioè, che sono ben strutturati e i cui componenti applicativi sono separabili in tre livelli di logica: di presentazione, applicativa e di accesso ai dati;
- ***Data* decomponibili (B);** sistemi con due livelli logici separati: da una parte i servizi di accesso ai dati, dall'altra parte la presentazione e la logica applicativa fusi in un unico blocco;
- ***Program* decomponibili (C);** sistemi sempre a due livelli: da una parte la logica di accesso ai dati e applicativa fuse insieme, dall'altra parte la logica di presentazione;
- **Monolitici (D);** sistemi in cui i tre livelli logici non sono in alcun modo separabili.

Fig. 1



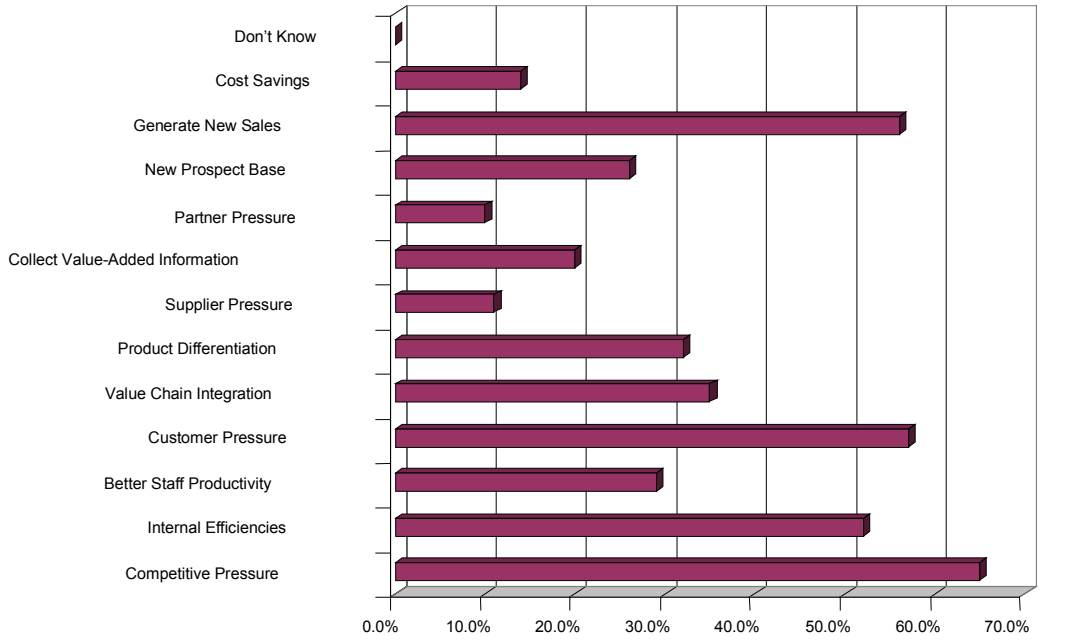
Questa classificazione consente, a sua volta, di definire la seguente scala di difficoltà teorica per l'integrazione di un sistema informativo *legacy* in un ambiente tecnologicamente più avanzato:

Tipologia di sistema	Integrabilità
Altamente decomponibile	Facile
Data decomponibile	Media
Program decomponibile	Media
Monolitico	Difficile

2. Le motivazioni a sostegno dell'integrazione

Nell'ICT è sempre difficile individuare la causa alla base della diffusione di un nuovo paradigma, essendo essa generalmente la risultante di numerose componenti concorrenti. Una riprova di questa affermazione si ha analizzando le diverse motivazioni che spingono gli investimenti per l'*e-business* (cfr. Fig. 2).

Fig. 2



(Fonte META Group)

In particolare, vengono in evidenza:

- la pressione della concorrenza (*competitive pressure*);
- la domanda della clientela (*customer pressure*);
- la creazione di nuovi *business* (*generate new sales*);
- l'aumento dell'efficienza interna (*internal efficiencies*).

Di seguito vengono analizzati questi fattori, esaminandone le implicazioni sotto il profilo dell'integrazione fra sistemi *legacy* e tecnologia *WEB*.

2.1 La pressione della concorrenza

La scorsa primavera la Banca d'Italia ha condotto un'indagine sulle funzioni produttive e distributive delle banche, basata su un campione di 329 aziende rappresentante oltre il 90% del sistema in termini di totale attivo, con particolare riguardo ai nuovi canali distributivi e ai servizi di *e-commerce* attivi alla fine del 2001.

Circa l'ampiezza del ricorso ai diversi canali distributivi, lo sportello risulta ancora il più utilizzato (85% dei casi) ma l'*Internet banking* con funzioni dispositive ha avuto un notevole incremento: il 62% del campione lo utilizza, seppure con diversi livelli di intensità. Più in particolare, solo il 4% lo considera il canale principale, il 16% lo vede come canale rilevante, mentre il 42% lo ritiene ancora un canale marginale.

Delle banche che propongono servizi *Internet,* soltanto un limitatissimo numero mantiene basi dati distinte per l'operatività tradizionale e per l'attività su *WEB*, gestendo rapporti di conto separato per canale distributivo. Le grandi banche adottano in prevalenza soluzioni basate su archivi unici, mentre le piccole usano basi dati distinte con procedure di allineamento *batch*. Questi pochi indicatori evidenziano come la piena integrazione fra sistemi *legacy* e quelli dedicati *all'Internet banking* rappresenti una delle questioni più rilevanti per le banche, soprattutto per quelle che perseguono strategie di multicanalità integrata.

2.2 La domanda della clientela

Sempre alla fine del 2001, i clienti *Internet* erano circa 1,3 milioni, (1,7 quelli che hanno operato attraverso i promotori; poco meno di 1 milione quelli che hanno utilizzato il canale telefonico). Se si considerano anche gli utenti con profilo solo informativo, il numero di utilizzatori di *Internet* sale a circa 3,2 milioni.

2.3 La creazione di nuovi business

La spinta verso l'integrazione dei sistemi tradizionali con il *WEB* non è però correlata soltanto alla necessità di differenziare i canali distributivi degli usuali servizi bancari.

Molte banche, sempre secondo quanto risulta dall'indagine della Banca d'Italia sull'utilizzo di *Internet*, hanno avviato progetti che sfruttano il *WEB* per estendere l'ambito di attività al di fuori del tradizionale comparto finanziario, in particolare nel settore dell'*e-commerce*, anche se non mancano iniziative avviate in collaborazione con società operanti nel settore tecnologico e delle telecomunicazioni; 59 banche del campione offrono servizi per la gestione delle infrastrutture informatiche (accesso a *Internet*, gestione di portale, etc.) e di sicurezza (certificazione e firma digitale).

2.4 L'aumento dell'efficienza interna

Secondo gli analisti di mercato, molte aziende stanno investendo negli strumenti necessari per portare i dati degli *host* sulle *Intranet*, sulle *Extranet* e sui siti *WEB*. Uno studio condotto recentemente da IDC su 500 grandi aziende statunitensi ed europee ha rivelato che il 46% di esse ha pianificato progetti per permettere l'accesso degli *host* alla *Intranet*, e il 40% sta sviluppando funzioni di accesso all'*host* per i suoi siti *WEB* pubblici.

I progetti *WEB-to-legacy* comprovano il principio della massa critica. Poiché moltissime applicazioni *legacy* hanno a che fare con l'immissione dei dati, i pagamenti, la fatturazione e così via, per loro natura coinvolgono i *partner* dell'azienda e i clienti. Quando uno di questi automatizza parte del processo con un'architettura *WEB-to-legacy*, traina anche gli altri.

Alla base dell'integrazione tra applicazioni *legacy* e mondo *WEB* vi è una profonda trasformazione di mentalità. Per generazioni sono stati progettati e sviluppati prodotti software finalizzati a risolvere un preciso problema e con un singolo scopo, soluzione che ora è considerata inadeguata.

Lo scopo dei sistemi di *Enterprise Application Integration* (EAI) invece è collegare e tenere sotto controllo i singoli software, creati per esempio per il controllo degli inventari di magazzino, per l'automazione delle vendite oppure per la gestione delle risorse umane, con lo scopo primario di liberare l'informazione e renderla disponibile a chiunque faccia parte della *value chain.*

È per questo motivo che spesso l'*Enterprise Application Integration* viene associata con l'*e-business*, in particolare con il commercio elettronico *business-to-business* (B2B).

Infatti i sistemi di *e-commerce*, per dare valore aggiunto a un'azienda, devono avere a disposizione in tempo reale dati provenienti da fonti diverse: un esempio è fornito dall'acquisizione di ordini *on line*, che comporta modifiche agli archivi di sistemi differenti, come quelli per la gestione delle vendite, per le fatturazioni, per la gestione dei magazzini e per l'aggiornamento delle scorte; inoltre un'operazione di questo tipo potrebbe attivare una serie di processi a cascata, come per esempio l'invio di un nuovo ordine ai fornitori e il controllo sugli acquisti, e fornire dati utili per le campagne pubblicitarie degli uffici *marketing*.

Da un punto di vista tecnico, inoltre, l'adozione della tecnologia *WEB,* se ben sfruttata, può determinare anche significative riduzioni dei costi. Per esempio, l'utilizzo di un *browser* consente di semplificare le attività di installazione e gestione delle equivalenti componenti applicative *client*, permettendo di ridurre il *Total Cost of Ownership* (TCO).

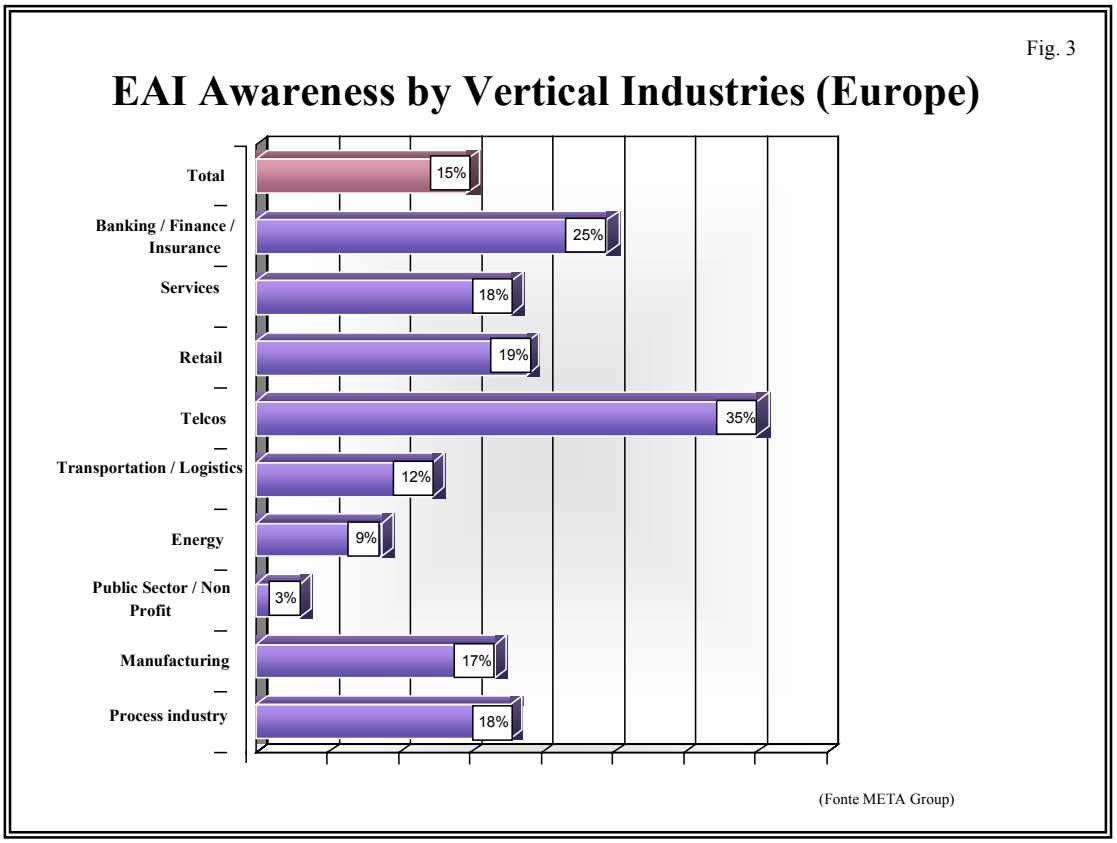
Sempre in questa ottica, un'altra fonte di risparmio può essere ottenuta attraverso un processo di unificazione di accesso ai sistemi informativi, che consente di migliorare l'efficienza e ridurre il tasso di errore delle attività operative.

3. I diversi approcci all'integrazione

3.1 L'integrazione fra sistemi diversi

L'*Enterprise Application Integration* (EAI) finora è stata effettivamente realizzata in contesti molto limitati. In questo senso le banche hanno svolto un ruolo pionieristico. Una recente ricerca condotta da META Group [Pat 02] (cfr. Fig. 3), nell'ambito di un campione di aziende europee, ha evidenziato infatti che, rispetto a un dato medio del 15%, il maggior grado di sensibilità alla tematica dell'EAI si rileva, dopo il settore delle telecomunicazioni (35 aziende su 100), in quello bancario/assicurativo (25 aziende su 100).

Fig. 3



I livelli e le modalità di integrazione sono numerosi e profondamente differenti in termini di impatti, tempi, costi e ritorni.

Una classificazione di riferimento, ancorché non l'unica possibile, è quella che vede le seguenti tipologie di integrazione:

- abilitazione alla comunicazione fra piattaforme applicative;
- accesso *standard* ai dati aziendali;
- interoperabilità fra applicazioni;
- interoperabilità fra processi, sia aziendali, sia inter-aziendali.

La prima tipologia, minimizzando gli impatti sulle applicazioni e tramite un "semplice" scambio di messaggi, consente un'integrazione di livello puramente infrastrutturale attraverso la connessione di più ambienti. Le modalità più diffuse sono le seguenti:

- MOM – *message oriented middleware*, che assicura una connettività di tipo asincrono tramite scambio di messaggi tra code;
- RPC - *remote procedure call*, che consente una connettività di tipo sincrono tra processi diversi, anche distribuiti;
- ORB – *object request broker*, che permette la definizione delle interfacce per una connessione sia sincrona, sia asincrona. Attualmente due sono gli standard più diffusi, tra loro non interoperabili: CORBA (*Common Object Request Broker Architecture*), che è promosso dall'*Object Management Group* (OMG), e OLE/DCOM (*Object Linking and Embedding/Distributed Common Object Model*).

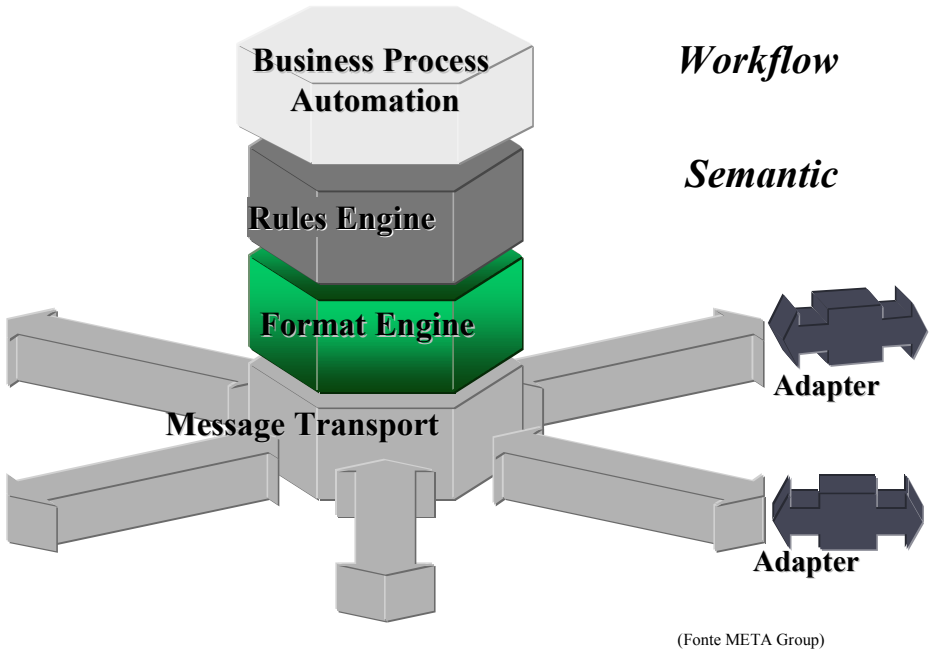
L'integrazione attraverso i dati aziendali, nell'ambito di un contesto di interventi minimali sulle applicazioni esistenti, pone l'accento sul significato dell'informazione andando oltre la semplice connessione tecnica fra due sistemi.

Un tipico esempio di questo livello di integrazione è quello fornito dall'ODBC (*Open Data Base Connectivity*) che consente di operare attraverso un'interfaccia standard su archivi che possono trovarsi su piattaforme eterogenee e/o con DBMS differenti.

Riassumendo, quindi, l'abilitazione alla comunicazione fra piattaforme applicative e l'accesso *standard* ai dati aziendali costituiscono un'integrazione "semplice" che, senza modificare la logica dei processi aziendali e minimizzando gli interventi di modifica sulle applicazioni esistenti, consente di mettere in comunicazione piattaforme informatiche diverse al fine di agevolare la condivisione dei dati.

Quando si considera invece l'integrazione dal punto di vista dell'interoperabilità fra applicazioni (*Rules Engine*) e, ancora di più, fra processi (*Business Process Automation*), aumenta il livello di astrazione dalla tecnologia e dalle infrastrutture (cfr. Fig. 4). Di conseguenza cresce l'impatto sui sistemi e il valore delle modifiche da apportare ai sistemi stessi.

Fig. 4



Un esempio di riferimento tecnologico per questa tipologia di "connessione" è rappresentato da framework di integrazione costituiti essenzialmente dai seguenti elementi:

- una piattaforma tecnologica di integrazione;
- un meccanismo di integrazione degli eventi;
- un sistema interattivo di *routing* e trasformazione dati basato su regole;
- *adapter* per la connessione con i prodotti di mercato più diffusi.

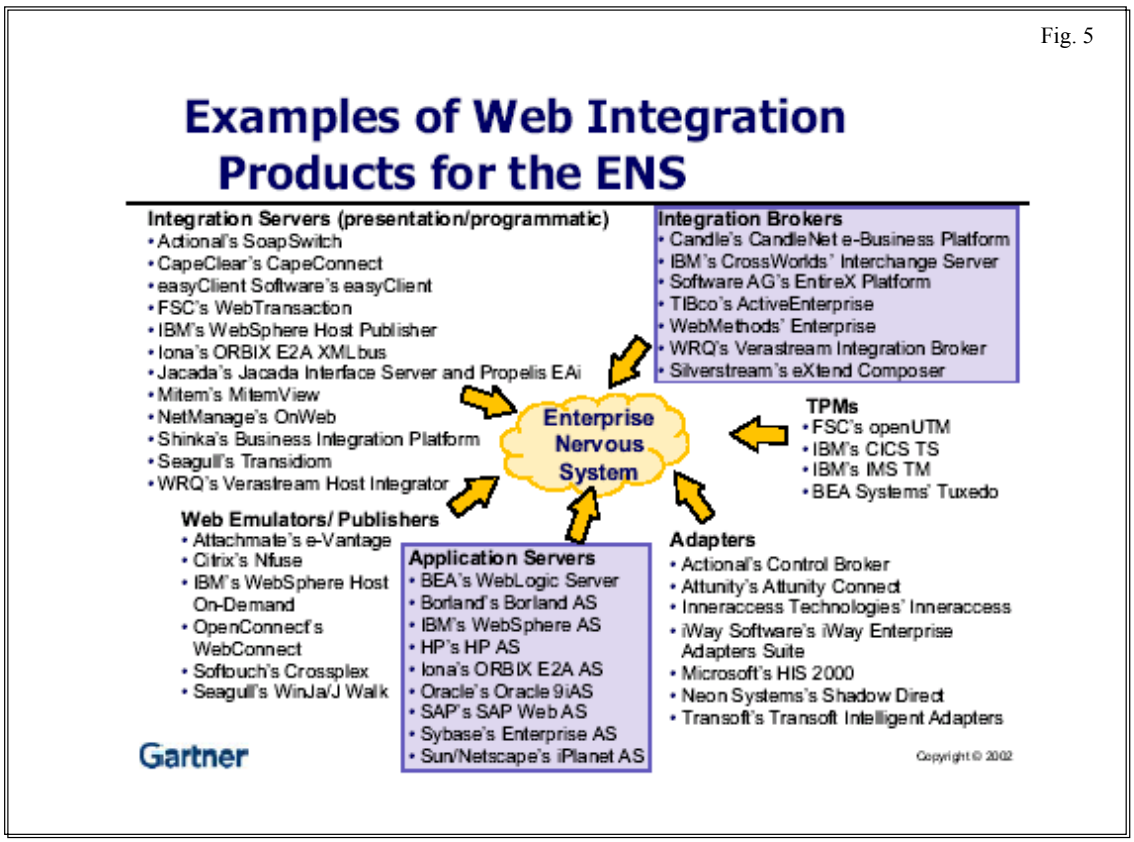
Indipendentemente dalle modalità tecniche con cui si ottiene l'integrazione, comunque, l'obiettivo ultimo dovrebbe essere quello di definire l'*Enterprise Nervous System* (ENS) aziendale, che significa avere un'infrastruttura di riferimento per collegare le applicazioni e portare a livello di *middleware* parte dell'intelligenza che oggi risiede interamente nelle applicazioni stesse[1].

A titolo puramente indicativo, in Figura 5 è riportato un elenco di prodotti che, a diversi livelli, permettono l'integrazione fra sistemi, pur facendo esplicito riferimento allo specifico caso del *WEB*. In effetti la necessità di integrare i sistemi informativi preesistenti con la tecnologia *WEB* è stato il

[1] Un'architettura tradizionale non fa altro che trasferire i dati dall'applicazione mittente a quella destinataria, scelta dal mittente stesso. Un'architettura ENS ha intelligenza pari a quella delle applicazioni che collega, ponendosi come intermediario fra gli utenti e le procedure informatiche. Ad esempio, è compito dell'ENS mantenere in archivi condivisi, non gestiti da alcuna applicazione, alcuni dati operativi significativi.

fattore trainante per un processo di evoluzione di molti prodotti di mercato. Anche soluzioni inizialmente nate per risolvere problemi di integrazione "sistemistica" fra differenti ambienti sono state velocemente riposizionate sul *WEB* che oggi rappresenta un *must* per ogni proposta commerciale.

Fig. 5



Il numero di fornitori che propongono prodotti di integrazione è in continua crescita ma, secondo Gartner [Pez 02], entro il 2005 almeno il 75% delle società che propongono strumenti dedicati alla sola integrazione con il *WEB* saranno assorbite dai grandi *vendor* di *middleware*. In una visione strategica, pertanto, un'azienda dovrebbe guardare a *partner* consolidati e con visione ad ampio spettro. Nonostante ciò, alcune soluzioni tattiche potrebbero anche passare per prodotti di nicchia che, nel breve, assicurino un rapporto costi/benefici molto vantaggioso.

Ed è in questo senso che vanno letti gli elenchi di prodotti riportati in Figura 5. Al di là delle classiche piattaforme *middleware* (TPM) che hanno subito un processo di evoluzione attraverso l'aggiunta di funzioni di supporto per i protocolli *Internet* e Java, gli altri prodotti fanno riferimento essenzialmente a due diverse politiche che i *vendor* perseguono.

I fornitori di *WEB-to-host terminal emulator, host publisher, adapter, presentation and programmatic integration server* puntano su una competenza profonda riguardo alle modalità di integrazione con il *WEB*, ma il loro mercato è generalmente ristretto poiché le soluzioni sono limitate a specifiche piattaforme. Di converso i fornitori di *application server e integration broker* offrono un più largo spettro di funzionalità e possono soddisfare ampi mercati orizzontali, a scapito della specializzazione e dell'ottimizzazione.

In questo senso i fornitori di *application server e di integration broker* tenderanno sempre di più, attraverso accordi commerciali o tramite acquisizioni, a far confluire gli strumenti specialistici nella loro offerta.

3.2 L'integrazione fra sistemi legacy e tecnologie WEB based

Come visto nel paragrafo precedente, oramai l'EAI è totalmente caratterizzata dal tema dell'integrazione dei sistemi tradizionali con la tecnologia *WEB*. In altre parole, non sembrerebbe avere più senso parlare di soluzioni per l'integrazione fra sistemi diversi senza considerare l'utilizzo del canale *WEB*. Ciò può essere accettato se si considera il tema dal solo punto di vista dei prodotti software.

Teoricamente, però, l'integrazione di un sistema *legacy* con la tecnologia *WEB* ha una dimensione autonoma.

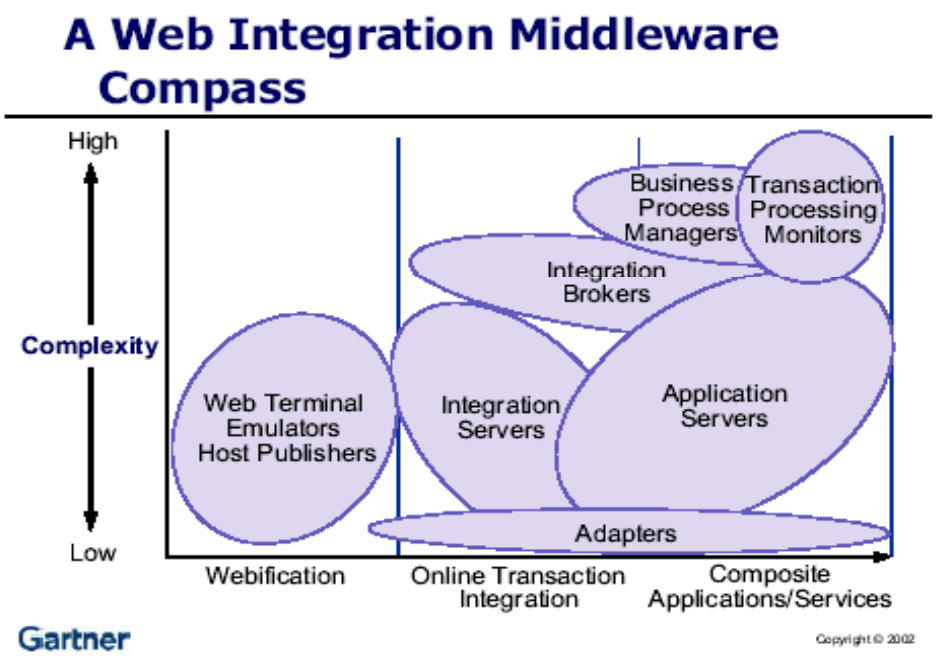
In questo senso, sembra più opportuno proporre una classificazione ortogonale a quella del paragrafo 3.1 e che faccia riferimento ai diversi livelli evolutivi che un sistema *legacy* può raggiungere attraverso l'utilizzo della tecnologia *WEB*. Generalmente, infatti, il livello di integrazione fra tecnologia *WEB* e *legacy system* è connesso con le differenti fasi del processo di adattamento delle applicazioni che un'azienda deve "sopportare" in nome del raggiungimento del maggior grado possibile di sfruttamento nell'ambito, ad esempio, dell'*e-business*.

In particolare, può essere di aiuto la seguente classificazione [Pez 02]:

- *Webification*: ridisegnare la sola interfaccia di un'applicazione per consentire l'utilizzo del canale *WEB*, lasciando la logica di presentazione inalterata. Sono comunque possibili miglioramenti "estetici" per favorire, ad esempio, l'utilizzo dell'applicazione anche da parte di utenti non esperti;
- *On-Line Transaction Integration*: consolidare la *presentation* di più applicazioni in un'unica interfaccia in modo da poter supportare un processo di *business* transazionale e interattivo;
- *Composite Application/Services*: creare nuove applicazioni, indipendenti dalla piattaforma e riutilizzabili, integrandole anche con componenti esistenti.

L'ortogonalità con la classificazione del paragrafo 3.1 appare evidente se si considera la proiezione grafica fra questa articolazione a tre livelli e le principali tipologie di strumenti di integrazione che il mercato dell'ICT mette a disposizione (cfr. Fig. 6).

Fig. 6



Il risultato della tumultuosa e, in un certo senso, forzata evoluzione di alcuni prodotti verso il *WEB* è stato quello di una sovrapposizione fra le classi di strumenti per l'integrazione. Pertanto non c'è univocità fra prodotti di mercato e tipologia di integrazione con il *WEB*.

Esiste quindi un problema di mancanza di specificità dei prodotti, sebbene nell'ambito dell'informatica sia ormai acclarata l'impossibilità di considerare domini a intersezione nulla. Ciononostante la strategia di lungo termine dovrebbe essere quella dell'utilizzo di *application server e integration broker* quali piattaforme di integrazione con il *WEB*, non escludendo la possibilità di utilizzare *tool* tattici, quali *integration server o host publishing server,* per ragioni di opportunità, soprattutto in un'ottica di massimizzazione del rapporto costi/benefici.

3.3 Dall'integrazione ai servizi

Sebbene i sistemi *legacy* continueranno a sopravvivere ancora per molto tempo, e l'integrazione con la tecnologia *WEB* rappresenti di fatto una modalità per prolungarne la vita residua, essi non rimarranno per sempre nella forma attuale. Le aziende prima o poi ravviseranno l'opportunità di procedere a un ri-disegno dell'*Enterprise Nervous System*. In sostanza si arriverà a un punto di discontinuità con il passato.

Ma quale è il nuovo modello a cui si dovrebbe tendere? Da sempre le aziende cercano di automatizzare i processi di *business*. In tempi di rete, le aziende cercano di automatizzare i processi di *business* collaborativi con un numero crescente di clienti o *partner*. Ma forme ubique di collaborazione, come avviene per il telefono e il fax, non sono automatiche. E forme automatizzate di collaborazione, come l'EDI, sono proprietarie e non ubique.

Si tratta allora di trovare una forma ubiqua che automatizzi i processi. Applicazioni collaborative con servizi esposti e documentati sistematicamente dal lato *server* e fruiti in modo agile ed opportunistico dal lato *client*. In altre parole, i sistemi interaziendalmente cooperanti vanno basati sulle *Service Oriented Architecture* (SOA).

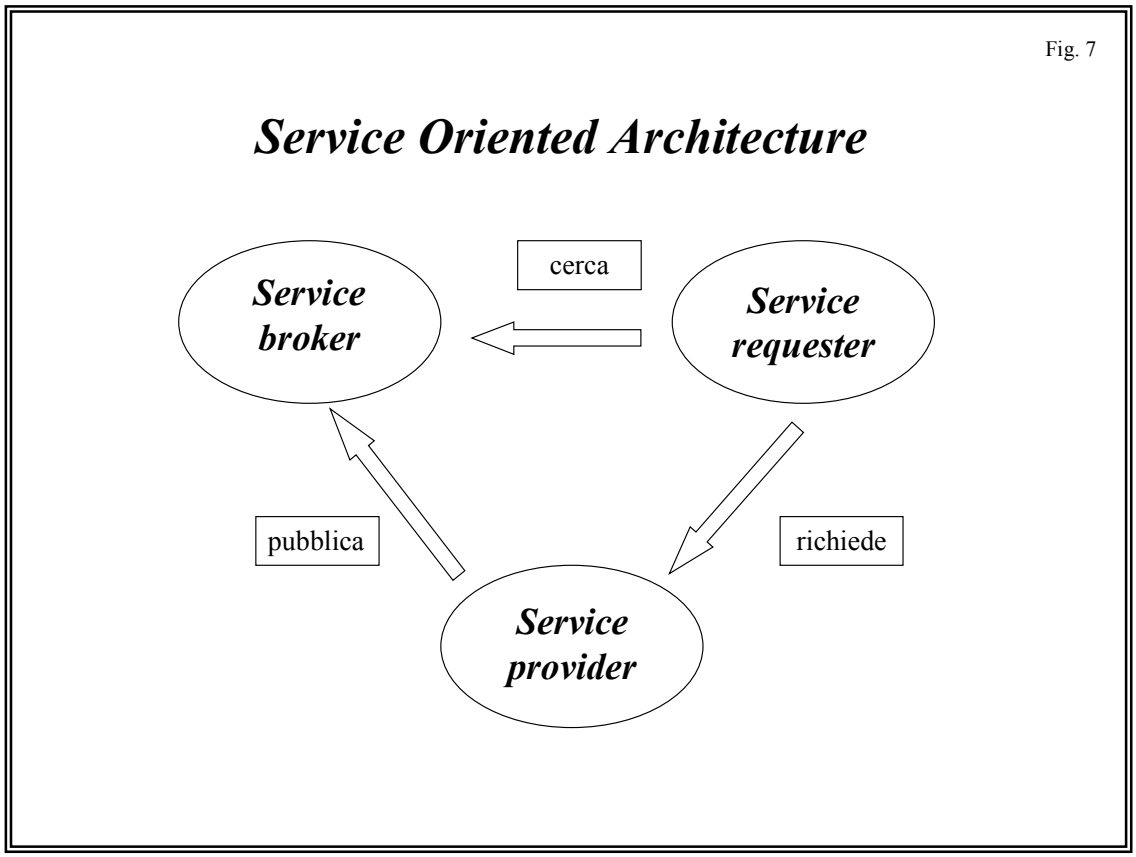
Le *Service Oriented Architecture* sono una derivazione del concetto *object oriented* dell'ingegneria del software, visto da una prospettiva di *business* piuttosto che tecnologica. Si basano, pertanto, sull'assunto che processi complessi possano essere scomposti in unità di dimensioni limitate e compiute in se stesse.

Le SOA nascono con l'obiettivo di realizzare processi di *business* (servizi) da rendere disponibili a un qualsiasi "richiedente". Un'architettura SOA consente di realizzare nuove applicazioni o *business services* utilizzando "componenti" riusabili che implementano singole funzioni.

Le componenti base di un'architettura SOA sono le seguenti (cfr. Fig. 7):

- ***service provider,*** che pubblica il proprio servizio e risponde alle richieste di utilizzo;
- ***service broker,*** che registra i servizi e offre funzioni di ricerca;
- ***service requester,*** che utilizza un *service broker* per individuare il servizio necessario e richiederne l'esecuzione, senza avere alcuna conoscenza dei singoli componenti che lo implementano.

Fig. 7



Un numero sempre maggiore di organizzazioni IT sta adottando il modello SOA per ottenere i benefici che possono provenire dalla condivisione dei *Business services* all'interno delle rispettive imprese. E le SOA a loro volta vengono estese fino a supportare le iniziative di B2B, abilitando quindi anche *Business partner*, fornitori e altri interlocutori esterni a condividere i servizi.

Condizione base per l'implementazione di una architettura SOA è la creazione delle funzioni modulari (componenti) da utilizzare per la definizione del servizio stesso; queste modifiche sono spesso onerose sui sistemi esistenti (cfr. par. 5.3) ma indispensabili per la realizzazione della nuova architettura, che è alla base di un'integrazione fra sistemi *legacy* e tecnologia *WEB* che vada al di là della semplice, seppure in qualche caso utile, *webification* (cfr. par. 3.2) di un'applicazione tradizionale.

4. I rischi connessi con l'integrazione

Indubbiamente, l'integrazione fra sistemi *legacy* e tecnologia *WEB* tende ad aumentare la complessità sistemica.

Più in generale, nel momento in cui si decide di aggiungere una nuova tecnologia a quelle già esistenti, di fatto, si eleva di uno o più livelli il complesso di architetture, strumenti, relazioni, etc.

Pertanto, il rischio maggiore dell'integrazione consiste nel fatto che, aumentando la complessità sistemica, aumenta automaticamente anche il livello di rischiosità del sistema stesso. E se l'approccio che guida un'azienda nel processo di innesto dell'innovazione è sempre quello dell'integrazione, la complessità assume l'andamento di una funzione crescente.

Un modo per interrompere la crescita di complessità è quello di utilizzare anche la "migrazione", magari anticipando i tempi rispetto al raggiungimento del punto di discontinuità (cfr. par. 4.3). Infatti, il passaggio graduale di un sistema da una vecchia tecnologia a una innovativa pone il problema della complessità soltanto per il periodo in cui le due tecnologie devono convivere. Nel momento in cui il passaggio al nuovo sistema è completato, la complessità dovrebbe diminuire. Ovviamente la migrazione comporta dei costi nel breve periodo superiori a quelli dell'integrazione e la discriminante sul suo utilizzo potrebbe essere proprio la valutazione dei benefici in termini di diminuzione del rischio sistemico.

Oltre alla complessità, però, esistono altre problematiche che, se non opportunamente affrontate, possono rappresentare delle aree di rischio. I paragrafi successivi sono dedicati all'esposizione delle tematiche più significative a tale riguardo.

4.1 La scelta dei partner tecnologici

La maggior parte dei *vendor* specializzati nei prodotti per l'integrazione sono relativamente piccoli e hanno un numero limitato di risorse da dedicare al supporto dei clienti[2]. Alcuni si stanno espandendo in modo molto rapido, altri sono concentrati solo su mercati verticali. I prodotti sono spesso complessi e non sempre aderenti agli standard più affermati.

Per questi motivi non è consigliabile avviare un progetto di integrazione fidando sulle capacità di risolvere *in house* i problemi che si incontreranno durante il percorso. La collaborazione con i *vendor* è fondamentale ed è quindi importante la fase di selezione del fornitore.

[2] Questo fenomeno è più marcato in Europa rispetto alla situazione degli Stati Uniti.

Tra l'altro un unico prodotto non risolve quasi mai tutti i problemi di integrazione. Soprattutto nelle aziende più grandi, infatti, la complessità e diversificazione del *business* dei vari settori impone l'utilizzo di strumenti di *vendor* diversi. In questo senso, quindi, la scelta di fornitori che possano garantire assistenza, conoscenza e "presenza" nel tempo è basilare per ogni iniziativa di integrazione.

Nonostante tutto, comunque, è importante considerare che il "tempo di vita" delle soluzioni tende a ridursi. Le nuove versioni e le evoluzioni degli strumenti, anche di quelli più diffusi, si susseguono velocemente creando, talvolta, problemi di compatibilità con il passato. É pertanto opportuno presidiare il mercato con risorse interne specializzate che garantiscano all'azienda l'utilizzo di soluzioni a elevata stabilità.

4.2 L'indebolimento della sicurezza

4.2.1 Non solo accessi indesiderati

Il tema della sicurezza sta diventando uno dei nodi cruciali per lo sviluppo delle tecnologie dell'informazione e della comunicazione. Sempre più spesso si ha notizia di qualche falla nella gestione della sicurezza, con risultati che vanno dal semplice imbarazzo fino alla perdita di ingenti quantità di denaro.

Queste situazioni possono verificarsi anche nei sistemi *legacy*, ma i rischi si moltiplicano in un ambiente distribuito, per esempio *Internet*.

Cosa si rischia collegandosi alla "rete"? La prima e immediata risposta riguarda la possibilità di intrusioni con lo scopo di distruggere, manomettere o sottrarre informazioni. Ma, similmente a quanto accade nel mondo fisico, gli "attacchi" possono essere anche molto sofisticati e comunque raffrontabili a quelli che generalmente possiamo definire come comportamenti illegali e/o scorretti.

Il problema è a tal punto sentito che la *Internet society* ha pubblicato *l'Internet security glossary*, che definisce le seguenti principali categorie attraverso cui può essere realizzata la sicurezza sul *WEB*:

- autenticazione;
- confidenzialità;
- controllo degli accessi;
- disponibilità;
- integrità;
- informazione sensibile (*sensitive information*);
- non disconoscimento;
- traccia di controllo (*audit trail*).

La sicurezza sul *WEB* è un tema complesso e articolato. L'aumento delle interconnessioni fra sistemi diversi impone di non pensare alla sicurezza soltanto in termini di tecnologia. In una recente pubblicazione [Sch 00], Bruce Schneier sostiene che ci sono tre tipi di attacchi possibili a una rete: gli attacchi fisici (che mirano a colpire gli elaboratori e l'elettronica in genere), gli attacchi sintattici (diretti ai pacchetti software, agli algoritmi di crittografia, ai

protocolli e alle restrizioni di accesso ai servizi) e quelli semantici (che agiscono sul corretto flusso delle informazioni).

Questi ultimi sono i più difficili da contrastare e contro di essi non sono sufficienti le firme digitali e i sistemi per l'autenticazione e per l'integrità. La difesa deve essere realizzata sul piano della progettazione dei sistemi, evitando che essi prendano decisioni sulla base di informazioni la cui credibilità è incerta. Solo informazioni ridondanti e provenienti da sorgenti o canali differenti e certificati possono meritare fiducia e determinare azioni a elevata criticità.

In altre parole, per quanto sia prevedibile che la ricerca fornirà sempre più sofisticate ed efficaci difese contro gli attacchi alla sicurezza, questo non giustifica una fiducia cieca nelle loro prestazioni. È pertanto necessario sviluppare sistemi che vanifichino gli attacchi semantici disegnando interazioni "uomo-macchina" che limitino all'essenziale gli automatismi procedurali [DeM 01].

4.2.2 Le tracce di audit

Sempre al fine di sottolineare l'importanza e la complessità del tema della sicurezza nell'universo *WEB*, è opportuno soffermare l'attenzione su una delle caratteristiche enunciate nel paragrafo precedente, *l'audit trail*.

La traccia di *audit* può essere definita come la "registrazione cronologica delle attività del sistema sufficiente per consentire la ricostruzione, la revisione e l'esame della sequenza di situazioni e di attività che hanno riguardato o che hanno condotto a un'operazione, una procedura o un evento in una transizione dal suo inizio ai suoi risultati finali".

Le *audit trail* sono, quindi, uno strumento fondamentale per comprendere "chi ha fatto che cosa" in un sistema e la loro difficoltà di implementazione è direttamente proporzionale al grado di distribuzione di un sistema informativo. Più è alto il numero delle relazioni fra componenti, maggiore è la probabilità di "perdere le tracce" del soggetto o del processo che hanno innescato un'azione rivelatasi dannosa oppure non corretta.

Anche in questo caso, pertanto, l'attenzione più che agli strumenti di mercato deve essere posta all'analisi del sistema, attraverso una progettazione che, eventualmente semplificando l'architettura, consenta il più alto grado di tracciabilità possibile.

4.3 Difficoltà di simulare l'ambiente di produzione

Quasi tutte le aziende sono dotate da tempo di ambienti di test, in genere separati da quelli di produzione, al fine di verificare e certificare i rilasci che provengono dagli ambienti di sviluppo. In tal modo vengono limitati i rischi di malfunzionamenti in ambiente operativo a causa delle nuove versioni di software.

La creazione di ambienti di test separati (cloni) per i sistemi *legacy* è stata generalmente risolta attraverso la generazione di nuove partizioni sui *mainframe* e il relativo allineamento dei dati.

La realizzazione di un ambiente clone per un'architettura *WEB* pone invece alcuni problemi, i più significativi dei quali sono rappresentati dalla parcellizzazione degli apparati e, conseguentemente, dal costo.

Circa il primo punto l'architettura *WEB*, a differenza di quelle tradizionali basate su *mainframe,* non offre di fatto la possibilità di utilizzare partizioni degli stessi ambienti di produzione. Generalmente, infatti, nelle architetture *WEB* si ha un aumento del numero degli apparati (hardware e reti) e una polverizzazione della dimensione degli stessi. Molto spesso i *server* sono rappresentati da PC che non sono fisicamente in grado di ospitare due ambienti distinti.

In questo senso, un'ulteriore difficoltà all'utilizzo degli ambienti di produzione per il test è rappresentata dall'obbligatoria disponibilità "24 X 7" delle applicazioni.

Generare un ambiente clone, pertanto, può anche richiedere la duplicazione degli apparati di produzione. Data la numerosità di elementi in gioco (hardware, software applicativo, *middleware*, etc.), ciò potrebbe comportare costi di impianto e, soprattutto, di gestione molto onerosi.

4.4 La rapidità di successione delle nuove versioni del software

Difficilmente l'innalzamento del livello di *release* di un prodotto è indolore per l'intero sistema. Questa problematica, già molto sentita nei sistemi tradizionali, aumenta in maniera esponenziale in un'architettura *WEB-legacy* in quanto il numero di sistemi che interagiscono è elevato.

Lo scenario tipico di un sistema *legacy* prevede una certa garanzia di funzionamento grazie a una certificazione di compatibilità (quasi sempre attendibile e documentata) fra le versioni dei prodotti che interagiscono.

Le architetture *WEB-legacy,* invece, sono solitamente costituite da un numero elevato di componenti di *vendor* diversi. Non è pertanto possibile avere delle garanzie sulla compatibilità dei singoli prodotti al cambio della *release* di uno degli stessi.

In altre parole, il cambio di *release* di un prodotto può comportare la necessità di innalzare a catena la versione di tutte le altre componenti, fino a quando ciò è possibile. Nel caso peggiore, infatti, ci si può trovare a gestire situazioni di incompatibilità fra varie versioni dei prodotti con relativa instabilità dell'intero sistema.

La rilevanza di questo problema è direttamente proporzionale al grado di frammentazione dell'architettura di riferimento. Più è diversificato l'insieme di prodotti/fornitori che concorrono alla realizzazione dell'architettura aziendale, maggiore è il rischio che nel tempo alcuni segmenti diventino incompatibili. Ciò anche a causa della mancanza di standard di mercato ampiamente utilizzati (cfr. par. 4.6).

4.5 Degrado delle prestazioni

Il fattore critico risiede nella "lunghezza" del collegamento. Le transazioni effettuate via *WEB* seguono percorsi molto più ampi e tortuosi rispetto a quelle realizzate con architetture tradizionali. Inoltre più fitto è il colloquio tra *client* e *server*, più breve dovrebbe essere il percorso dei dati.

Un altro aspetto riguarda la scalabilità del numero degli utenti supportati, a cui non sempre corrispondono i livelli di transazioni, volumi e latenza desiderati. Applicazioni critiche con alto volume di transazioni non sono le migliori candidate per sistemi *WEB based* [Tod 02].

A questi problemi strutturali si affianca la difficoltà di *tuning* dei sistemi.

Il tema della verifica delle prestazioni degli ambienti *legacy* è stato da tempo affrontato e il mercato può ritenersi in una certa misura maturo, sia dal punto di vista degli strumenti a disposizione, sia del *know-how* delle aziende.

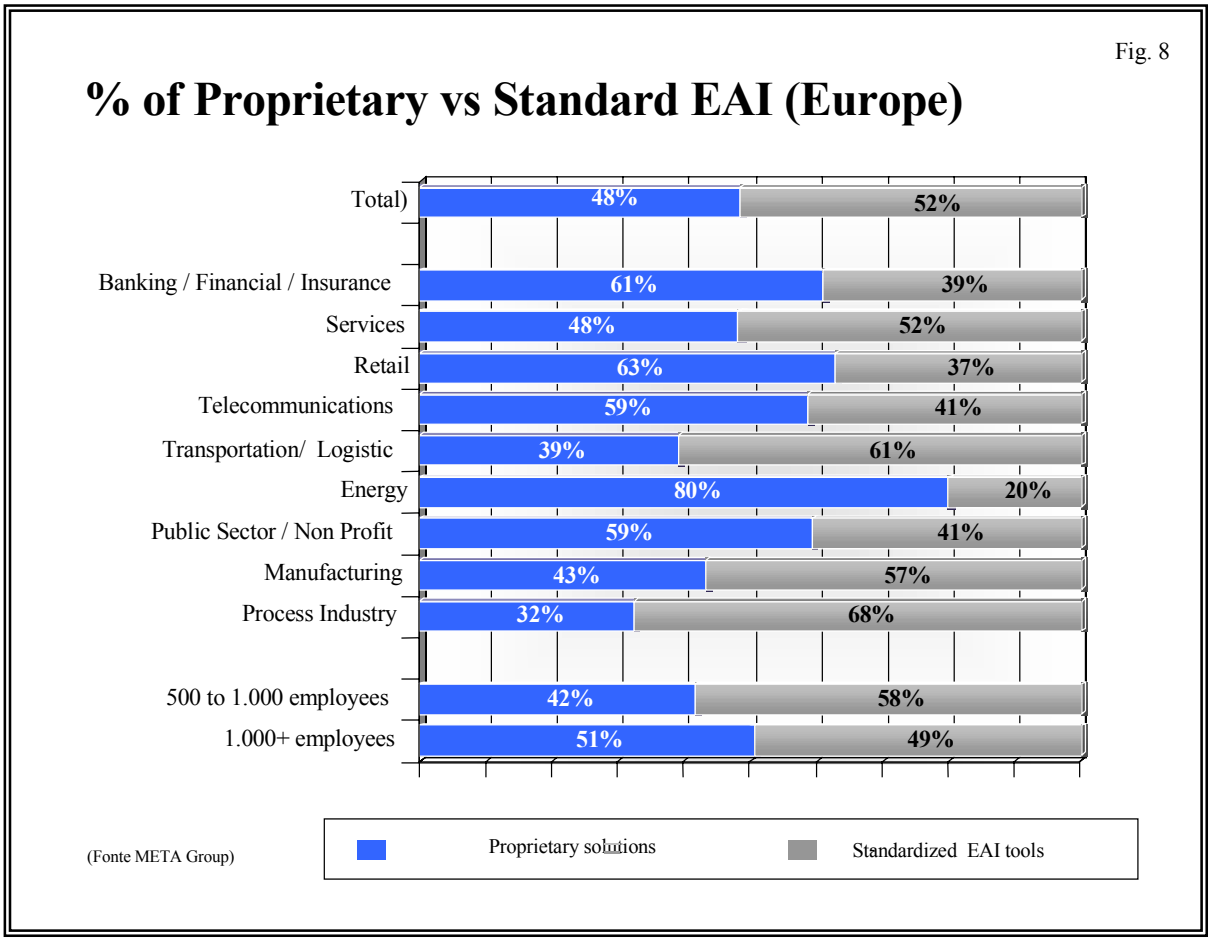
Individuare in un sistema *legacy* la componente responsabile di prestazioni degradate è, perciò, ragionevolmente semplice.

Le cose si complicano in un'architettura *WEB-legacy* a causa, fondamentalmente, della mancanza di strumenti che siano in grado di operare sull'intera architettura, piuttosto che su sezioni della stessa. Pertanto, è difficoltoso un approccio sistemico al problema delle prestazioni, con conseguente difficoltà a individuare le eventuali componenti critiche.

4.6 L'utilizzo di soluzioni proprietarie

Come visto in precedenza, esiste un elevato numero di prodotti e soluzioni per l'integrazione. Molto spesso, in media in più del 50% dei casi, i problemi di integrazione sono risolti attraverso percorsi non standard (cfr. Fig. 8).

Fig. 8



In questo senso, il mondo bancario non gode di una posizione di privilegio, poiché predilige soluzioni proprietarie. Nonostante ciò, è bene rammentare che la via del "fai da te" è, in un contesto in rapida evoluzione quale quello del *WEB*, estremamente rischiosa poiché la sola garanzia di stabilità che si può avere è quella derivante dall'adozione di standard di mercato.

4.7 "Webizzare" a tutti i costi

Infine, per quanto ovvio, considerati i costi e i rischi fin qui analizzati, è importante sottolineare che l'utilizzo della tecnologia *WEB* in un contesto tradizionale deve rispondere a effettive esigenze e non essere frutto di un'acritica adesione a una moda o a un orientamento generalizzato (o presunto tale).

Esistono situazioni in cui l'inserimento della tecnologia *WEB* nei sistemi *legacy* ha impatti puramente estetici, senza incidere minimamente sulla sostanza del servizio fornito. È pertanto importante che il livello di utilizzo di questa nuova tecnologia nel contesto aziendale sia valutato rispetto all'effettivo miglioramento che può comportare nei processi di *business*.

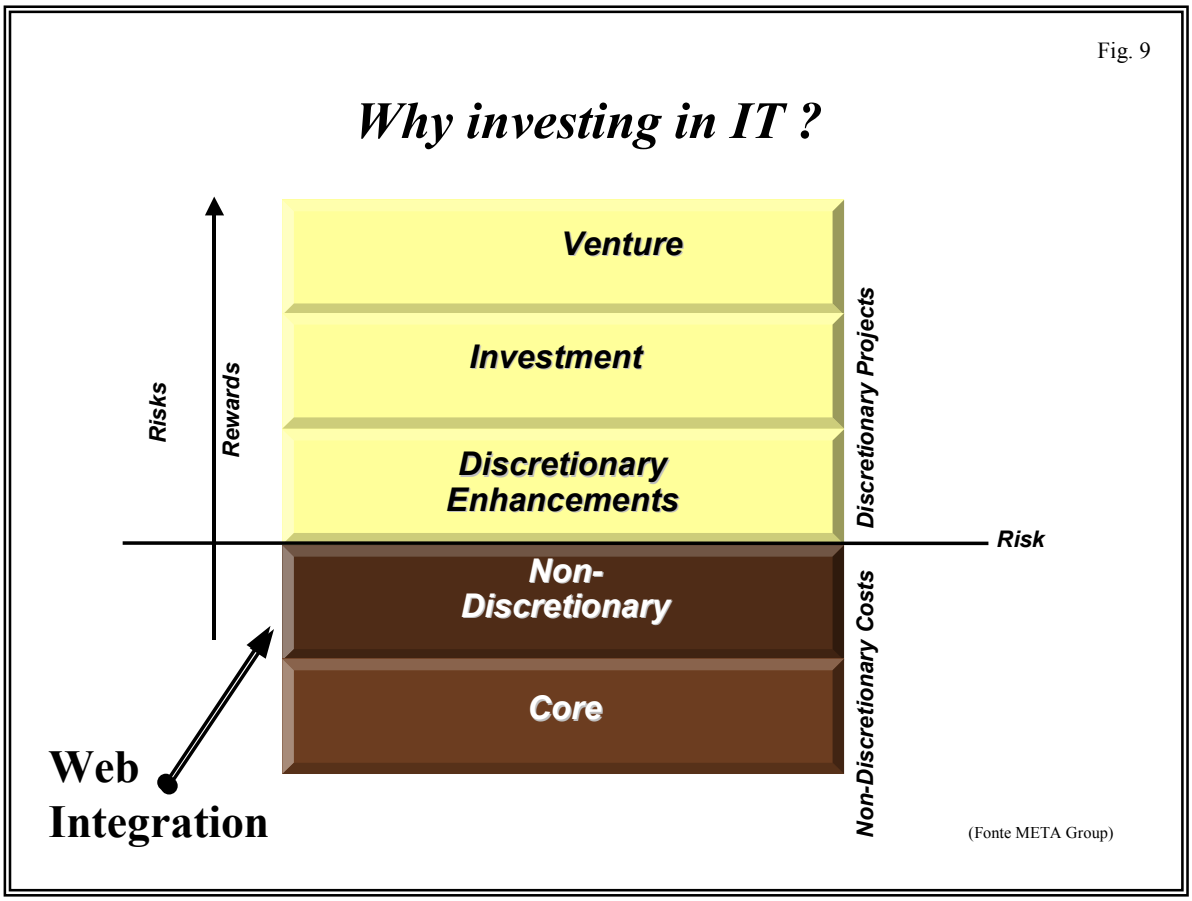
5. I costi dell'integrazione

L'integrazione di applicazioni *legacy* con il software connesso al *WEB* è spesso un'operazione costosa in termini di tempo e di risorse da impiegare: secondo alcune stime, il peso dell'integrazione può rappresentare fino a un terzo dell'ammontare totale di un progetto.

D'altro canto la tecnologia *WEB* ha oltrepassato la frontiera del *business* aziendale e la sua popolarità è oramai tale da non poter più essere ignorata nell'ambito delle visioni strategiche.

Ciò significa che gli investimenti sul *WEB,* e sulla sua integrazione con i sistemi esistenti, iniziano a essere collocati allo stesso livello di quelli relativi al *core business* (cfr. Fig. 9), pur variando di entità a seconda degli specifici obiettivi aziendali.

Fig. 9



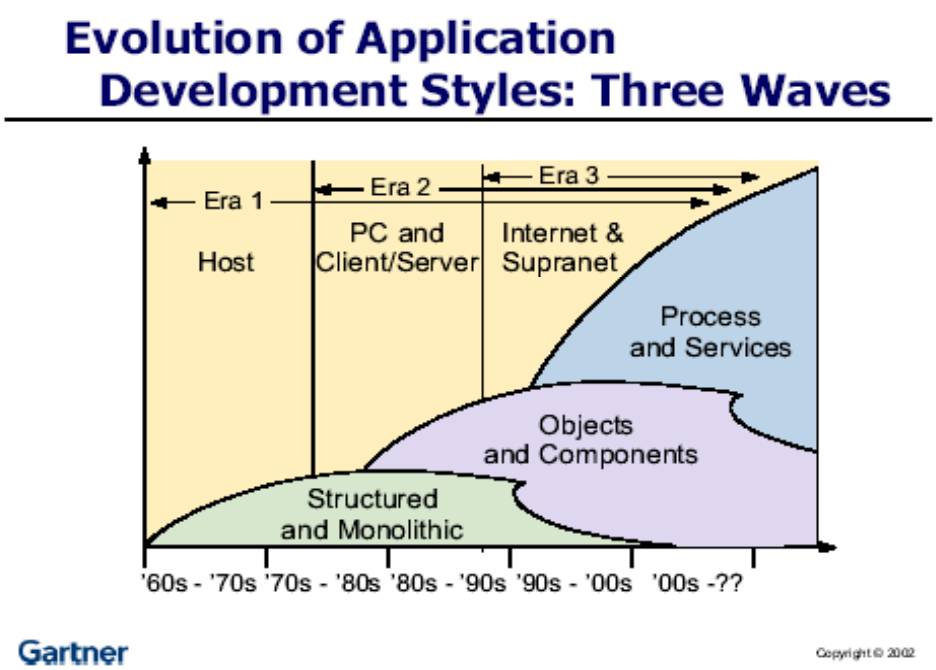
5.1 I nuovi paradigmi di sviluppo delle applicazioni

Un'assunzione strategica di Gartner [Smi 02] sostiene che entro il 2006 le figure professionali necessarie per lo sviluppo del 70% dei progetti applicativi dovranno essere formate sui processi piuttosto che sui linguaggi di programmazione.

Questa previsione è strettamente collegata alla necessità di migrare verso sistemi orientati ai servizi (cfr. par. 3.3). E proprio in questa direzione sta emergendo un nuovo approccio allo sviluppo dei sistemi informativi denominato *Service Oriented Development of Applications* (SODA).

Adottare SODA significa però cambiare radicalmente la modalità di realizzazione del software applicativo (cfr. Fig. 10). Non si deve infatti cadere nell'errore di considerare SODA alla stregua dello sviluppo per componenti, poiché questi ultimi non sono necessariamente dei servizi. Infatti i componenti sono realizzati con specificità tali (linguaggi, standard di interfaccia, etc.) da renderli strettamente dipendenti dai sistemi e/o dalle piattaforme di riferimento.

Fig. 10



Realizzare software con una metodologia SODA significa essenzialmente dover gestire:

- un *repository* dell'interfaccia fra i servizi;
- un motore di ricerca per l'individuazione dei servizi già esistenti;
- un ambiente di sviluppo delle applicazioni che consenta di mantenere distinte la progettazione dei servizi da quella dei componenti software.

Lo *skill* delle figure professionali coinvolte nello sviluppo si modifica sostanzialmente e questo, soprattutto in aziende di dimensioni medio-grandi, può comportare significativi costi organizzativi di formazione e conversione.

Il passaggio all'approccio SODA potrebbe essere comunque pianificato in un arco temporale che va al di là del breve-medio periodo. Ciononostante alcuni elementi innovativi devono essere presi in considerazione da subito. Primo fra tutti deve essere citato il linguaggio JAVA.

Nonostante in teoria l'integrazione fra sistemi *legacy* e tecnologia *WEB* possa essere realizzata attraverso strumenti e linguaggi di sviluppo tradizionali, di fatto il linguaggio JAVA diviene un *must*. Basti pensare che l'utilizzo di alcuni *application server* impone, attraverso gli ambienti di sviluppo compatibili, lo sviluppo di codice JAVA.

Anche in questo caso ci si trova di fronte a necessità di profili professionali molto diversi da quelli richiesti nello sviluppo tradizionale (ad es. COBOL).

In un recente studio [Vec 01] è stato ipotizzato che il costo per trasformare un programmatore COBOL in uno "professionale" JAVA sia circa 2,5 volte superiore a quello necessario per ottenerlo da un "colto" conoscitore di JAVA stesso. Per quanto riguarda il tempo di migrazione, invece, il programmatore COBOL richiederebbe circa 12 mesi, rispetto ai 10 mesi del programmatore Visual Basic e dei 5 mesi del programmatore C++.

Infine il rischio di fallimento sarebbe pari al 60% per il programmatore COBOL, contro il 40% di quello Visual Basic e il 5% di quello C++.

Tutto ciò dovrebbe far optare per un'acquisizione diretta di figure professionali JAVA rispetto alla conversione di programmatori COBOL. Ma, considerato che il principale elemento che inibirà almeno fino al 2004 l'utilizzo su ampia scala del linguaggio JAVA sarà proprio la mancanza di un numero sufficiente di figure professionali, la migrazione rappresenterà l'unica possibilità per molte aziende. E il costo di questa operazione non può essere sottovalutato.

5.2 L'acquisto di nuovi prodotti

Il costo della realizzazione di un progetto di integrazione, anche non particolarmente complesso, può agevolmente giungere, per le sole licenze d'uso, a 500.000 euro, a cui si aggiungono gli eventuali costi per l'hardware e quelli, successivi, per la manutenzione.

Di fatto i *tool* di integrazione sono ancora molto costosi e alla portata di poche aziende. La sfida su cui si confronteranno nel medio periodo i maggiori *vendor* sarà proprio quella di fornire soluzioni alla portata della media impresa. Una sfida non solo tecnologica, ma che investe anche la capacità di lavorare con terze parti (cfr. par. 4.1).

È pertanto opportuno che l'eventuale acquisto di strumenti di integrazione sia basato su un'analisi profonda supportata da un modello di valutazione. In questo senso potrebbe essere utile fare riferimento allo schema di valutazione proposto da META Group che considera le seguenti sei variabili fondamentali:

- potenzialità delle funzionalità di integrazione;
- gestione della sicurezza;
- disponibilità di funzioni di gestione del processo (es. workflow);
- aderenza agli standard di mercato;

- disponibilità di un ambiente per lo sviluppo di ulteriori funzionalità;
- qualità dell'architettura.

5.3 La modifica dei sistemi legacy

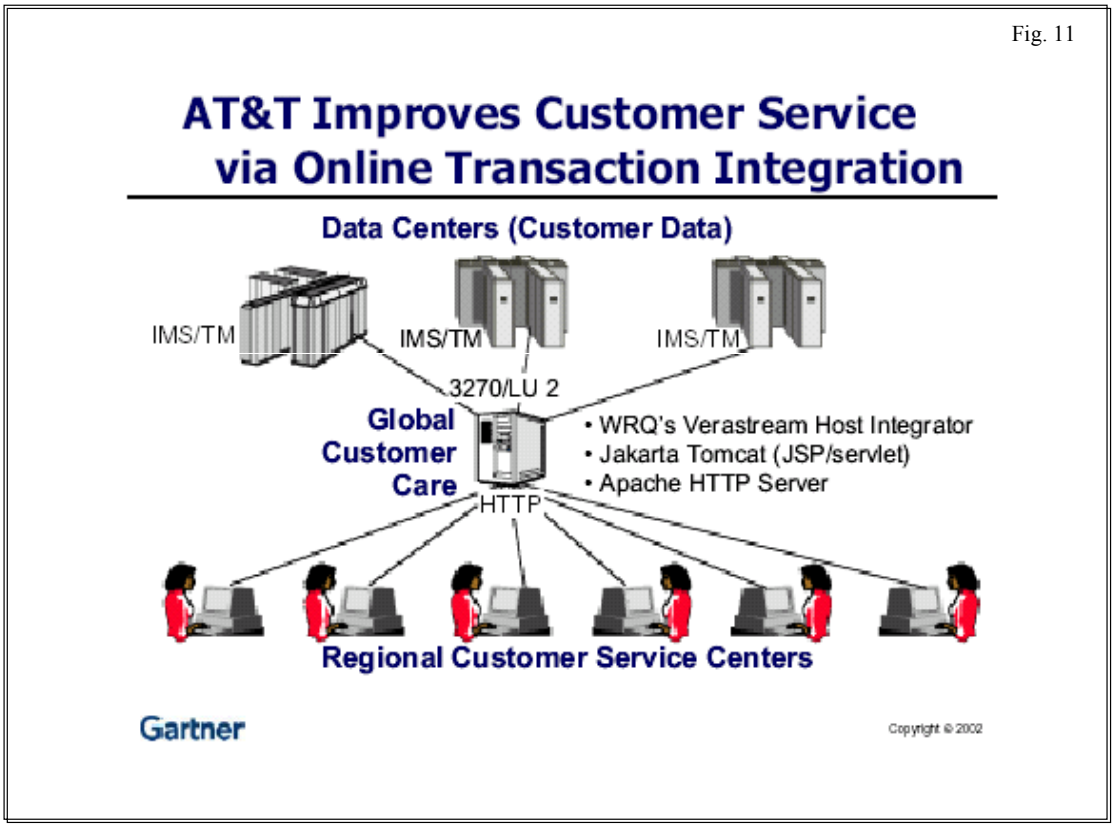
Uno dei fattori che influenza in modo determinante il costo di un progetto di integrazione è la dimensione delle modifiche che si rendono necessarie sui sistemi preesistenti. Infatti (cfr. cap. 2), a seconda della struttura del sistema *legacy* sul quale si deve operare, si può andare da interventi praticamente nulli (caso di sistemi altamente decomponibili) fino ad attività di manutenzione confrontabili con il rifacimento del sistema stesso (caso dei sistemi monolitici).

Ma ciò non dipende solo dalla struttura del *legacy*. L'altra variabile fondamentale è la tipologia di integrazione che si vuole realizzare (cfr. par. 3.2). Allo scopo di fornire alcuni riferimenti quantitativi si possono considerare due casi aziendali di progetti di integrazione con caratteristiche e costi completamente diversi.

Il primo caso è quello della società telefonica americana AT&T che ha circa 1.100 dipendenti, suddivisi su sei centri regionali, dedicati al supporto per le informazioni ai clienti. Per rispondere ad alcune richieste dei propri clienti, l'operatore doveva accedere a tre diverse applicazioni residenti su tre distinti elaboratori *mainframe*, attivare diverse transazioni, prendere nota di alcuni dati significativi e, infine, richiamare l'utente per fornire la risposta. Il tempo di completamento dell'intero processo andava da un minimo di sette minuti fino a un massimo di quindici.

La AT&T è stata in grado di raggiungere tempi di risposta di circa dieci secondi attraverso un processo di unificazione dell'interfaccia di accesso ai sistemi realizzato tramite la tecnologia *WEB* (cfr. Fig. 11).

Fig. 11



Più precisamente, l'operatore accede tramite *browser* a un'applicazione JAVA che è in grado di collegarsi ai diversi sistemi *mainframe* e di riportare in una pagina *WEB* le informazioni necessarie per fornire la risposta al cliente. Il progetto è stato realizzato in sei settimane con una spesa di circa 80.000 euro e un ritorno dell'investimento praticamente immediato attraverso un significativo miglioramento del grado di soddisfazione dell'utente finale.

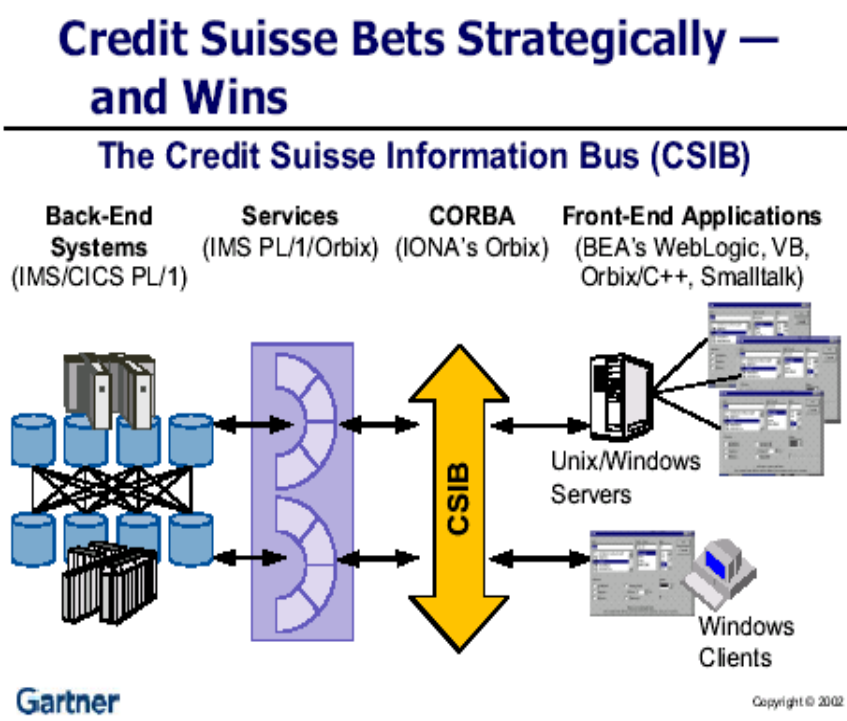
Un altro esempio è quello che ha visto coinvolto Credit Suisse, il cui sistema informativo è la risultante di un insieme di fusioni e di adattamenti dovuti alle varie ristrutturazioni organizzative che la banca ha subito nel tempo.

Nel 1997 Credit Suisse ha deciso di realizzare una SOA partendo dal parco applicativo che consisteva in una serie di applicazioni (80% IMS e 20% CICS) operanti su due mainframe S/390 (cfr. Fig. 12). È stata così creata un'infrastruttura, attivata nel maggio del 1999, basata su circa 500 servizi applicativi, ottenuti come aggregazione e composizione di una o più delle transazioni IMS e CICS precedenti, condivisi da più di 50 *front-end* applicativi, realizzati a loro volta attraverso l'uso di tecnologie quali J2EE, C++ e Visual Basic.

Il progetto è stato estremamente impegnativo con una significativa componente di *re-engineering* delle applicazioni *legacy*. Il costo complessivo dell'operazione è stato di circa 10 milioni di euro.

Il beneficio più importante che Credit Suisse ha ottenuto, oltre a una razionalizzazione del parco applicativo e alla possibilità di sfruttare la tecnologia *WEB* per raggiungere i circa 100.000 utenti (fra dipendenti e clienti), è stato quello di una drastica riduzione dei costi di sviluppo, poiché il 75-80% dei servizi richiesti è già disponibile. Di fatto, ci troviamo di fronte alla piena implementazione dell'approccio SODA (cfr. par. 5.1).

Fig. 12



6. Linee guida per gestire il processo di integrazione

In questa sezione, riprendendo anche parte dei contenuti di una recente pubblicazione di Gartner [Tho 01], viene fornita un'indicazione degli "ingredienti" necessari per impostare un'architettura tale da consentire la gestione nel tempo del processo di integrazione.

Regola 1: agevolare l'avvio del processo di integrazione tramite un centro di competenza; è importante costituire, almeno inizialmente, un *team* di progetto specializzato che abbia il compito di disegnare, realizzare, rendere operativa e far evolvere l'infrastruttura di integrazione dell'azienda. Il centro di competenza deve, inoltre, definire una politica di integrazione coerente con gli obiettivi aziendali e fornire servizi di consulenza che facilitino il lavoro delle unità deputate allo sviluppo e alla manutenzione delle applicazioni. Le conoscenze di questo *team* dovrebbero divenire nel tempo patrimonio comune dell'azienda ed entrare nelle procedure di lavoro condivise.

Regola 2: definire una "mappa" aziendale dell'integrazione; lo sviluppo di un'architettura non può mai essere improvvisato. É necessario definire e comprendere precisamente, attraverso un modello concettuale, gli obiettivi di *business*, organizzativi e tecnici.

Regola 3: tenere presente sia il B2B (Business to Business) sia lo A2A (Application to Application); B2B e A2A sono due varianti dello stesso problema: collegare due sistemi per rendere possibile lo scambio di dati e messaggi. Generalmente, però, le aziende affrontano il problema della definizione di un'architettura di integrazione, enfatizzando il collegamento dei propri processi di *business* con l'esterno (B2B). La comunicazione fra sistemi all'interno dei confini aziendali (A2A) è però altrettanto importante e le scelte architetturali dovrebbero tenerne conto.

Regola 4: documentare le interfacce fra le applicazioni; in un'azienda di grande dimensione, le interfacce fra le applicazioni possono essere anche alcune centinaia. È pertanto opportuno archiviare in un *repository* tutte le interfacce per facilitarne l'eventuale riutilizzo, che dovrebbe essere uno dei principali obiettivi di una efficace gestione dell'architettura di integrazione.

Regola 5: un solo prodotto può non essere sufficiente; molto spesso il primo passo di un'azienda verso l'integrazione è la scelta della "migliore" piattaforma. Ogni prodotto ha i suoi pregi e i suoi difetti e difficilmente può dare risposta a tutti i requisiti di integrazione di un'azienda. Generalmente un'infrastruttura di integrazione è realizzata utilizzando prodotti di più *vendor* con l'aggiunta, eventuale, di sviluppi "ad hoc".

Regola 6: scegliere i prodotti sulla base delle strategie aziendali; i prodotti per l'integrazione possono sembrare a prima vista simili, ma hanno, invece, significative differenze. La scelta deve essere quindi fatta sulla base di requisiti dettagliati che consentano di apprezzare le diversità degli strumenti.

Regola 7: creare competizione fra i fornitori; le piattaforme per l'integrazione delle applicazioni sono spesso molto costose. Un progetto di integrazione non molto complesso ha un costo medio di 500.000 euro relativo alle sole licenze software, escludendo, quindi, gli eventuali costi per l'hardware e per la successiva manutenzione. La trattativa commerciale è pertanto importante e deve essere condotta in un contesto di concorrenza.

Regola 8: sperimentare i prodotti prima di acquistarli; considerate le similitudini e le specificità dei vari prodotti e il loro elevato costo, la scelta deve essere fatta sulla base di prove dal "vivo" sui prodotti stessi.

Regola 9: scegliere il "partner" giusto; oltre ad essere costose, le piattaforme per l'integrazione sono anche molto complesse. La capacità di un *vendor* di fornire servizi e assistenza altamente qualificata sul prodotto è, pertanto, un fattore critico di successo dell'intero progetto. Alcuni produttori di questo settore di mercato hanno dimensioni limitate e sono in una fase di crescita "esplosiva". É quindi necessario avere garanzie tangibili sull'effettiva disponibilità, da parte del fornitore, di risorse in numero e *skill* tali da assicurare il miglior supporto possibile al potenziale cliente.

Regola 10: sviluppare l'architettura con un approccio incrementale; come sempre, infine, è consigliabile procedere per gradi nella fase realizzativa, sulla base, però, di una strategia di riferimento ben definita che faccia riferimento agli standard di mercato.

7. Uno sguardo al futuro

7.1 I WEB services

7.1.1 Cosa sono

Il termine *WEB services* è spesso utilizzato in accezioni differenti. Semplificando, i *WEB services* possono essere analizzati da tre visuali differenti:

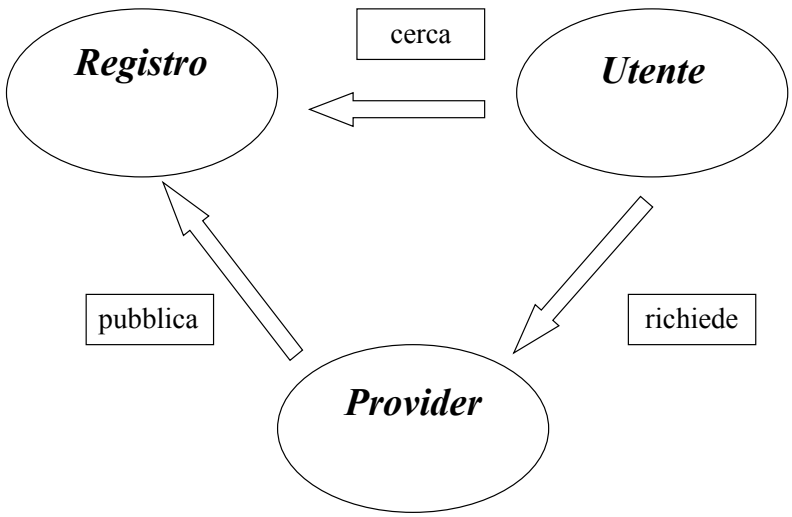
- tecnologica (fisico);
- architetturale (logico);
- di servizio (concettuale).

Concettualmente i *Web services* sono un mezzo per rendere disponibile, indipendentemente da qualsiasi piattaforma tecnologica, la logica applicativa dei software aziendali sotto forma di componenti software. In questo senso, non sono strettamente correlati alle nuove tecnologie, poiché una qualsiasi vecchia applicazione COBOL può essere definita come *WEB service.*

Questi componenti possono essere integrati o aggregati in applicazioni di *business* più complesse o in processi di *business*. L'obiettivo è quello di mettere insieme dinamicamente *Web services* situati nei siti più disparati per creare l'infrastruttura necessaria per le attività di commercio *business to business*.

Da un punto di vista architetturale, i *WEB services* rappresentano semplicemente un'istanza della *Service Oriented Architecture* (cfr. par. 3.3 e Fig. 13).

Fig. 13



A livello tecnico, infine, non sono altro che un nuovo tipo di tecnologia software, basata su specifici standard, che permette ai programmatori di combinare in modo nuovo i sistemi informativi, sia della propria azienda, sia delle aziende *partner*.

In questo senso, la definizione più semplice è quella fornita da Sridhar Iyengar, membro dell'OMG Architecture Board [Sri 01], secondo cui i *WEB services* sono applicazioni modulari e riusabili che possono essere:

- richiamate sul *Web* tramite il protocollo *Simple Object Access Protocol* (SOAP)[3];
- descritte tramite uno specifico linguaggio, detto *Web Services Description Language* (WSDL);
- pubblicate in un registro (*Universal Description Discovery and Integration* - UDDI).

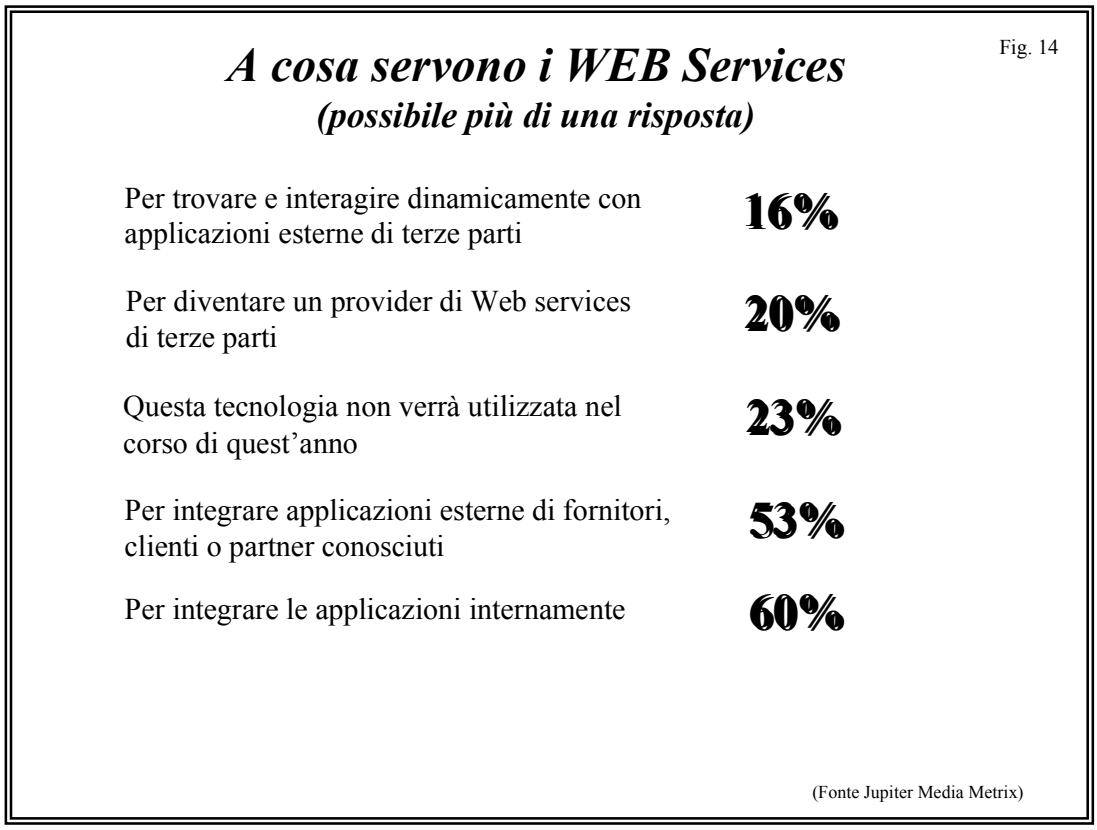
Indipendentemente da come li si vuole inquadrare, i *WEB services*, in prospettiva, dovrebbero permettere alle organizzazioni di colmare le lacune di comunicazione tra applicazioni scritte con linguaggi diversi tra loro, oppure sviluppate da aziende diverse, oppure in esecuzione su ambienti elaborativi eterogenei.

[3] Il protocollo SOAP definisce soltanto il formato del messaggio che deve essere "trasportato", mentre il protocollo di trasporto vero e proprio è normalmente rappresentato dall'HyperText Transport Protocol (HTTP).

7.1.2 Gli ambiti di utilizzo

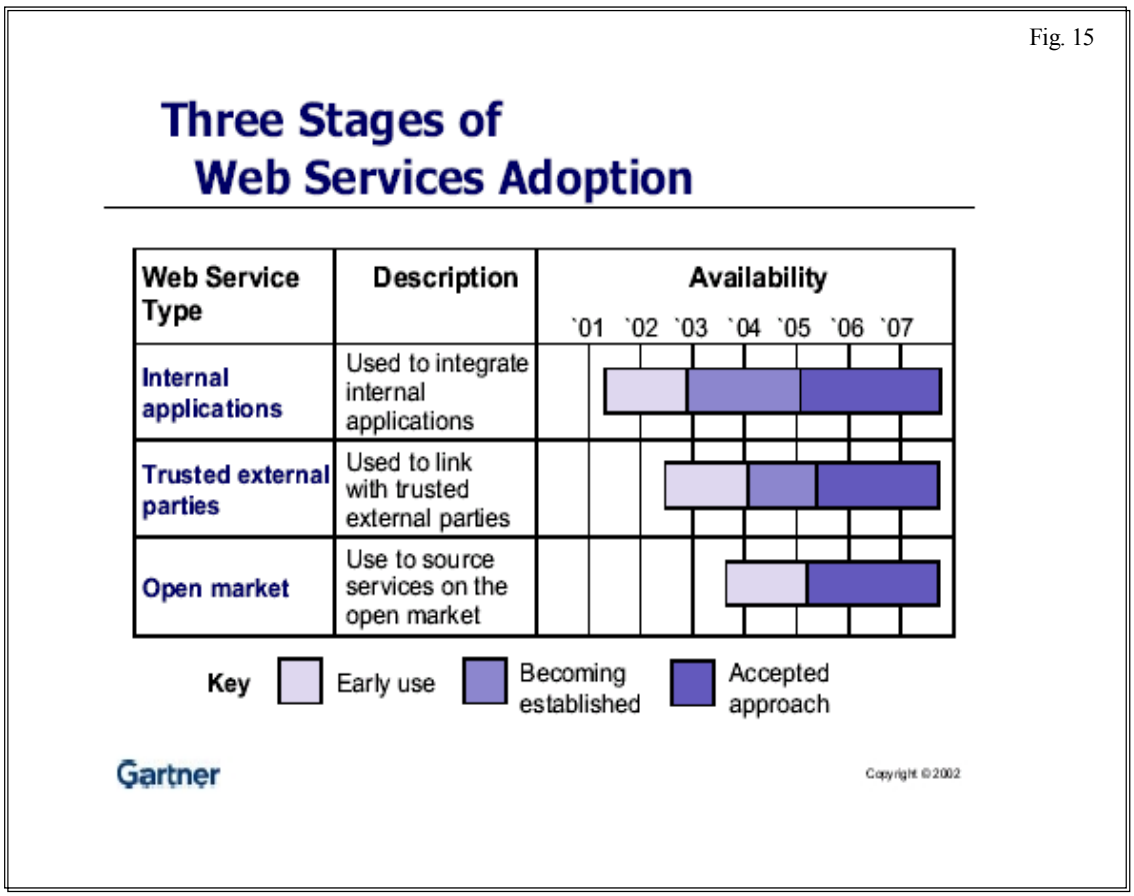
Una recente ricerca condotta presso 471 responsabili dei sistemi informativi di società degli Stati Uniti (cfr. Fig. 14) ha confermato che le aziende iniziano a utilizzare i *WEB services* partendo generalmente da progetti interni di integrazione, poiché sono ritenuti meno rischiosi e più facili da collaudare.

Fig. 14



Alcuni pionieri hanno già iniziato a sviluppare applicazioni esterne che utilizzano questi servizi. Occorreranno però almeno altri 12-18 mesi prima che si inizino a realizzare *WEB services* in grado di oltrepassare i *firewall* aziendali (cfr. Fig. 15).

Fig. 15



7.1.3 Uno strumento per l'integrazione tra WEB e legacy

Una stima di Gartner [Nat 01] ritiene altamente probabile che la maggior parte del *WEB commerce* sarà supportato, entro il 2006, da software appartenente alla tecnologia dei *WEB services*.

In generale, comunque, i *WEB services* promettono di agevolare le aziende nell'integrazione del software, riutilizzando ciò che loro stesse o altri hanno prodotto.

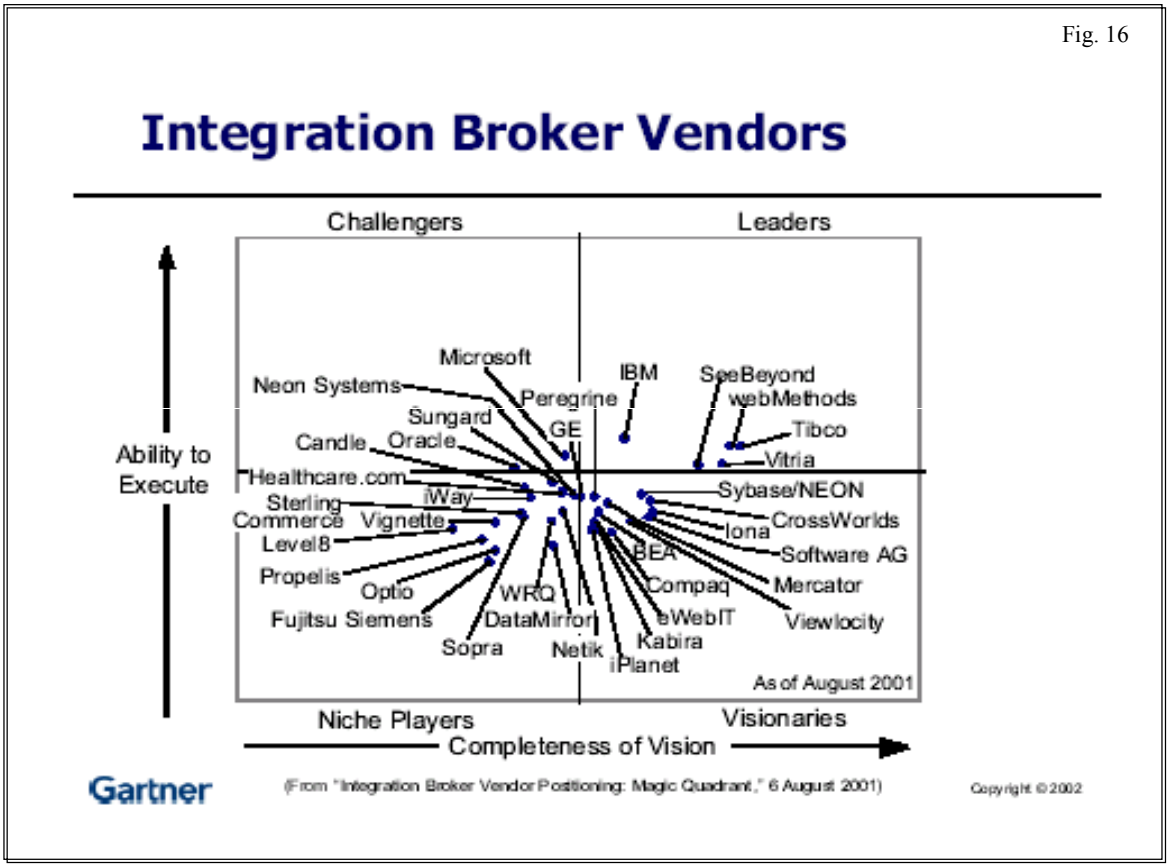
In questo senso un ruolo non secondario spetterebbe proprio al problema della collocazione sul *WEB* di applicazioni tradizionali (c.d. *legacy wrapping)*.

Infatti, sebbene siano disponibili tecnologie per l'integrazione offerte da almeno 30 fornitori fra i più rappresentativi del mercato dell'ICT (cfr. Fig. 16), difficilmente la soluzione al problema è ottenibile soltanto tramite l'utilizzo di prodotti commerciali.

Molto spesso i progetti di integrazione richiedono lo sviluppo di software *ad hoc*, generalmente molto costoso. Ciò è essenzialmente dovuto al fatto che gli *adapter* realizzati dai differenti *vendor* non sempre rispettano degli standard universali.

La tecnologia dei *WEB services* consente di rendere standard il processo di *wrapping* delle applicazioni *legacy* in un formato completamente riutilizzabile e, pertanto, di eliminare progressivamente dal portafoglio applicativo dell'azienda gli *adapter* di tipo proprietario e/o quelli sviluppati internamente [Met 01].

Fig. 16



7.1.4 Non solo luci .... anche qualche ombra

Sebbene i *WEB services* siano una tecnologia molto promettente e accattivante, non si devono nascondere alcuni problemi ancora aperti e dovuti, probabilmente, alla giovane età.

L'incognita maggiore riguarda la sicurezza. A oggi non esiste uno standard, sebbene il *World Wide Web Consortium* (W3C) stia lavorando per risolvere il problema. Inoltre, occorre rilevare che il trasporto HTTP non offre garanzie di *delivery* tali da resistere a *failures* di rete, delle applicazioni o dei *server* stessi, inserendo di fatto un potenziale elemento di criticità nelle applicazioni tipicamente dispositive[4].

[4] Supponiamo di avere un'applicazione Web composta da:
- un Client Web Browser;
- un Web Application server con un'applicazione che accede a un database o che innesca (via MQ o connector) un'applicazione su un sistema legacy os/390.

Il flusso di richiesta di un servizio potrebbe pertanto essere il seguente:
1. il client innesca la richiesta di web services via SOAP con protocollo HTTP;
2. il web services innesca l'applicazione host che esegue l'attività e prepara il messaggio di risposta;
3. il web services inoltra la risposta via HTTP ma il client potrebbe non essere più attivo o raggiungibile. In questa situazione la risposta viene "persa" e dovrebbero essere attivate delle opportune attività (inquiry applicativo o analisi dei log) per verificare l'esito dell'operazione.

Particolare attenzione quindi deve essere posta alla tipologia di servizi che vengono messi a disposizione e alla descrizione delle procedure necessarie alla gestione degli errori.

La totale affermazione dei *WEB services* si potrà avere soltanto quando si abbandoneranno le soluzioni proprietarie e ci sarà sicurezza sull'identità del sistema connesso, sul fatto che il messaggio sia stato inviato una volta (e solo una) e che tutto il processo di *business* sia completato.

C'è infine da sottolineare che i *WEB services*, benché rendano più semplici gli aspetti tecnici di individuazione, attivazione e integrazione tra le varie componenti applicative, non forniscono, al momento, alcun contributo per la semantica dei servizi disponibili. A questo problema la risposta sarà data probabilmente dalla creazione di ulteriori standard di definizione di più alto livello logico.

7.2 La Model Driven Architecture

7.2.1 Cosa è

L'obiettivo di rendere possibile alle aziende di realizzare sistemi informativi integrabili con il passato, il presente e il futuro è stato all'origine della nascita, nel 1989, *dell'Object Management Group* (OMG).

Questo obiettivo è stato perseguito, fino a poco tempo fa, sviluppando versioni sempre più complete dello standard CORBA (cfr. par. 3.1). Ma, come ha rilevato Fred Waskiewicz, direttore degli standard dell'OMG, "CORBA è stato un primo passo importante, ci sono però molti altri passi da fare".

Il problema più rilevante che le aziende devono oggi affrontare è quello della difficoltà, forse dell'impossibilità, di definire un'unica piattaforma di *middleware*. Per alcune la numerosità di piattaforme è causata dai differenti requisiti dei diversi settori interni, per altre la motivazione va cercata nel *mix* di tecnologie adottate. Comunque, anche nell'ipotesi più favorevole, il problema nasce quando si vuole interagire con altre *company* o con i *B2B market* [Sol 00].

In mancanza di uno standard universale, le aziende devono prendere atto che dispongono di applicazioni (sia interne, sia esterne) che fanno riferimento a numerosi *middleware* e che bisogna "semplicemente" farle comunicare, sebbene ciò implichi un'attività complessa e dispendiosa.

Il metodo che l'OMG propone per gestire questo problema è la *Model Driven Architecture* (MDA).

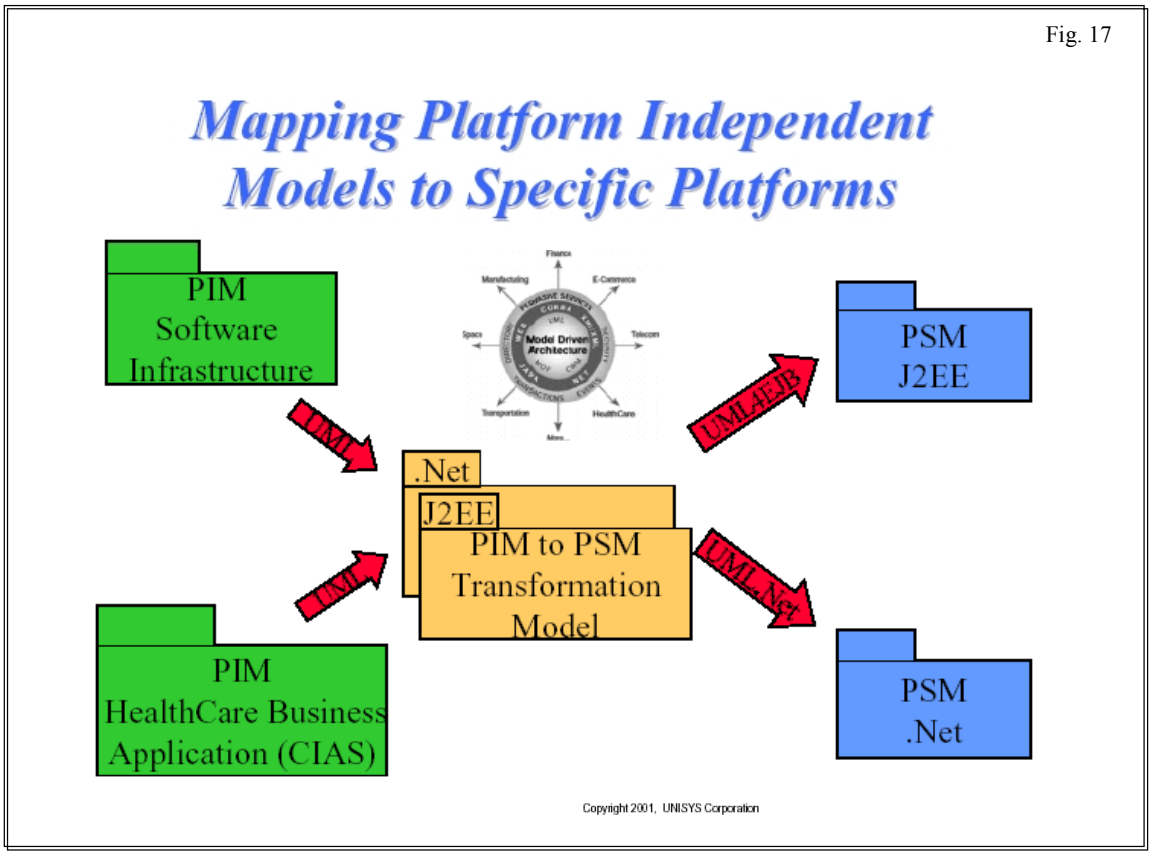
Se CORBA era un tentativo di rendere interoperabili le diverse piattaforme, MDA punta a rendere interoperabili le applicazioni, utilizzando, però, una tecnologia di modellazione.

In altre parole, la MDA è una nuova modalità di definizione e sviluppo delle applicazioni, basata su un modello indipendente dalla piattaforma (*Platform Independent Model* – PIM o modello base), che permette di "generare" le applicazioni stesse su qualsiasi piattaforma *middleware* coerente con il modello base. Pertanto il PIM ha soltanto funzionalità e caratteristiche di *business* non influenzate da aspetti o considerazioni sulla tecnologia.

L'indipendenza tecnologica permette al modello base di essere valido per diversi anni e di necessitare di modifiche soltanto al variare delle condizioni di *business*.

La proiezione di un modello base su una specifica piattaforma (*target*) porta a un modello di secondo livello definito *Platform Specific Model* – PSM (cfr. Fig. 17).

Fig. 17



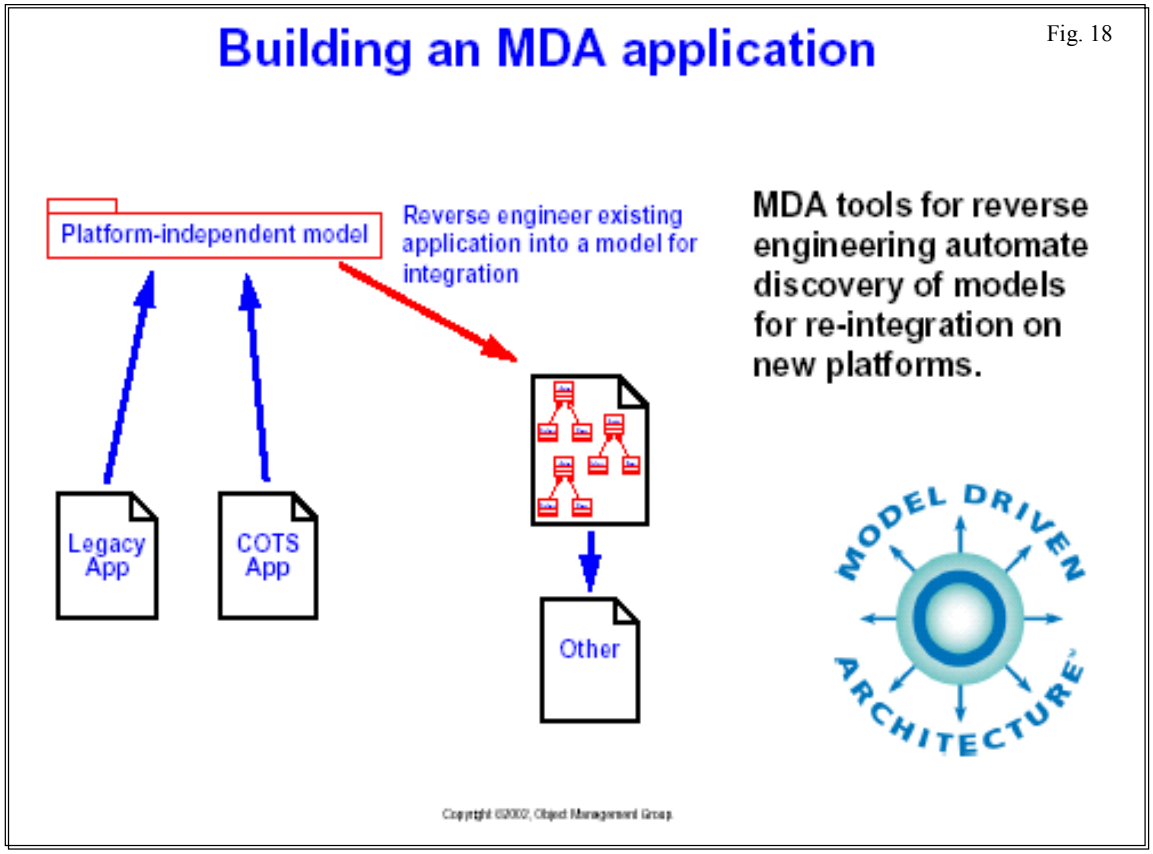
Le tecniche di modellazione di supporto alla definizione di una Model Driven Architecture sono le seguenti [Sie 01], basate sullo *Unified Modeling Language* (UML):

- **Meta-Object Facility (MOF)**, che fornisce un *repository standard* per il modello e una struttura di ausilio all'utilizzo concorrente di più gruppi di lavoro del modello stesso;
- **Common Warehouse Model (CWM)**, che costituisce lo standard per la rappresentazione di modelli di database (schema), modelli di trasformazione degli schemi e modelli per il *data mining*;
- **XML Metadata Interchange (XMI)**, che permette la trasformazione in XML di modelli rappresentati in UML.

7.2.2 L'utilità per l'integrazione tra WEB e legacy

Ovviamente la MDA dovrebbe fornire anche la soluzione ai problemi di integrazione dei sistemi tradizionali con la tecnologia *WEB*. Infatti, l'obiettivo finale della Model Driven Architecture è quello di risolvere qualsiasi problema di integrazione e comunicazione fra applicazioni attraverso la produzione automatica dei PSM a partire dai modelli indipendenti dalla piattaforma.

Ma ancora di più. Negli intenti, la MDA dovrebbe mettere a disposizione anche dei modelli di trasformazione per derivare la PIM da un sistema esistente, anche *legacy*, per eventualmente riproiettarla su una piattaforma specifica, ad esempio quella *WEB* (cfr. Fig. 18).



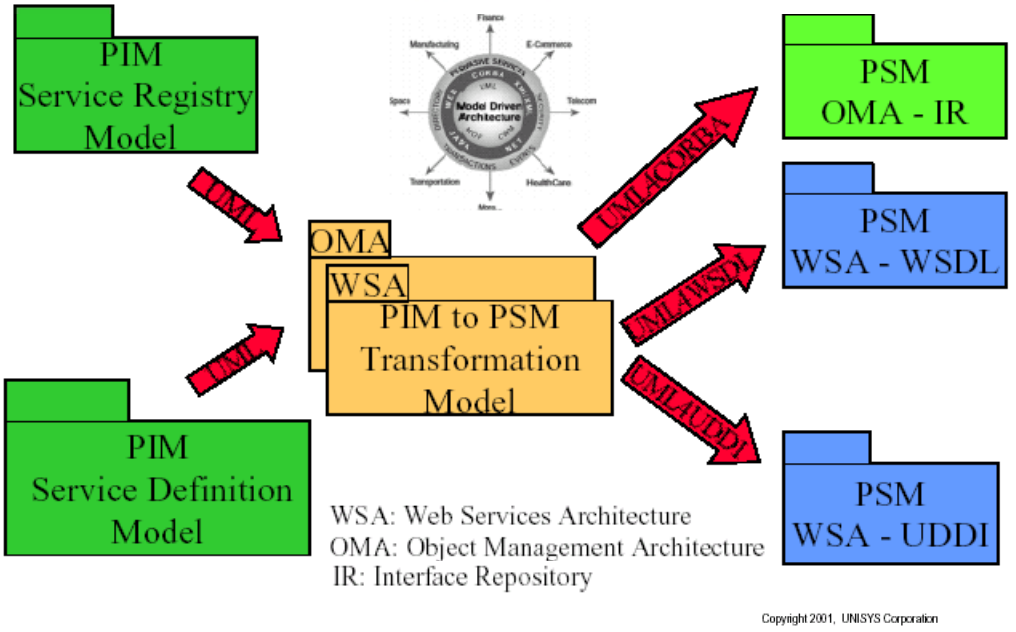
D'altro canto il completamento della MDA si avrà soltanto quando saranno realizzati e commercializzati dei *tool* che consentano di automatizzare tutte le trasformazioni a cui si è fatto cenno.

In quel momento, sfruttando anche le tecniche di progettazione SODA (cfr. par. 5.1), si potranno realizzare sistemi informativi orientati ai servizi e indipendenti dalla piattaforma tecnologica [Bur 00].

Da tutto ciò consegue che la MDA va pianificata nel lungo periodo, nell'ambito di un approccio strategico al problema dell'integrazione. Nonostante ciò è importante segnalare che attualmente l'OMG sta lavorando alla definizione dei modelli di trasformazione per la

generazione dei *WEB services* a partire da una descrizione in linguaggio di alto livello dell'architettura (cfr. Fig. 19).

Fig. 19



8. Tattica o strategia ?

Da quanto esposto nei capitoli precedenti emerge la disponibilità di metodi, tecniche e strumenti per operare, rispettivamente, nel breve, nel medio e nel lungo periodo. Di fatto, il problema dell'integrazione fra tecnologie *WEB based* e applicazioni tradizionali lascia aperto il campo a scelte sia tattiche, sia strategiche.

Ovviamente non esiste "la scelta ideale" poiché, come sempre, ogni decisione deve essere valutata sulla base delle caratteristiche e peculiarità della singola azienda.

Ciò posto, sono sostanzialmente tre i possibili approcci all'integrazione [Scu 02]:

- eliminare;
- incapsulare;
- accettare alcune singolarità.

La prima modalità è quella più radicale. I "vecchi" sistemi vengono eliminati e sostituiti con un nuovo insieme di applicazioni progettate e realizzate con le necessarie connessioni al *WEB*. Tutto va bene se l'azienda ha le risorse finanziarie e il tempo necessari. In realtà, come recentemente

riportato da Roy Schulte (vice presidente di Gartner), il tasso di insuccesso registrato nei casi in cui si è proceduto in questo modo è superiore al 75%.

È comunque importante notare che questo approccio può diventare conveniente nei casi in cui lo stato delle applicazioni sia tale (cfr. cap. 1 – Sistemi monolitici) da richiedere interventi di manutenzione di grossa entità per integrare le applicazioni stesse con la tecnologia *WEB*[5].

Un approccio meno estremo è quello che persegue l'integrazione con il *WEB* attraverso la tecnica del *wrapping*. Le funzioni e i dati di un'applicazione sono "incapsulate" in componenti di *business* accessibili come servizi tramite *WEB*. Una modalità di realizzazione di questo approccio è quella che prevede l'utilizzo dei *WEB services*.

Questa soluzione ha una connotazione tattica piuttosto che strategica. Permette di risolvere velocemente alcuni problemi di integrazione ma, a lungo andare, comporta un aumento della logica applicativa, rappresentata essenzialmente dalla necessità di descrivere le regole di utilizzo del servizio[6].

Una strada che media tra i due approcci appena descritti è quella che accetta alcuni livelli di eterogeneità. D'altro canto l'evoluzione della tecnologia è oggi talmente rapida che sarebbe di fatto impossibile voltare pagina per ottenere, da quel momento in poi, un'architettura costantemente uniforme.

Ciò non vuol dire che qualche applicazione non debba essere sostituita (eliminare) perché tecnologicamente obsoleta o che in alcuni casi non si possa utilizzare un *WEB service* (incapsulare) per proporre sull'*Intranet* aziendale una vecchia transazione CICS.

Accettare l'eterogeneità, però, significa convivere, in alcuni casi, anche con la ridondanza dei dati[7] e, pertanto, mettere a punto procedure di documentazione e di riconciliazione più frequenti e certificate.

In questo senso, assumono rilevanza strategica (cfr. cap. 6 – Regola 1 e 2):

- la costituzione di un centro di competenza per l'integrazione che gestisca l'ENS aziendale[8]. Questo *team* ha una composizione che, nella maggior parte dei casi, va da un minimo di 4 persone a un massimo di 8. La componente operativa di questo *team* ha il compito di selezionare, installare e mantenere il software dell'infrastruttura di integrazione. La componente di sviluppo gestisce il modello logico aziendale dell'integrazione e ha il compito di fornire consulenza ai singoli gruppi di progetto per la realizzazione degli specifici *adapter*;
- la definizione di una mappa dei sistemi informativi aziendali (*Enterprise Nervous System*) che rappresenti la guida per ogni decisione inerente all'integrazione, sia che si tratti di sostituire un sistema informativo con un ERP, sia che si tratti di "webizzare" un'applicazione *legacy* portando su *WEB* la sola componente di interfaccia di una procedura COBOL.

[5] In generale, si può assumere che convenga realizzare una nuova applicazione quando la vecchia procedura richiede modifiche superiori al 30% per l'adeguamento ai nuovi requisiti di business.

[6] A ogni modifica della logica dell'applicazione di partenza deve corrispondere un cambiamento delle specifiche descrittive del servizio basato sull'applicazione stessa.

[7] Un'assunzione strategica di Gartner stabilisce che nel 2005 la ridondanza dei dati e la duplicazione delle funzioni sarà superiore a quella odierna [Scu 02].

[8] Gartner sostiene che entro il 2005 più del 50% delle grandi aziende avrà un centro di competenza aziendale per l'integrazione, rispetto all'attuale 25% [Scu 02].

9. Glossario

B2B
Il B2B (*business to business*) è la gestione elettronica di tutte le principali attività dell'azienda, che coinvolge anche tutti i partner, fornitori e clienti dell'azienda stessa.

Browser
Applicazione software basata su un'interfaccia GUI che consente di visualizzare le pagine HTML di *Internet* e del WWW.

COM
Component Object Model. Si tratta di una serie di specifiche sviluppate da Microsoft per componenti software che possono essere inseriti nei programmi o per aggiungere funzionalità ai programmi esistenti che vengono eseguiti su piattaforme Microsoft Windows.

CORBA
Common Object Request Broker Architecture. Architettura di sviluppo software che consente di creare oggetti utilizzabili in ambienti diversi, consentendo una completa interoperabilità fra software, senza riguardo ai sistemi operativi o ai processori utilizzati. Utilizzato anche per la creazione di siti Internet. Gli standard CORBA sono definiti dal OMG. Il nucleo centrale di CORBA è ORB.

DCOM
Distributed Component Object Model. Modello per componenti distribuiti; è basato sulle specifiche COM e introduce alcune caratteristiche per semplificare l'utilizzo di componenti distribuiti in una LAN o in Internet. È uno standard proprietario di Microsoft, alternativo a CORBA.

E-business
Electronic Business. Controllo automatizzato di tutti i processi aziendali attraverso cui è possibile monitorare costantemente ogni tipo di attività (comprare, vendere, fornire assistenza on line, consultare listini e cataloghi in rete, etc.). Il termine fu introdotto per la prima volta nel 1997 da IBM.

E-commerce
Electronic Commerce. Il "commercio elettronico", cioè la possibilità di acquistare prodotti e servizi on line, attraverso il World Wide Web.

EDI
Electronic Data Interchange. Insieme di standard utilizzati per controllare il trasferimento di documenti d'affari tra sistemi informativi. Per poterlo applicare, gli utenti devono accordarsi sulle modalità da adottare per scambiare informazioni. L'EDI, esistente fin dagli anni '70, può essere considerato il precedente storico delle attività di *e-commerce*.

ERP
Enterprise Resource Planning. Il termine ERP è stato coniato all'inizio degli anni '90 e comprende numerose attività supportate da applicazioni software che riguardano la gestione integrata di tutte le risorse che partecipano alla creazione dei prodotti/servizi di un'azienda. Ottimizzano la collocazione delle risorse aziendali e realizzano la fornitura di beni e servizi con la massima efficacia. I vantaggi che un sistema ERP può apportare sono numerosi, dalla qualità dei dati alla massima tempestività dell'analisi, alla trasparenza sulla gestione e sulla proprietà dei processi. Di solito i sistemi ERP vengono utilizzati e integrati con sistemi di database.

Extranet
Una rete simile ad Internet ma limitata nell'accesso a partner, fornitori o clienti di un'azienda, cui è stata fornita un'apposita password. Permette di condividere in modo semplice e conveniente informazioni e risorse.

Firewall
Il firewall (letteralmente "parete antincendio") è un sistema che impedisce gli accessi non autorizzati. In pratica è un sistema in grado di controllare l'accesso alle reti intercettando tutti i messaggi in entrata e in uscita. Il firewall, a seconda della configurazione e della tipologia, permette infatti il passaggio solamente di determinati tipi di dati, da determinati PC e da determinati utenti. Il firewall separa e protegge la rete interna, definendo e rafforzando le policy di rete. I computer esterni alla rete devono attenersi a una specifica procedura per ottenere l'accesso alle risorse, agli host e a tutte le altre informazioni. Se l'accesso viene autorizzato, l'utente può “passare”, a patto che si attenga alla procedura definita dal firewall.

GUI
Graphic User Interface. Interfaccia utente che riceve comandi non tramite la digitazione sulla tastiera, ma utilizzando forme grafiche come puntatori, icone, finestre, menu e pulsanti.

HTML
Hyper Text Markup Language. Linguaggio di formattazione dei documenti utilizzato per preparare le pagine che devono essere visualizzate dai browser Web. Creato nel 1991, l'HTML è stato elaborato successivamente dal CERN di Ginevra. Esistono molte versioni; tutte, comunque, devono ottenere l'approvazione del W3C.

HTTP
Hyper Text Transfer Protocol. È il protocollo di comunicazione di Internet che consente lo scambio di pagine scritte in HTML.

Intranet
Rete simile ad Internet ma limitata nell'accesso ai soli dipendenti dell'azienda o membri dell'organizzazione.

Middleware
Insieme di componenti software che realizzano una macchina virtuale (ovvero un insieme di servizi fra loro coerenti e simulanti il comportamento di un unico elaboratore che fosse progettato per erogarli). La macchina virtuale è messa a disposizione delle applicazioni, che la usano mediante chiamate ai servizi da questa offerti. Il middleware realizza la macchina virtuale usando servizi offerti da apparati hardware e software di livello più basso.

MOM
Message Oriented Middleware. Identifica una tecnologia di integrazione che si basa sullo scambio di messaggi tra i programmi oggetto dell'integrazione usando delle apposite code.

ODBC
Open Data Base Connectivity. Standard universalmente diffuso per l’accesso ai dati in maniera trasparente, indipendentemente dalla piattaforma sulla quale risiedono.

OLE
Object Linking and Embedding. Collegamento e incorporamento di oggetti: in Windows, definisce la possibilità di inserire un documento (o parte di esso) creato da una applicazione all'interno di un documento creato da un'altra applicazione, mantenendo un legame attivo tra i due. Per estensione il termine si riferisce anche alle applicazioni in grado di supportare tale protocollo di comunicazione fra processi.

ORB
Object Request Broker. Nucleo centrale operativo di CORBA. ORB intercetta le richieste effettuate dal software client di tipo CORBA, sia che risieda sulla stessa macchina che in rete; quindi si incarica di trovare l'oggetto software che può soddisfare la richiesta, gli "passa" i parametri e ne riceve i risultati che "passa" al software client. É così possibile utilizzare computer con sistemi operativi diversi e con architettura diversa nella stessa rete, senza alcuna controindicazione né inconveniente, conservando la completa compatibilità del software e dei dati.

Router
Termine che indica un dispositivo che sposta i dati tra segmenti di rete diversi ed è in grado di leggere l'header del pacchetto di dati per determinare il percorso di trasmissione migliore. I router possono collegare segmenti di rete che utilizzano protocolli diversi.

RPC
Remote Procedure Call. Meccanismo che, in ambiente di rete, permette di distribuire le elaborazioni tra più computer. Questa tecnica consente, da una parte, di sfruttare meglio le risorse del server e, dall'altra, di ridurre il traffico di rete.

SGML
Standard Generalized Markup Language. È un meta-linguaggio che fornisce un insieme completo di regole sintattiche per modellare la struttura di documenti e dati. HTML è un sottoinsieme di SGML.

SOAP
Simple Object Access Protocol. È un protocollo "leggero" per lo scambio di informazioni in un ambiente distribuito e decentrato. Tale scambio di informazioni avviene mediante messaggi codificati in un formato XML; si parla, pertanto, di messaggistica XML.

TCO
Total Cost of Ownership. Modalità di valutazione degli investimenti nel settore informatico. Il TCO si calcola considerando i costi diretti e indiretti di un'azienda derivanti dall'acquisto di materiale tecnologico.

UDDI
Universal Description Discovery and Integration. È una specifica per la realizzazione di registri distribuiti, basati sul Web, di servizi Web; i registri UDDI sono usati per la ricerca e la promozione di servizi Web.

UML
Unified Modeling Lanugage. È un linguaggio grafico per visualizzare, specificare, costruire e documentare i prodotti di un sistema basato sul software. Offre una modalità standard per scrivere il blueprint di un sistema, contenente sia oggetti concettuali, come i processi di business e le funzioni, sia oggetti concreti, quali gli statements di un linguaggio di programmazione, lo schema del database, i componenti software riutilizzabili. L'UML rappresenta la "summa" delle best practices

nell'ambito della modellazione di tipo object oriented. Lo standard di utilizzo dell'UML è emanato dall'OMG.

W3C
World Wide Web Consortium. Il W3C è stato creato per portare il Web ai massimi livelli, sviluppando protocolli comuni che permettano la sua evoluzione e assicurino l'interoperabilità tra i diversi sistemi. È un consorzio industriale internazionale guidato dal MIT Laboratory for Computer Science (MIT LCS) negli USA, dal National Institute for Research in Computer Science and Control (INRIA) in Francia e dalla Keio University in Giappone. I servizi offerti dal Consorzio includono: un archivio completo sul World Wide Web per sviluppatori e utenti, implementazioni e codici di riferimento e diverse applicazioni per dimostrare l'utilizzo della nuova tecnologia. Oggi il W3C conta più di 410 organizzazioni.

WSDL
Web Services Description Language. È un formato XML per la descrizione di servizi di rete. Un documento WSDL fornisce tutte le informazioni necessarie per l'utilizzo di un servizio Web (formato dei messaggi, protocollo di trasporto, etc.).

XML
eXtensible Markup Language. È un meta-linguaggio standard per la modellazione di documenti e dati. XML è basato su SGML, ma con un insieme ridotto di caratteristiche che lo rende più idoneo alla distribuzione sul Web. XML consente di creare tag personalizzati per fornire funzionalità non disponibili in HTML.

10. Bibliografia

[Bat 00] C. Batini, G. Santucci, *Sistemi informativi per la Pubblica Amministrazione: metodologie e tecnologie;* Scuola Superiore della Pubblica Amministrazione, 2000.

[Bur 00] S. Burbeck, *The evolution of Web applications into service-oriented components with Web services;* The Tao of e-business services – IBM, ottobre 2000.

[DeM 01] G. De Michelis, *Fermiamo gli inventori stregoni o Internet diverrà incontrollabile;* Telèma 25 – Sicurezza e privacy nelle comunicazioni, estate 2001.

[Haz 02] T.K. Hazra, *MDA brings standards-based modeling the EAI teams;* Application Development Trends, maggio 2002.

[Met 01] META Group, *Enterprise Application Infrastructure: Web Services;* Toolnews n° 10, dicembre 2001.

[Nat 01] Y. Natis, J. Thompson, *What can Web Services do for you ?*; Gartner (COM-13-8485), luglio 2001.

[Pat 02] N. Patrignani, *Soluzioni di integrazione per l'e-business;* META Group, Presentazione per il Gruppo di Lavoro CIPA, aprile 2002.

[Pez 01] M. Pezzini, *Core business applications will last, but not forever*; Gartner (COM-12-8164), febbraio 2001.

[Pez 02] M. Pezzini, *Integrating legacy with the WEB: nothing created, nothing destroyed, evrything transformed;* Atti del convegno "Application Integration and Web Services", 10-11 giugno 2002 Roma.

[Tho 01] M. Pezzini, J. Thompson, *Ten golden rules for starting application integration*; Gartner (TG-14-2678), novembre 2001.

[Sch 00] B. Schneier, *Inside risk: semantic network attacks,* "Communication of the ACM"; 43.12, 2000, pag. 168.

[Scu 02] R. Schulte, *The Enterprise Nervous System Changes Everything;* Atti del convegno "Application Integration and Web Services", 10-11 giugno 2002 Roma.

[Sie 01] J. Siegel, *Developing in OMG's Model Driven Architecture,* "Object Management Group – White paper"; novembre 2001.

[Smi 02] D. Smith, *Service-Oriented Development: how and why it will work,* Atti del convegno "Application Integration and Web Services"; 10-11 giugno 2002 Roma.

[Sol 00] R. Soley, *Model driven architecture,* "Object Management Group – White paper"; novembre 2000.

[Sri 01] S. Iyengar, *Model driven architecture meets Web services,* Atti del convegno "Software Development Web Services World"; 27-31 agosto 2001 Boston.

[Tod 02] P. Todorovich, *Quando è in gioco l'accesso all'host;* Computerworld Italia, 11 marzo 2002 Roma.

[Vec 01] D. Vecchio, J. Duggan, J. Feiman, J. Sinur, T. Berg, *Legacy Evolution: Strategies for Reuse, Not Abuse;* Gartner (R-13-3255), giugno 2001.