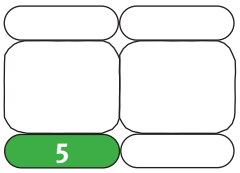
PROGETTO IT4PS
IL COMPUTER PER LA SOLUZIONE DI PROBLEMI

Cristiana Rita Alfonsi
Dino Pedreschi
Nello Scarabottolo
Maria Simi

IT4PS (*Information Technology for Problem Solving*) è un progetto condotto dalla Fondazione CRUI, la Conferenza dei Rettori delle Università Italiane, con AICA. L'obiettivo di IT4PS è quello di stimolare l'apprendimento all'uso approfondito degli strumenti ICT, attraverso una metodologia centrata sui problemi che tali strumenti permettono di risolvere nelle diverse discipline universitarie. IT4PS vuole dunque incoraggiare una formazione orientata alla risoluzione di problemi, e fornire al tempo stesso gli skill richiesti nelle certificazioni professionali di livello avanzato.



1. INTRODUZIONE

Il progetto IT4PS (*Information Technologies for Problem Solving*) – condotto dalla Fondazione CRUI (la struttura operativa della Conferenza dei Rettori delle Università Italiane) insieme con AICA – nasce dalla pluriennale esperienza dei suoi promotori nella formazione, all'interno degli Atenei italiani, all'utilizzo dei principali strumenti informatici (elaboratori di testi, fogli elettronici, gestori di basi di dati). Tale formazione è stata sostenuta in particolare dal progetto Campus*One* – un progetto di dimensione nazionale, gestito dalla Fondazione CRUI, che ha visto la partecipazione di 70 atenei e il coinvolgimento di quasi 300 corsi di laurea, attraverso la diffusione della certificazione ECDL come attestato di raggiungimento del livello di competenza minimale degli studenti universitari nell'utilizzo degli strumenti informatici.
Questa esperienza, sicuramente molto significativa non solo per il numero di studenti universitari coinvolti (dai risultati dell'Osservatorio AICA-CINI-Fondazione CRUI, si parla di più di 50.000 studenti nel solo anno solare 2004) ma anche per il riconoscimento da parte degli atenei di una certificazione rilasciata da terze parti come strumento di verifica delle competenze acquisite, ha mostrato però due limiti principali.
Un primo limite è legato al livello di competenze cui la certificazione ECDL fa riferimento (livello definito dal *syllabus*, ovvero dalla descrizione delle competenze richieste per ottenere la certificazione): si tratta infatti di un livello abbastanza elementare, sufficiente a garantire dimestichezza con gli strumenti informatici ma senz'altro poco adeguato per un utilizzo professionale di tali strumenti, quale quello che ci si può aspettare dagli studenti universitari (futuri professionisti di alto profilo).
Un secondo limite è invece dovuto al tipo di formazione che viene solitamente adottata per preparare alla certificazione: si tratta infatti tipicamente di una formazione *package oriented*, che presenta le potenzialità dei vari strumenti informatici in modo avulso da uno specifico contesto applicativo. La certificazione attesta dunque la familiarità dello studente con lo strumento informatico, ma non

dice nulla circa l'effettiva capacità dello studente di utilizzare in modo proficuo lo strumento stesso per risolvere i propri problemi professionali.
Se il primo limite può considerarsi transitorio – destinato cioè a scomparire automaticamente, poiché le nuove generazioni di studenti tendono ad affacciarsi all'università con competenze operative sugli strumenti informatici sempre più evolute – il secondo limite richiede invece un ripensamento profondo del modo in cui la formazione all'uso degli strumenti informatici deve essere affrontata.
A questo proposito, vale la pena fare un'osservazione apparentemente banale ma profondamente vera: "si impara ad usare bene ciò che serve, che è utile". Ovvero, è più facile imparare ad utilizzare quegli strumenti che dimostrano una evidente utilità nella soluzione di problemi percepiti come importanti. Non è un caso che gli strumenti informatici di base, quali per esempio la videoscrittura o la posta elettronica (ma anche gli SMS scambiati con i telefoni cellulari, che pure richiedono un uso certo non particolarmente confortevole della tastiera) abbiano trovato rapidamente una crescente diffusione: sono infatti strumenti che rispondono in modo naturale ad esigenze diffuse – scrivere documenti e scambiarsi informazioni – con evidenti vantaggi in termini di flessibilità e semplicità rispetto alle tecnologie tradizionali.
Il quadro è però molto diverso quando si passa dalla pura trasmissione delle informazioni alla loro organizzazione ed elaborazione: quando si passa cioè dalla comunicazione al calcolo. Sono disponibili ormai da anni strumenti informatici per esprimere esigenze di calcolo comunemente riscontrabili nell'ambito professionale (ma anche personale e familiare): i *fogli elettronici*; analogamente, sono disponibili gli strumenti informatici per organizzare, gestire e reperire le informazioni comunemente trattate nei vari ambiti professionali, personali e familiari: i gestori di *basi di dati*. Gli uni e gli altri strumenti possono essere usati per risolvere problemi significativi in modo ben più efficace dei metodi tradizionali: dall'impostazione di un bilancio familiare alla gestione delle cartelle cliniche dei pazienti di uno studio medico. Eppure, la diffusione del foglio elettronico e della base di dati è ben più scarsa rispetto a quella degli strumenti per scrivere e comunicare. La capacità delle persone di usare questi strumenti informatici più evoluti è assai limitata ovunque al mondo, ma l'Italia si trova ad affrontare un ritardo marcato, rispetto agli altri Paesi sviluppati, e nell'indagine promossa nel 2003 da AICA sul *Costo dell'ignoranza nella società dell'informazione* [5] si stima che tale ritardo abbia influito e influisca in modo pesante come fattore deprimente della nostra economia.
I motivi di questo ritardo sono senz'altro legati al fatto che, padroneggiare questi strumenti, richiede uno sforzo concettuale decisamente maggiore di quello necessario a padroneggiare la videoscrittura o la posta elettronica: il foglio elettronico e la base dati sono strumenti dalle molteplici potenzialità, ma che richiedono una concettualizzazione e una modellizzazione dei problemi da risolvere per poter essere applicati proficuamente. Inoltre, sono strumenti di uso generale, non legati ad uno specifico problema, ma plasmabili (programmabili) per risolvere problemi diversi. Per questi motivi, entrambi gli strumenti non si prestano ad essere appresi in modo puramente "strumentale", lasciandosi guidare dalle funzionalità rese disponibili agli utenti. C'è in realtà un solo vero modo per imparare ad usarli e diventarne progressivamente esperti: *lasciarsi guidare dai problemi* risolubili attraverso questi strumenti, e farlo utilizzando problemi familiari all'utente, da questo percepiti come rilevanti, naturali, stimolanti, difficili o scomodi da trattare utilizzando le tecnologie tradizionali di risoluzione. Basando cioè l'apprendimento su un approccio non orientato allo strumento, bensì *orientato ai problemi* (*problem oriented*) risolubili con lo strumento, *contestualizzati* in un ambito familiare all'utente. Infatti, come già detto, si impara ad usare solo ciò che serve.
Questo tipo di percorso di apprendimento, dosato attraverso problemi di crescente complessità, consente di esplorare tutte le principalità funzionalità offerte dagli strumenti informatici, arrivando a padroneggiare non solo le idee di fondo su cui questi si

basano, ma anche la tecnica per costruire effettivamente soluzioni ai problemi utilizzando appieno le tecnologie del foglio elettronico e della base di dati. Ovviamente, per raggiungere queste conoscenze e competenze avanzate, è necessario sostenere la motivazione dell'utente, proponendo problemi non artificiosi ma rilevanti e stimolanti, calati cioè nel contesto degli interessi di chi apprende.
Le considerazioni sopra riportate sono alla base del progetto IT4PS, lanciato nel 2003 proprio con l'obiettivo di creare esperienze di apprendimento all'uso contestualizzato dei più diffusi strumenti di gestione di fogli elettronici e di basi di dati, e sperimentarle nelle università italiane.

2. IL PROGETTO IT4PS

Come già ricordato nell'introduzione, il progetto IT4PS nasce da una collaborazione fra Fondazione CRUI e AICA, iniziata a metà 2003, che ha visto la conclusione della sua prima fase sperimentale a metà del 2005.
Le assunzioni alla base del progetto IT4PS, già discusse in precedenza, possono essere così riassunte:
- si impara a conoscere bene gli strumenti informatici se li si utilizza quotidianamente per risolvere problemi concreti;
- la formazione all'operatività informatica (ovvero la preparazione alla certificazione ECDL) è di livello non universitario e avulsa dall'effettivo uso quotidiano degli strumenti (troppo *package oriented*);
- gli strumenti informatici più diffusi (foglio elettronico e gestore di basi di dati) possono essere utilizzati in modo avanzato per risolvere problemi specifici del contesto curricolare degli studenti universitari;
- una certificazione di livello avanzato è un valore aggiunto, soprattutto per studenti di corsi di laurea non specificamente tecnici.

Va senz'altro ricordato come il ruolo essenziale dell'università, sostenuto da progetti di sistema come Campus*One*, sia stato quello di supplire nelle competenze operative di base, fornendo ai propri studenti i livelli di alfabetizzazione all'uso del PC. Non c'è dubbio però che il livello di competenza della certificazione ECDL non sia adeguato alla realtà universitaria: una competenza puramente strumentale, basata sulle funzionalità di base offerte dagli strumenti e slegata dai contenuti e dai metodi delle discipline di studio universitario, non è certo un fine della formazione universitaria. In questa prospettiva, è importante dare degli strumenti informatici una conoscenza che metta nel giusto rapporto il "sapere" con il "saper fare", ma che al tempo stesso non si riduca a una semplice conoscenza dell'interfaccia dello strumento: questo tipo di conoscenza, infatti, non corrisponde, se non in minima parte, a una reale capacità di utilizzo dell'applicativo nella risoluzione dei problemi legati alla propria professione.
Non è affatto un caso che le competenze previste nell'ECDL Core per il foglio elettronico, e ancor più per la base di dati, siano insufficienti per un uso efficace di questi strumenti più "concettuali" in contesti realistici. Era quindi auspicabile che l'università passasse quanto prima a farsi carico delle competenze operative ICT più avanzate – i veri e propri *e-skills* – integrandole nelle discipline e nelle professioni, ovvero integrandole appieno nell'apprendimento di livello universitario.
Per dare una risposta concreta a queste considerazioni, IT4PS si è proposto tre obiettivi fondamentali:
1. insegnare agli studenti universitari a usare gli strumenti informatici di contenuto concettuale significativo (fogli elettronici e gestori di basi di dati) in un'ottica *problem oriented*;
2. fornire agli studenti universitari una conoscenza approfondita degli strumenti, tale da permettere di accedere alla certificazione A-ECDL (*European Computer Driving Licence, Advanced Level*): una certificazione destinata come l'ECDL all'utente finale dello strumento informatico, ma rivolta a testimoniare una competenza e una comprensione profonda delle possibilità dello strumento;
3. permettere un auto-apprendimento e una auto-valutazione delle capacità di *problem solving* raggiunte da parte degli studenti.
In sintesi il progetto IT4PS si è proposto di predisporre una formazione di competenze operative avanzate ICT di livello universitario, in riferimento particolare al foglio elettro-

nico ed alla base di dati, secondo un approccio didattico orientato al *problem solving* e contestualizzato all'ambito disciplinare di specifiche famiglie di corsi di laurea. Con un importante effetto collaterale aggiuntivo: la preparazione dello studente al conseguimento della certificazione A-ECDL. Lo studente che ha completato con successo la formazione *problem-oriented* allo strumento deve potere, con un moderato impegno aggiuntivo, sostenere con successo la verifica per il conseguimento di tale certificazione.
Inoltre, dal momento che l'apprendimento di uno strumento informatico implica inevitabilmente una grossa componente di lavoro autonomo da parte dello studente, il progetto IT4PS si è preoccupato di realizzare supporti che facilitassero tale lavoro autonomo, in particolare per quanto riguarda la verifica del livello di competenza raggiunto.

3. STRUTTURA DEL PROGETTO IT4PS

La fase sperimentale del progetto IT4PS è stata gestita da un team di conduzione del progetto stesso, costituito dagli autori del presente lavoro.
Il team di conduzione ha individuato le seguenti tre aree disciplinari nelle quali sperimentare un approccio *problem oriented* all'apprendimento dell'uso del foglio elettronico e del gestore di basi di dati:

1. economia;
2. medicina e farmacia;
3. statistica per le scienze sociali.

Per ciascuna di queste tre aree, è stato poi costituito un team di contestualizzazione, ovvero un gruppo interdisciplinare di docenti dell'area e di esperti informatici, incaricato di sviluppare un metodo di insegnamento e di verifica della capacità dello studente di elaborare soluzioni creative a problemi non elementari specifici del singolo contesto curricolare, per ciascuno dei due strumenti informatici – foglio elettronico e basi di dati.
In parallelo alle attività dei team di contestualizzazione, ha operato un quarto team di sviluppo dei supporti alla auto-valutazione delle capacità di *problem solving* raggiunte dagli studenti. Tale gruppo ha avuto il compito di realizzare due sistemi autore (uno per i fogli elettronici, l'altro per le basi di dati) pensati per:

- ❑ consentire al docente di definire un possibile schema di soluzione a un problema da proporre agli studenti;
- ❑ permettere a ciascuno studente di impostare una propria soluzione al problema proposto dal docente, verificandone autonomamente la correttezza.

Il progetto IT4PS si è sviluppato secondo una tempistica data dalle seguenti scadenze:
dicembre 2003: predisposizione del materiale didattico e del supporto di auto-valutazione per l'erogazione di un modulo formativo per l'apprendimento dell'uso del foglio elettronico in ciascuna delle tre aree di contestualizzazione;
marzo 2004: avvio della sperimentazione in aula dei tre moduli di contestualizzazione del foglio elettronico. Ogni modulo, che prevedeva indicativamente 30 h di formazione frontale, è stato erogato da alcuni corsi di studio che hanno aderito alla sperimentazione; la sperimentazione ha coinvolto 7 atenei;
maggio 2004: svolgimento di sessioni di certificazione A-ECDL AM4 (foglio elettronico) per valutare il livello di competenza raggiunto dagli studenti sperimentanti;
giugno 2004: predisposizione del materiale didattico e del supporto di auto-valutazione per l'erogazione di un modulo formativo per l'apprendimento dell'uso del gestore di basi di dati in ciascuna delle tre aree di contestualizzazione;
settembre 2004: avvio della sperimentazione in aula dei tre moduli di contestualizzazione del gestore di basi di dati. Ogni modulo, che prevedeva indicativamente 30 h di formazione frontale, è stato erogato da alcuni corsi di studio che hanno aderito alla sperimentazione; la sperimentazione ha coinvolto 10 atenei;
novembre 2004: svolgimento di sessioni di certificazione A-ECDL AM5 (basi di dati) per valutare il livello di competenza raggiunto dagli studenti sperimentanti;
dicembre 2004: predisposizione di un testo didattico per ciascuno dei moduli di contestualizzazione del foglio elettronico, in grado di costituire un supporto completo all'erogazione dei moduli stessi da parte degli atenei interessati a proporli nella propria offerta formativa;

aprile 2005: predisposizione di un testo didattico per ciascuno dei moduli di contestualizzazione del gestore di basi di dati, in grado di costituire un supporto completo all'erogazione dei moduli stessi da parte degli atenei interessati a proporli nella propria offerta formativa;
giugno 2005: predisposizione di un testo didattico per l'utilizzo – da parte del docente – dei sistemi autore, onde consentire sia la progettazione di nuovi problemi nelle tre aree di contestualizzazione, sia soprattutto l'estensione dell'approccio *problem oriented* di IT4PS ad altre aree disciplinari.
Tutti i sette testi prodotti nell'ambito del progetto IT4PS (riprodotti nella Figura 1), editi dalla casa editrice McGraw-Hill Italia, sono attualmente disponibili in libreria.
A ciascuno dei primi sei testi è allegato un CD-ROM contenente il materiale necessario allo studente per seguire la trattazione sperimentando immediatamente le tecniche proposte, alcuni esercizi di auto-valutazione realizzati con il sistema-autore PS-Welcome (per i fogli elettronici) o Access Test Manager (per le basi di dati) e un esempio di test di certificazione Advanced ECDL.
Il settimo testo contiene invece un CD-ROM con entrambi i sistemi-autore.

4. I MODULI DI CONTESTUALIZZAZIONE DEL FOGLIO ELETTRONICO

In questa sezione e nella successiva, si riportano alcune brevi note informative sui problemi effettivamente affrontati, rispettivamente nei tre moduli formativi basati sull'utilizzo del foglio elettronico e negli analoghi basati sull'utilizzo delle basi di dati.
Come ovvio, la scelta di tali problemi si è spesso indirizzata a un'area particolare della disciplina considerata: è sintomatico a questo riguardo il caso dell'uso del foglio elettronico in medicina, che ha preso come ambito di riferimento la dietologia e il calcolo del fabbisogno calorico.
È però importante sottolineare come tale scelta non sia assolutamente limitativa della generalità dell'approccio IT4PS: lo studente viene infatti educato ad affrontare e a risolvere con lo strumento informatico un problema concreto, e

FIGURA 1
I testi prodotti nell'ambito del progetto IT4PS

i passi che lo conducono a tale risultato (analisi del problema, impostazione di una soluzione minimale, raffinamento della soluzione ecc.) costituiscono uno strumento metodologico che lo studente stesso potrà poi facilmente riutilizzare anche in altre aree disciplinari.

4.1. Il foglio elettronico per Economia

Il modulo didattico realizzato parte da una serie di problemi di interesse per l'area economica e per ciascuno di essi viene mostrato come lo strumento informatico fornisca un valido supporto per ottenere una soluzione in modo semplice ed efficiente.
I problemi affrontati sono:
- le scelte di investimento di un'impresa la cui attività è finalizzata all'ottenimento del massimo profitto: a fronte di un caso specifico, viene illustrata una possibile soluzione che consiste nella individuazione del livello di investimento che minimizza la distanza tra il rendimento e il tasso di interesse di mercato;
- le scelte di consumo intertemporale di un individuo, cioè il problema della scelta di allocazione del reddito di un consumatore tipo tra consumo corrente e consumo futuro attraverso la possibilità di dare o di prendere fondi a prestito sul mercato dei capitali;
- la programmazione dei flussi di cassa, che consiste nell'elaborazione di uno schema di pagamenti ottimale;
- la determinazione del Prodotto Interno Lordo (PIL) di una nazione;

❑ l'analisi delle vendite di un gruppo di consulenti finanziari, un problema che consente di illustrare le tecniche per la produzione di un *report* che mostri l'attività di consulenti finanziari nell'ambito di una banca;
❑ l'analisi del punto di equilibrio della contabilità dei costi (*break-even point*);
❑ il calcolo del tasso di congrua remunerazione di un titolo azionario: si risolve un esempio specifico attraverso il calcolo del coefficiente beta.

4.2. Il foglio elettronico per Medicina e Farmacia

Il modulo didattico realizzato parte da un problema di calcolo tipico dell'area sanitaria: quello della prescrizione dietetica. L'articolazione del modulo segue quindi i diversi sotto-problemi coinvolti nella definizione di una dieta, quali la determinazione dell'indice di massa corporea del paziente, la determinazione del suo fabbisogno calorico, fino ad arrivare alla composizione di una dieta che tenga conto degli aspetti fisiologici e patologici del paziente.
Il testo realizzato come supporto al modulo presenta una breve introduzione ai problemi dell'obesità e della prescrizione dietetica. Le tematiche di contesto sono poi sviluppate per passi successivi:
❑ prima viene affrontato il problema della valutazione di un paziente, ovvero di come determinarne il peso ottimale calcolando l'indice di massa corporea e confrontandolo con i valori di riferimento;
❑ quindi viene affrontato il problema di come determinare il fabbisogno calorico di un paziente, calcolando sia il metabolismo basale che il dispendio calorico per attività;
❑ successivamente viene affrontato il problema della prescrizione dietetica per un paziente, componendo una dieta che soddisfi il fabbisogno di carboidrati, proteine e lipidi;
❑ infine, vengono illustrate tecniche per un'analisi quantitativa e qualitativa della dieta composta.
Nella figura 2, si mostra un esempio di foglio per l'immissione controllata delle portate di un pasto dietetico e per l'immediato computo dei suoi apporti nutrizionali.

4.3. Il foglio elettronico per la Statistica nelle Scienze sociali

Il modulo didattico realizzato parte da una serie di problemi tipici dell'area statistica. In particolare, ci si concentra sull'analisi di dati disponibili in un generico Ateneo: profili degli studenti, carriere degli studenti, dati per la gestione del personale docente e non docente.
Il testo realizzato come supporto al modulo presenta innanzitutto la ricerca statistica che si intende effettuare, con i relativi obiettivi, e mostra come devono essere importati e preparati i dati per le elaborazioni successive.
Successivamente, si analizzano i dati sugli studenti, evidenziando dei profili anagrafici e analizzando la carriera passata e presente. Per esempio, nella figura 3, si mostra come costruire un istogramma che riporti la distribuzione di frequenza del voto di maturità per gli studenti iscritti all'Ateneo a partire dai dati di ciascuno.

Colazione

Alimento	Quantità	Carboidrati	Proteine	Lipidi	Apporto calorico
fette biscottate	80 gr	66,4	9,0	4,8 gr	345,0 Kcal
marmellata	25 gr	14,7	0,1	0,0 gr	59,2 Kcal
mela	150 gr	16,1	0,6	0,2 gr	68,0 Kcal
yogurt intero frutt	125 gr	15,8	3,5	4,1 gr	114,1 Kcal
TOTALI	380 gr	97,1 388,5	9,8 39,1	5,0 gr 44,6 Kcal	586,2 Kcal

FIGURA 2
Un esempio di foglio elettronico per la definizione di un pasto dietetico

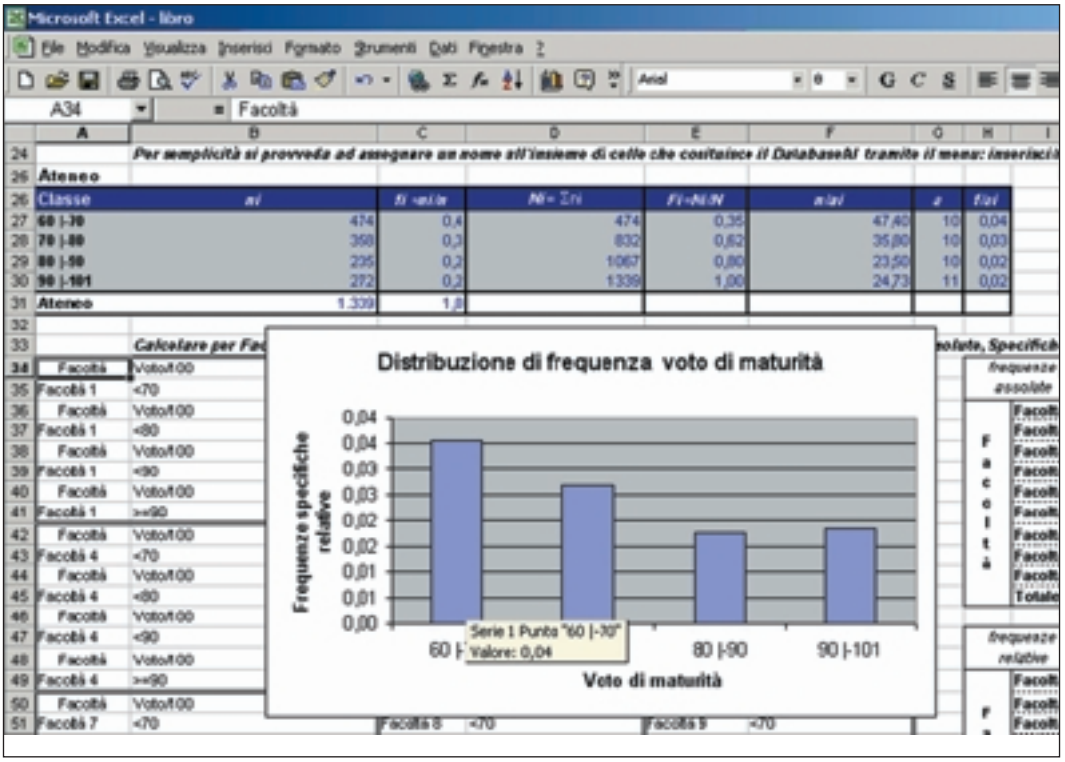


FIGURA 3
Un esempio di foglio elettronico per la valutazione dei voti di maturità degli studenti

Si prende poi in considerazione il personale docente e non docente, analizzando il problema delle retribuzioni.

5. I MODULI DI CONTESTUALIZZAZIONE DELLE BASI DI DATI

5.1. Le basi di dati per Economia

Il modulo didattico realizzato parte da due macro-problemi di interesse per l'area economica:

- la costruzione di indici macroeconomici per il mercato del lavoro;
- la costruzione di indici per l'analisi di bilancio delle aziende.

Dopo una breve introduzione al problema della costruzione di indici macroeconomici per il mercato del lavoro, il testo prosegue con la descrizione di un contesto specifico in termini dei requisiti che la base di dati deve soddisfare, degli indici richiesti e delle funzioni di interrogazione sui dati necessarie alla loro costruzione. Segue poi l'illustrazione di una possibile soluzione che, partendo dalla creazione delle strutture di memorizzazione dei dati, illustra i meccanismi più efficienti per la costruzione di interrogazioni complesse per il calcolo degli indici di interesse, i metodi per la visualizzazione dei risultati e per automatizzare l'esecuzione sequenziale delle interrogazioni.

Successivamente, la stessa metodologia didattica viene applicata al problema della costruzione di indici per l'analisi di bilancio delle aziende. In questo caso, viene ipotizzato uno scenario in cui i dati di partenza provengono da ambienti e formati eterogenei, e vengono quindi anche descritti e risolti vari problemi derivanti dall'esigenza di armonizzare i dati e integrarli in una base di dati unica.

5.2. Le basi di dati per Medicina e Farmacia

Il modulo didattico realizzato parte da un problema di particolare interesse per l'area sanitaria: quello della gestione di tutte le informazioni relative ai pazienti di un medico di medicina generale. L'articolazione del modulo segue quindi tutti i diversi problemi coinvolti in tale gestione: come organizzare i dati relativi ai pazienti, come effettuare ricerche e statistiche su di essi, come gestire dati relativi a visite e patologie dei pazienti.

Il testo realizzato come supporto al modulo presenta dapprima una breve introduzione ai problemi della gestione dei dati relativi ai pazienti

di un medico di medicina generale: vengono analizzati i limiti degli archivi cartacei e gli aspetti principali da considerare nella progettazione di una base di dati. Le tematiche di contesto sono poi sviluppate per passi successivi:

- prima viene affrontato il problema di come organizzare i dati relativi ai pazienti e di come permettere l'inserimento, la cancellazione e la modifica di tali dati;
- si passa poi a illustrare come svolgere ricerche e statistiche sui dati dei pazienti;
- successivamente viene affrontato il problema di come estendere la base di dati in modo da poter introdurre dati relativi a visite e a patologie dei pazienti;
- viene quindi illustrato come poter determinare correlazioni tra i dati relativi a pazienti, visite e patologie;
- infine, viene descritta la possibilità di prevedere un accesso facilitato, tramite interfaccia grafica, sia all'interrogazione sia alla gestione dei dati contenuti in una base di dati.

5.3. Le basi di dati per la Statistica nelle Scienze sociali

Il modulo didattico realizzato si concentra sull'analisi dei dati di una Compagnia Assicurativa: analisi dei dipendenti e dei concorrenti.
Il testo realizzato come supporto al modulo presenta dapprima la ricerca statistica che si intende effettuare, con i relativi obiettivi, e mostra come devono essere importati e preparati i dati per le elaborazioni successive, che partendo dai dati sui dipendenti, si pongono una serie di obiettivi di business quali il lancio di una nuova polizza, l'analisi dei dipendenti che ricevono premi e sostengono straordinari, la verifica della soddisfazione dei dipendenti rispetto alla propria polizza assicurativa.
Da ultimo, il testo si concentra sulle fonti pubbliche dei dati e in particolare sui *data warehouse* pubblici dell'ISTAT: la parte finale è dedicata al confronto delle performance delle diverse filiali basandosi proprio sui dati interni e sulle informazioni reperibili tramite le fonti ISTAT.

6. I SISTEMI AUTORE

Come già sottolineato, nell'ambito del progetto IT4PS sono stati realizzati anche due sistemi autore – *PS-Welcome* (per i fogli elettronici) e *Access Test Manager* (per le basi di dati) – che possono essere utilizzati dal docente al fine di predisporre soluzioni guidate a problemi anche complessi, e allo studente per sviluppare la propria soluzione a tali problemi, confrontando il comportamento di quanto da lui prodotto con quanto previsto dal docente.
Si tratta dunque a tutti gli effetti di due supporti all'auto-apprendimento, concepiti per aiutare lo studente nella costruzione di soluzioni creative a problemi non elementari e tali da incoraggiare una metodologia di risoluzione mediante decomposizione dei problemi stessi in una gerarchia di sotto-problemi più semplici: risolvendo i sotto-problemi e sfruttando via via i risultati parziali, si arriva alla soluzione del problema nella sua interezza.
Uno degli aspetti più significativi dei due sistemi autore è la modalità di verifica della correttezza della soluzione proposta dallo studente: i due strumenti si pongono infatti l'obiettivo di valutare le soluzioni (ovvero le formule risolutive nel caso del foglio elettronico e le interrogazioni nel caso delle basi di dati) non in base alla loro similitudine con quanto proposto dal docente, bensì in termini dei *risultati* che con esse si ottengono: una soluzione dello studente viene dunque considerata corretta se produce risultati esatti su una buona varietà di dati iniziali, e non in base a quanto si avvicina alla proposta del docente. Si tratta in altre parole di un'analisi di tipo *comportamentale* e non *strutturale* della soluzione (vedere un esempio nel riquadro).
Per evitare che gli errori "si accumulino", sono previste modalità di risoluzione (e valutazioni) parziali: se lo studente non è in grado di risolvere un sotto-problema, può comunque utilizzare il risultato fornito dal docente per risolvere i problemi di livello superiore. La soluzione dei sotto-problemi può quindi essere valutata separatamente. Inoltre lo studente può adattare il livello di difficoltà dell'esercizio alle sue capacità, chiedendo suggerimenti e pagando un "prezzo" in termini di valutazione.

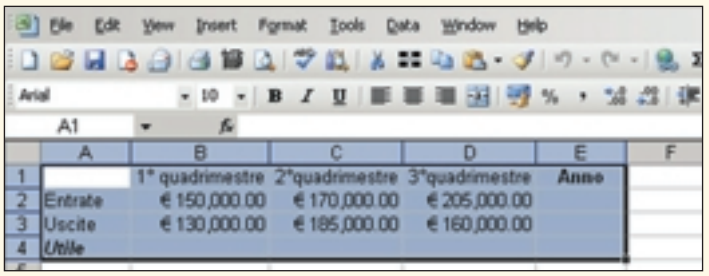
6.1. PS-Welcome

PS-Welcome è il sistema autore che supporta l'apprendimento delle capacità di risoluzione di problemi legate alla definizione di formule risolutive e alla creazione di fogli di calcolo riutilizzabili. Nella versione attuale, PS-Welcome prevede l'utilizzo di Microsoft Excel

La valutazione comportamentale della correttezza di una soluzione a un problema con i fogli elettronici

Come indicato nell'articolo, i sistemi autore PS-Welcome e Acces Test Manager – sviluppati nell'ambito del progetto IT4PS – affrontano la correzione della soluzione che uno studente dà a un problema proposto dal docente sulla base del *comportamento* di tale soluzione, non della sua *struttura*.

Per meglio comprendere il funzionamento dei sistemi autore nella fase di valutazione della correttezza della soluzione proposta dallo studente, si può fare riferimento al semplice schema di calcolo riportato nella figura, nel quale, a partire da entrate e uscite quadrimestrali, si chiede di calcolare entrate e uscite annuali e l'utile netto sia quadrimestrale che annuale, quindi di inserire formule opportune nelle celle E2 (entrate annuali), E3 (uscite annuali), B4, C4 e D4 (utili quadrimestrali) e E4 (utile annuale).



Un semplice esempio di problema da risolvere

Il docente risolverebbe probabilmente il problema inserendo in E2 la formula =SOMMA(B2:D2), copiando la cella E2 in E3, inserendo in B4 la formula =B2-B3 e copiando la cella B4 in C4, D4 e E4. Tuttavia, una diversa soluzione – magari meno brillante ma altrettanto valida *ai fini della risoluzione del problema* – è quella di inserire in B4 la formula =B2-B3, copiare la cella B4 in C4 e D4, poi inserire in E2 la formula =B2+C2+D2 e copiare la cella E2 in E3 e E4. È invece chiaramente sbagliato scrivere in B2 l'espressione =150000-130000: a prima vista il risultato è il medesimo, ma basta ovviamente modificare i dati di ingresso per notare che l'utile quadrimestrale non viene aggiornato.

Un sistema di valutazione della correzione basato sul confronto fra quanto specificato dal docente e quanto inserito dallo studente (quindi su un'analisi *strutturale*) definirebbe errate entrambe le soluzioni: PS-Welcome invece (così come Access Test Manager per le basi di dati) verifica automaticamente come si *comporta* la soluzione dello studente al variare dei dati di ingresso, e ritiene valida la prima proposta (che si comporta allo stesso modo della soluzione del docente) mentre individua l'errore nella seconda.

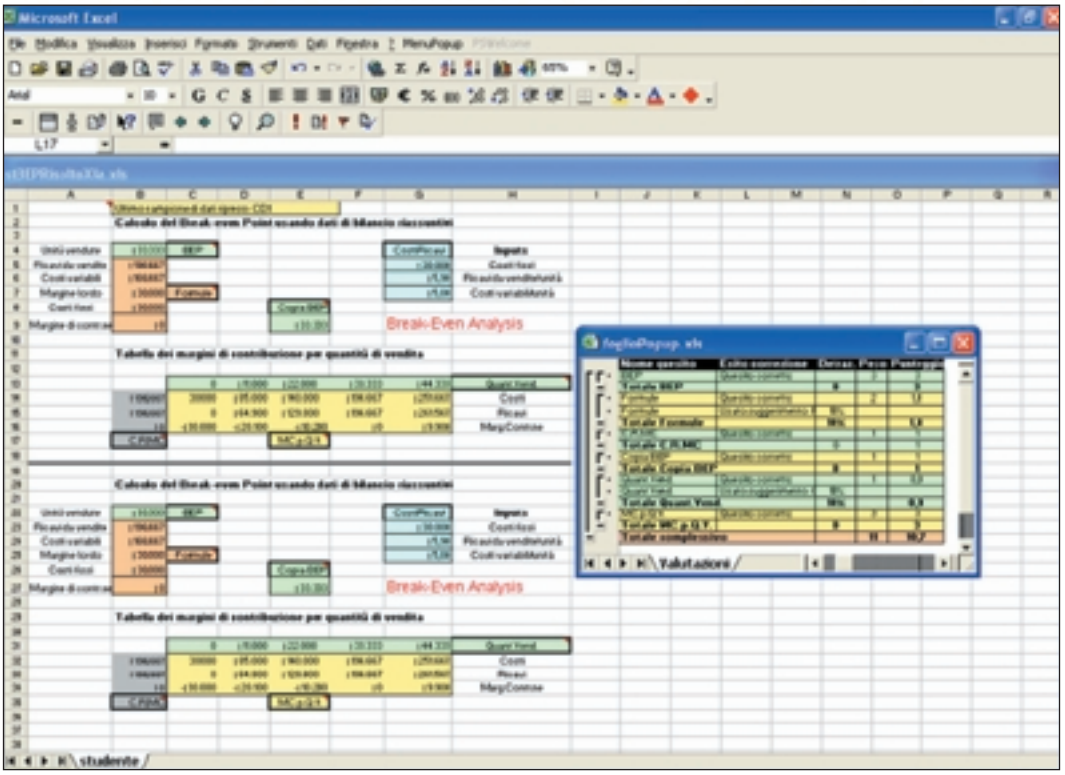


FIGURA 4
Interfaccia per lo studente di PS-Welcome

2002 (presente nella suite Office XP) come gestore di fogli elettronici.

Nella figura 4 è mostrata l'interfaccia per lo studente di PS-Welcome, mentre lo studente interagisce per risolvere un problema preparato dal docente. Nella parte bassa dello schermo sono visualizzati i risultati attesi. La parte alta è la zona in cui lo studente immette le sue formule per cercare di riprodurre i risultati corretti. La finestra pop-up mostra l'e-

sito di una valutazione intermedia della soluzione proposta dallo studente.
Si noti come l'interfaccia di PS-Welcome sia assolutamente analoga a quella dello strumento Excel, opportunamente arricchita con menù per le funzionalità aggiuntive.

6.2. Access Test Manager

Access Test Manager è il sistema autore che supporta l'apprendimento della capacità di risoluzione di problemi mediante la formulazione di interrogazioni in basi di dati relazionali. Nella versione attuale, Access Test Manager prevede l'utilizzo di Microsoft Access 2002 (presente nella *suite* Office XP) come gestore di basi di dati.
Nella figura 5 è mostrata l'interfaccia per lo studente di Access Test Manager, mentre lo studente interagisce per risolvere un problema preparato dal docente. Nella parte bassa dello schermo sono visualizzate le tabelle che lo studente dovrebbe ottenere come risultato della interrogazione che risolve il quesito proposto. La parte alta è la zona in cui lo studente formula una interrogazione per cercare di riprodurre i risultati. Le finestre pop-up mostrano il testo del quesito proposto e l'esito di una valutazione intermedia delle soluzioni fornite dallo studente ai vari quesiti.
Si noti come anche l'interfaccia di Access Test Manager sia assolutamente analoga a quella dello strumento Access, opportunamente arricchita con menù per le funzionalità aggiuntive.

7. GLI ESITI DELLA SPERIMENTAZIONE

Come già ricordato, la sperimentazione dei materiali didattici prodotti nell'ambito del progetto IT4PS si è svolta in due fasi: la prima relativa ai moduli per i fogli elettronici, la seconda a quelli per le basi di dati.
La sperimentazione sul foglio elettronico ha coinvolto 7 atenei, in cui sono stati attivati 10 moduli didattici distinti: 3 nel settore economico, 4 nel settore di medicina-farmacia, 3 nel settore della statistica per le scienze sociali. I moduli didattici sono stati offerti agli studenti tipicamente come attività aggiuntive, e solo in alcuni casi come parte di corsi più ampi. Gli studenti hanno visto riconosciuto il proprio impegno di partecipazione ai corsi in termini di cfu (crediti formativi universitari: tipicamen-

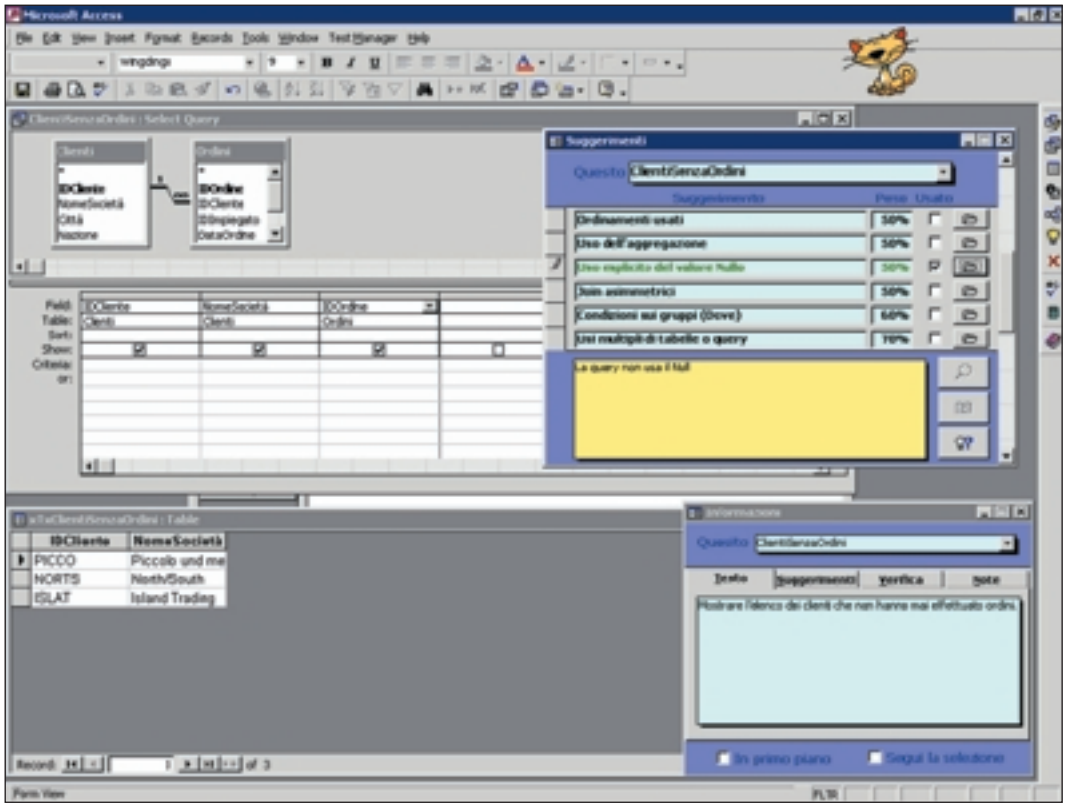


FIGURA 5
Interfaccia per lo studente di Access Test Manager

te 3) e hanno avuto l'opportunità di sostenere l'esame per la certificazione A-ECDL (relativamente al syllabus AM4 – foglio elettronico).
La sperimentazione sulle basi di dati – svolta in modo simile a quella per i fogli elettronici – ha coinvolto 12 atenei, in cui sono stati attivati 12 moduli didattici distinti: 4 nel settore economico, 3 nel settore di medicina-farmacia, 5 nel settore della statistica per le scienze sociali. Al termine dei moduli, gli studenti hanno naturalmente potuto sostenere l'esame per la certificazione A-ECDL (relativamente al syllabus AM5 – basi di dati).

7.1. La customer satisfaction

Per valutare il livello di gradimento dell'approccio contestualizzato, al termine delle sperimentazioni sono stati raccolti dei questionari, con domande simili proposte sia ai docenti sia agli studenti; scopo principale era quello di stabilire l'efficacia dell'approccio didattico, del materiale didattico fornito e del sistema di auto-apprendimento. Sono inoltre stati richiesti suggerimenti su come migliorare i diversi aspetti.
Dall'analisi delle risposte, emerge chiaramente che l'approccio didattico orientato alla risoluzione dei problemi di contesto e il materiale fornito a supporto delle lezioni hanno ricevuto giudizi ampiamente positivi, sia da parte dei docenti sia degli studenti.
La maggior parte dei docenti ha integrato il materiale fornito con altro materiale e in misura minore non ha utilizzato tutto il materiale fornito. Nella fase di valutazione della sperimentazione, il materiale aggiuntivo prodotto è stato raccolto ed analizzato per migliorare i corsi. In modo simile si è cercato di attenuare il problema della carenza di esercizi, anche questo segnalato da docenti e studenti.

7.2. Il comportamento degli studenti alla certificazione

Già dall'analisi dei questionari di *customer satisfaction*, si notano alcune riserve riguardanti il fatto che i problemi presentati, ancorché giudicati interessanti, fossero i più idonei ad apprendere le potenzialità del foglio elettronico e in particolare quelle richieste dal syllabus A-ECDL, soprattutto da parte degli studenti. Si noti che, al momento della somministrazione del questionario, gli studenti non si erano ancora sottoposti all'esame di certificazione, e quindi tale indicazione fa riferimento alla sola percezione dei docenti e degli studenti).
L'esito degli esami ha poi confermato, in maniera estremamente evidente, che la preparazione ricevuta non era adeguata per il superamento dell'esame ufficiale di certificazione: dei primi studenti sottopostisi a certificazione sul foglio elettronico, solo 4 hanno superato l'esame, mentre 74 non lo hanno superato; sicuramente migliore è stato l'esito della prima tornata di certificazioni sulle basi di dati, dove comunque a fronte di 20 superamenti si sono avuti 44 fallimenti.
Se da un lato questo è attribuibile al poco tempo che i docenti hanno avuto per preparare il corso e gli studenti per prepararsi agli esami, questo dato dice che, nonostante il syllabus fosse stato coperto dai corsi al 90%, senza una preparazione mirata e l'esposizione ad esami tipo, gli studenti si trovano impreparati a sostenere una certificazione di livello avanzato ma inevitabilmente NON *problem oriented*, bensì orientata a verificare la conoscenza approfondita dei dettagli dell'interfaccia dello strumento e la conseguente capacità di svolgere le operazioni richieste in tempi brevi. Questa differenza di impostazione risulta ancora più evidente nel caso del foglio elettronico, dove l'esame di certificazione richiede una conoscenza approfondita di alcune funzionalità dello strumento di uso poco frequente, mentre nel caso delle basi di dati si riesce a verificare una conoscenza avanzata ricorrendo ad aspetti concettualmente più profondi (come interrogazioni particolarmente sofisticate) e quindi più facilmente utilizzati anche in un approccio *problem oriented*.

8. VERSO UNA CERTIFICAZIONE A-ECDL/IT4PS

L'impostazione del progetto IT4PS – che tenta di coniugare una preparazione *problem oriented* con il conseguimento di una certificazione europea avanzata come A-ECDL – ha mostrato l'indubbio pregio di calare l'apprendimento operativo dello strumento nel contesto curricolare dello studente, ma di salvaguardare al tempo stesso il valore aggiunto che una certificazione di terze parti riveste nel percorso formativo personale dello studente stesso.

D'altro canto, come dimostrato nella sperimentazione (in particolare, quella relativa al foglio elettronico) il tipo di conoscenze richieste allo studente per conseguire la certificazione si discosta in modo significativo da quello necessario per affrontare la risoluzione di problemi: per verificare una conoscenza approfondita *dello strumento*, i test di certificazione insistono infatti su dettagli operativi poco significativi dei problemi trattati, ma che sono discriminanti sull'esito finale dell'esame in sede di certificazione.
Per minimizzare tale scollamento, e rendere l'approccio *problem oriented* sempre più centrale non solo nella fase formativa ma anche in quella finale di verifica, il progetto IT4PS – grazie alla collaborazione con AICA – è recentemente entrato in una seconda fase, mirata alla realizzazione di una certificazione avanzata (denominata A-ECDL/IT4PS) che utilizzi come test di certificazione esempi di problemi analoghi a quelli affrontati dallo studente durante la formazione contestualizzata, e soprattutto che verifichi le competenze acquisite con domande che implichino per lo studente la capacità di affrontare e risolvere problemi anche complessi, piuttosto che la conoscenza dei dettagli operativi dello strumento utilizzato. Tale certificazione, che sarà inizialmente sperimentata a livello italiano per essere poi proposta in sede europea, sarà resa disponibile agli studenti universitari entro il 2006.

9. CONCLUSIONI E PROSPETTIVE

Come già detto in precedenza, il materiale didattico prodotto e raffinato attraverso la sperimentazione IT4PS è divenuto il punto di partenza per la stesura di una serie di libri, che hanno visto la luce dopo un faticoso processo di ideazione, sperimentazione e validazione dell'approccio didattico, oltre che di accurato *editing*. L'ambizione del progetto IT4PS è far sì che questa offerta possa rivelarsi interessante anche al di fuori delle aule universitarie, come occasione di formazione continua delle persone che nel proprio ambito – professionale, personale, familiare – e ad ogni età vogliano guadagnarsi una più profonda cultura informatica.
Proprio in quest'ottica, un'altra linea di sviluppo del progetto IT4PS si propone di analizzare in modo più approfondito le metodologie formative utilizzate nella sperimentazione, gli strumenti di auto-apprendimento realizzati e le tecnologie per la formazione a distanza disponibili (e più in generale tutto ciò che viene attualmente considerato supporto all'*e-learning*) per progettare percorsi formativi che riducano sempre più la necessità di una formazione frontale a vantaggio di una formazione autonoma. Se in ambito universitario questo significa diminuire la pressione sull'utilizzo delle strutture informatiche dei vari atenei – più o meno cronicamente carenti – in ambito di formazione continua questo consente di raggiungere in maniera decisamente più efficace e produttiva quella vasta utenza, la cui insufficiente preparazione all'uso degli strumenti informatici costituisce la principale voce di costo dell'ignoranza informatica del nostro Paese.

Ringraziamenti

Ringraziamo tutte le persone che, a vario titolo, hanno contribuito con il loro lavoro al successo del progetto. In particolare: Giulio Occhini, Franco Filippazzi (AICA) ed Emanuela Stefani (CRUI) per avere creduto nel progetto; il gruppo di Economia di Roma (Andrea De Checchi, Stefano Fontana, Claudio Guido Garzarelli, Giuseppe Sindoni, Mario Tirelli), coordinato da Paolo Atzeni; il gruppo di Medicina e Farmacia di Pisa (Adriano Martinelli, Vincenzo Gervasi, Paolo Manghi), coordinato da Antonio Brogi; il gruppo di Statistica di Milano (Daniela Bagnati, Noemi Viscusi, Silvia Salini), coordinato da Giovanna Nicolini; il gruppo di sviluppo (Annalina Fabrizio, Giuseppe Fiorentino, Michael Antonio Gargiulo, Anna Paola Pala, Andrea Pratesi, Lorenzo Valdambrini), coordinato da Giuliano Pacini; Emanuela Scalzotto (AICA), che ci ha seguito in tutta la fase degli esami di certificazione; lo staff CRUI (Monica Cantiani, Simona Giuffrida); Silvia Spazzacampagna e Paolo Bonfatti, che insieme a Nello Scarabottolo sono stati gli esaminatori per A-ECDL nella prima sperimentazione; tutti i docenti che hanno partecipato alla prima fase di sperimentazione: Giovanni Sculco, Agostino Gnasso, Emanuela Gualdi, Nicola Perfetti, Sergio Flesca, Nicoletta Fiore Pasquale, Caterina Guiot, Santina Di Giacomantonio, Carlo Vaccari, Luciano Fanfoni, Luisa Mich, Pietro Marzani, Elli Vassiliadis e in particolare Ilaria Bencivenni, che ha svolto un prezioso lavoro di collaudo dello strumento PS-Welcome.
Un ringraziamento particolare allo staff della Fondazione CRUI (Monica Cantiani, Simona Giuffrida e Moira Leo) che ha reso possibile la realizzazione di questo progetto.

Bibliografia

Fonti e approfondimenti

[1] Alfonsi C., Scarabottolo N., Pedreschi D., Simi M.: *IT4PS: Information Technology for Problem Solving*. Conference on Integrating Technology into Computer Science Education, Leeds, United Kingdom, 2004, Inroads, ACM SIGCSE Bulletin, ITiCSE 2004 Proceedings, Vol 36, n. 3, Sept 2004, p. 241.

[2] *Tecnologie dell'informazione per la risoluzione di problemi: il progetto IT4PS*. Atti di Didamatica 2005, Potenza, maggio 2005.

[3] Fiorentino G., Pacini G., Fabrizio A.: *Learning Problem Solving with Spreadsheet and Database Tools*. Conference on Integrating Technology into Computer Science Education, Leeds, United Kingdom, 2004, *Inroads*, ACM SIGCSE Bulletin, ITiCSE 2004 Proceedings, Vol 36, n. 3, Sept 2004, p. 267.

[4] Fiorentino G., Pacini G., Fabrizio A.: *Information Technology for Problem Solving: An approach to automatic assessment of open answers*. T.E.L.'04 Technology Enhanced Learning, Milano, Nov. 2004.

[5] *Il costo dell'ignoranza nella società dell'informazione*. Rapporto AICA e SDA-Bocconi, 2003.

[6] AICA: <http://www.aicanet.it/>

[7] CRUI: <http://www.crui.it/>

[8] ECDL Foundation, ECDL-A: <http://www.ecdl.com/main/adv_modules.php>

Testi prodotti dal progetto IT4PS

[9] Atzeni P., De Checchi A., Sindoni G., Tirelli M., Fabrizio A., Pacini G.: *Il foglio elettronico per Economia*. McGraw-Hill, 2005.

[10] Brogi A., Martinelli A., Gervasi V., Manghi P., Fabrizio A., Pacini G.: *Il foglio elettronico per Medicina e Farmacia*. McGraw-Hill, 2005.

[11] Bagnati D., Nicolini G., Viscusi N., Salini S., Fabrizio A., Pacini G.: *Il foglio elettronico per la Statistica nelle Scienze sociali*. McGraw-Hill, 2005.

[12] Atzeni P., De Checchi A., Sindoni G., Tirelli M., Fiorentino G., Pala A.P.: *Le basi di dati per Economia*. McGraw-Hill, 2006.

[13] Manghi P., Brogi A., Gervasi V., Martinelli A., Fiorentino G., Pala A.P.: *Le basi di dati per Medicina e Farmacia*. McGraw-Hill, 2006.

[14] Bagnati D., Nicolini G., Salini S., Viscusi N., Fiorentino G., Pala A.P.: *Le basi di dati per la Statistica nelle Scienze sociali*. McGraw-Hill, 2006.

[15] Fabrizio A., Fiorentino G., Pacini G.: *I sistemi autore PSWelcome e Access Test Manager*. McGraw-Hill, 2006.

Cristiana Rita Alfonsi dal 1996 è Responsabile dell'Unità Progetti, Servizi e Formazione della Fondazione CRUI, occupandosi in particolare della progettazione, della gestione e del coordinamento, del monitoraggio e della valutazione, della formazione del personale universitario non docente, dell'orientamento.
I progetti più importanti di cui si è occupata in questo periodo sono: Campus (1996-2000); Credits (1998-2000); Apollo (1999-2000); CampusOne (2000-2003); IT4PS (2003/2004); Eucip4U (2005/2007); B1-on-line (2005/2007).
Ha diretto e coordinato la ricerca sul tutorato universitario e il management didattico. Ha progettato l'Osservatorio sull'e-learning universitario in Italia ed è Responsabile del Progetto ELUE (E-Learning and University Education) co-finanziato dall'Unione Europea.
E-mail: alfonsi@fondazionecrui.it

Dino Pedreschi è professore ordinario di Informatica presso l'Università di Pisa dal 2000.
È stato professore visitatore presso la University of Texas at Austin, il centro di ricerca CWI ad Amsterdam e la University of California at Los Angeles.
Ha diretto vari progetti di ricerca nazionali, europei ed internazionali, ed è autore di oltre cento pubblicazioni su riviste e atti di convegni internazionali. I suoi interessi di ricerca sono centrati nelle basi di dati, ed in particolare rivolti all'analisi dei dati, al data mining, al ragionamento spazio-temporale, all'impatto del web sulle basi di dati (web mining, e-learning).
È stato docente in numerosi corsi sui linguaggi di programmazione, sulle basi di dati e sul data mining presso università italiane e straniere.
E-mail: pedre@di.unipi.it

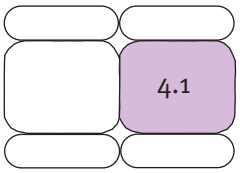
Nello Scarabottolo è professore ordinario di Informatica presso il Dipartimento di Tecnologie dell'Informazione dell'Università di Milano, collabora da tempo con la Fondazione CRUI e con AICA su progetti di definizione, diffusione e monitoraggio delle certificazioni ICT nelle università italiane.
Fa parte del Gruppo di Lavoro che ha realizzato l'Osservatorio Permanente delle Certificazioni Informatiche negli Atenei Italiani (http://osservatorio.consorzio-cini.it).
È Vice President del CEPIS (il Council of European Professional Informatics Societies) l'ente che riunisce le Associazioni europee di informatica, di cui AICA è la rappresentante per l'Italia.
E-mail: nello.scarabottolo@unimi.it

Maria Simi si è laureata in Scienze dell'Informazione presso l'Università di Pisa nel 1974.
Dal 1992 è professore associato presso l'Università di Pisa dove tiene corsi di Intelligenza artificiale e di Progettazione web. Dal 1979 al 1981 ha collaborato, presso il MIT di Boston, con il gruppo di "Message Passing Semantics" diretto dal Prof. Carl Hewitt. Più recentemente ha trascorso periodi di ricerca presso l'ICSI di Berkeley e il Computer Science Laboratory della Sony a Parigi.
I principali risultati di ricerca sono collocabili nell'ambito dell'intelligenza artificiale (rappresentazione della conoscenza, apprendimento automatico), dei sistemi informativi e servizi legati al Web (assistenti alla configurazione e commentatori on-line), question answering, estrazione di conoscenza da testi).
E-mail: simi@di.unipi.it

ICT GOVERNANCE
CHE COSA È?

Marco Bozzetti

La pervasività dell'ICT in ogni processo e attività lavorativa, nonché la crescita della complessità e la globalizzazione, rendono l'ICT una tecnologia abilitante che deve essere efficacemente ed efficientemente governata, ossia gestita in maniera coerente con gli obbiettivi e con le esigenze dell'Ente utilizzatore in un contesto di economicità e di generazione di valore. L'articolo evidenzia la logica del passaggio dalla gestione alla "governance" dei sistemi informativi e le modalità operative ormai consolidate dalle "best practices".



1. INTRODUZIONE

L'espressione inglese *ICT*[1] o *IT*[2] *Governance*, ossia governo dell'informatica e delle telecomunicazioni, sta iniziando a diffondersi sia nell'ambito informatico che nell'Alta Direzione degli Enti, ma il suo utilizzo è ancora embrionale o, come vedremo in seguito, limitato ad alcuni aspetti, prevalentemente tecnici.
Lo stesso termine d'altro canto, che non vede per ora l'uso diffuso della traduzione in italiano con "governo dell'IT o dell'ICT", è usato spesso in maniera generalistica e con significati diversi.
ICT Governance è derivato dall'analogo *Corporate Governance*, che include processi, pratiche, metodi e strumenti per il governo dell'intero Ente[3] e sta ad indicare il processo per il controllo e la direzione dell'ICT a garanzia del raggiungimento degli obiettivi dell'Ente stesso: in caso di Azienda, l'obiettivo primario è il business, in caso di Pubblica Amministrazione, l'obiettivo primario è l'efficace ed efficiente svolgimento dei propri compiti istituzionali.
Non esiste ancora una canonica e universalmente accettata definizione di *ICT Governance*, ma secondo la più qualificata ed accettata letteratura tecnico-manageriale, essa include le seguenti principali aree:

- allineamento strategico dell'ICT con il business e con le attività dell'Azienda/Ente;

[1] ICT, *Information and Communication Technology*, è l'acronimo ormai universalmente usato per indicare l'avvenuta convergenza delle tecnologie dell'informatica e delle telecomunicazioni, tecnicamente integrate nel trattamento di ogni tipo di informazione in una stringa di bit, dai dati alla voce, dalle immagini fisse a quelle in movimento. L'acronimo è usato per individuare l'intero settore dell'informatica, delle telecomunicazioni e della multimedialità.

[2] IT, *Information Technology*, è l'acronimo che indica le tecnologie informatiche, incluso il mondo degli applicativi.

[3] L'Ente in oggetto può essere un'azienda di qualsiasi dimensioni e settore, così come una Pubblica Amministrazione centrale o locale.

❑ controllo dei costi e del "valore" che l'uso dell'ICT genera o può generare;
❑ gestione dei rischi legati all'ICT ed ai suoi progetti di sviluppo;
❑ gestione delle risorse, intese nell'ampia accezione di infrastrutture ICT, programmi software, informazioni-conoscenza, persone e Società fornitrici-partner;
❑ gestione e misura delle prestazioni, sia strettamente dell'ICT sia, più in generale, del business e delle attività (il così detto BPM, *Business Performance Management*).

L'*ICT Governance* è quindi un processo o un insieme di processi, dinamico e continuo (ossia sistematico e ciclico), parte integrante del governo dell'Azienda/Ente (*Corporate Governance*), la cui responsabilità è del management esecutivo, in particolare del responsabile dell'intera azienda (in Italia tipicamente l'Amministratore Delegato o Amministratore Unico, nei Paesi anglosassoni il CEO, *Chief Executive Officer*) e del responsabile dei sistemi informatici, chiamato spesso con l'acronimo inglese CIO, *Chief Information Officer*.

Tale governo deve essere in grado di far fronte alle diverse prospettive dei differenti interlocutori dell'Ente, dai responsabili e dal personale interno fino ai referenti esterni, quali ad esempio i clienti e i fornitori, sia nel breve che nel medio-lungo termine. Le principali prospettive sono di tipo economico, di mercato, di ottimizzazione dei servizi agli interlocutori (per esempio, i servizi ai cittadini per le Pubbliche Amministrazioni), di miglioramento dei processi e della cultura-competenza del personale interno, oltre che, ovviamente, l'ottenimento di una gestione continuamente migliorativa (il così detto *continuous improving*) nell'erogazione dei servizi ICT. Le prospettive economiche, in particolare la creazione di valore per l'Ente, e/o per i suoi clienti-utenti, come ad esempio per le Pubbliche Amministrazioni e per le aziende fornitrici di servizi, unitamente alle prospettive di mercato, saranno in seguito delineate per fornire una visione generale del governo dell'ICT.

Esso si basa su una miscellanea di competenze tecniche e manageriali e include l'insieme di strumenti, procedure organizzative, metodi e metriche di misura e di controllo.

Al contrario, e spesso anche tra gli addetti ai lavori, per *ICT Governance* si intende il solo strumento informatico, tipicamente "il cruscotto" con varie finestre che riportano lo stato delle risorse, delle attività, degli utenti attivi, dei problemi e delle prestazioni, che consente di avere un quadro di sintesi per la gestione complessiva dei sistemi ICT.

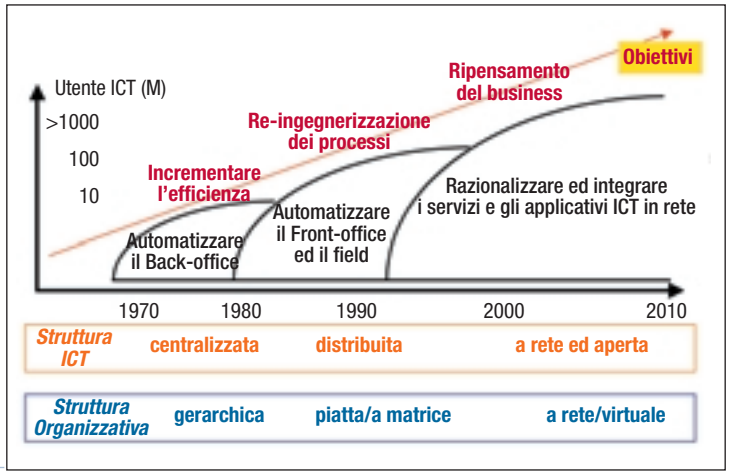
L'*ICT Governance* rappresenta, storicamente, la fase attuale dell'evoluzione della gestione dei sistemi, dei servizi e delle applicazioni ICT in un determinato contesto.

Con la nascita dell'informatica nasce anche il problema di controllarla e di gestirla: a livello di produzione del software, con la necessità di controllare la sua bontà-qualità, iniziando dalla documentazione chiara e sistematica del software sviluppato; a livello dei test funzionali e prestazionali, prima del passaggio dall'ambiente di sviluppo a quello di produzione, per verificare la rispondenza del software ai requisiti e alle aspettative dell'utenza e la sua effettiva interfacciabilità e interoperabilità con gli altri applicativi, dati e sistemi. E infine, il problema principe, quello di gestire l'intero sistema informativo.

Per molti decenni la gestione dei sistemi è stata principalmente un tema tecnico che si è andato man mano complicando con la diffusione dell'informatica distribuita e con l'avvento dei PC e di Internet. I singoli sistemi costituenti il sistema informativo sono divenuti sempre più eterogenei, distribuiti geograficamente e connessi fra loro via reti geografiche (WAN, *Wide Area Network*) e locali (LAN, *Local Area Network*).

La figura 1 schematizza le tre principali ondate dell'evoluzione dei sistemi informatici, a partire dagli anni '60 - '70. La crescita del-

FIGURA 1
Le tre principali fasi evolutive dell'informatica



la complessità è evidente: dai sistemi centralizzati basati su mainframe, con pochi utenti conosciuti e identificabili tramite la loro registrazione al sistema, a sistemi connessi ad Internet ai quali possono accedere milioni di utenti anonimi, con capacità e intenzoni sconosciute: da qui l'insorgere prepotente del fenomeno sicurezza. Il nuovo contesto è caratterizzato da forte eterogeneità di sistemi, soluzioni ed utenti, che devono operare tra loro: l'utilizzo dell'informatica diviene praticamente obbligatorio e la necessità di un governo dei sistemi cresce e si fa sempre più necessario e pressante.

2. LA NECESSITÀ DI UN GOVERNO

L'ICT è uno strumento a crescente complessità e pervasività in tutti i processi e nella struttura di qualsiasi organizzazione. Per le Aziende utenti dell'ICT, in particolare le PMI, così come per le piccole e medie Pubbliche Amministrazioni Locali, l'ICT è visto più come commodity[4] che come asset strategico. Le aziende italiane hanno investito e continuano ad investire in robotica e nei così detti *embedded system*, ossia nell'ICT all'interno di macchine, strumenti ecc.. In tali casi il ritorno è facilmente misurabile e a breve termine. Più difficile la gestione e la valutazione del ritorno aziendale per i "classici" sistemi di gestione aziendale, dai pacchetti di contabilità agli ERP[5] di nuova generazione (nei quali si includono anche i sistemi di CRM[6], SCM[7], PLM[8] e di SFA[9]) e la gestione degli strumenti di simulazione e di supporto alle decisioni manageriali. L'ICT è stato visto, nella maggioranza dei casi, come uno degli strumenti per ridurre i costi.
Pur con queste pregiudiziali, l'Alta Direzione e l'Amministratore dell'Ente si pongono con frequenza e con preoccupazione maggiore domande tipo: quale è il contributo reale che l'informatica fornisce al business o (per gli Enti Pubblici) alle attività della struttura? L'ICT è allineato a supportare il business e fornisce i servizi richiesti con i livelli attesi? La spesa per l'ICT è troppo limitata o troppo elevata? ecc..
Il Responsabile dei sistemi informativi (CIO), ha il problema di dover gestire tutte le risorse informatiche e di telecomunicazione e di rispondere alle crescenti richieste delle sue utenze interne, spesso non disponendo al proprio interno delle competenze necessarie e dei budget adeguati per terziarizzare. E, cosa più grave, non potendo il più delle volte interfacciarsi e confrontarsi direttamente con l'Amministratore, non essendo normalmente alle sue dirette dipendenze ma ad un livello inferiore nella struttura organizzativa[10].

---

4 Con il termine di *commodity* si indica un qualsiasi prodotto o servizio molto diffuso, uniforme e comune, acquisito per i suoi contenuti/caratteristiche e non per un particolare valore aggiunto, tanto che è spesso solo il prezzo il criterio di scelta.

5 ERP, *Enterprise Resource Planning*: applicativi software in grado di gestire in maniera integrata tutti i processi di un Ente, tipicamente tutta la parte amministrativa, finanziaria e di controllo, la gestione delle risorse umane, il controllo e la gestione della produzione, la gestione dei progetti. Gli ERP si stanno evolvendo includendo più ampie funzionalità, dal CRM alla SCM ed agli strumenti decisionali (indicati anche con il termine di *Business Intelligence*).

6 CRM, *Customer Relationship Management*: applicativo in grado di fornire una vista integrata dei dati sui clienti; esso consente di operare in maniera più efficace sui clienti acquisiti o potenziali; include normalmente le vendite, il marketing, il call center e funzionalità analitiche per l'analisi dei dati storici.

7 SCM, *Supply Chain Management*: applicativo che gestisce la catena dei fornitori, ossia la sequenza di aziende e processi che partecipano nella produzione e distribuzione di un prodotto, dall'acquisizione di materie prime alla consegna del prodotto finale. L'SCM include normalmente anche la logistica.

8 PLM, *Product Lifecycle Management*: applicativo che gestisce i dati dell'intero ciclo di vita di un prodotto, dalla sua pianificazione (*product planning*), alla sua progettazione e realizzazione (*manufacturing*).

9 SFA, *Sales Force Automation*: applicativo per la gestione del processo di vendita e dei venditori. Tipiche funzionalità SFA includono la gestione dei contatti e del calendario, la ripartizione delle vendite per settori/territorio, la gestione dei premi ecc..

10 Nella maggior parte delle aziende e degli Enti (anche pubblici) italiani il Responsabile dei sistemi informatici è posizionato al secondo livello della struttura organizzativa; tipicamente dipende o dal Responsabile risorse umane – organizzazione o dal Responsabile amministrativo, talvolta dalla Direzione servizi generali o dalla Direzione strategie. In aziende di grandi dimensioni ed alta tecnologia, sta emergendo il ruolo del Responsabile delle tecnologie, nel cui ambito viene collocato il Responsabile dei sistemi informatici.

Risulta quindi necessario un efficace ed olistico controllo dell'intero sistema ICT, affinché questo sia governabile, fornisca servizi in linea con gli obiettivi e le attività/business e sia capace di ridurre i costi di manutenzione e di fornire "valore". I principali obiettivi che un Ente si pone in termini di governo dell'ICT includono:

- ❑ porre l'ICT in linea con la strategia dell'Ente in modo che possa portare i benefici attesi e misurabili;
- ❑ migliorare l'erogazione dei servizi ICT agli utenti:
  - razionalizzando i criteri di scelta e di prioritizzazione delle richieste;
  - analizzando gli impatti sia sulle risorse disponibili, sia sui sistemi/applicativi, sui processi e sull'organizzazione;
  - garantendo un maggior e miglior controllo dello stato di avanzamento delle richieste sia per i Responsabili ICT sia per l'Alta Direzione;
- ❑ responsabilizzare maggiormente gli utenti sulle richieste e sul loro impatto tecnico ed economico;
- ❑ valutare e bilanciare i rischi ICT tecnici, organizzativi ed economici con il ritorno atteso;
- ❑ monitorare periodicamente e sistematicamente le prestazioni e l'uso dell'ICT.

3. IN CHE COSA CONSISTE IL GOVERNO DELL'ICT

Le soluzioni per il governo dell'ICT sono diverse e molteplici, ed evolvono con l'evolversi dell'azienda, della sua organizzazione e della tecnologia ICT.
Sul mercato non esistono consolidate soluzioni di riferimento: le soluzioni più avanzate possono essere viste come un "ombrello" sotto il quale sono integrati/integrabili diversi sotto-sistemi, ciascuno focalizzato a risolvere e a gestire uno specifico problema. Nell'ampia offerta disponibile, che non verrà tuttavia analizzata in questo articolo, si possono individuare due diversi approcci: il primo è la fornitura di moduli per espletare specifiche funzionalità, il secondo è la proposta di *suite* integrate che includono vari moduli funzionali e che promettono, modularmente, un governo completo dell'ICT.
La figura 2 schematizza un modello concettuale del governo dell'ICT: la linea tratteggiata ne delinea l'area di riferimento. Gli elementi del governo sono un insieme di processi, procedure organizzative e strumenti informatici che possono essere raggruppati sia per funzionalità che per ambiente ICT, distinguendo quest'ultimo in ambiente di sviluppo dei sistemi e dei programmi software, in ambiente di test e di collaudo e in ambiente di produzione.

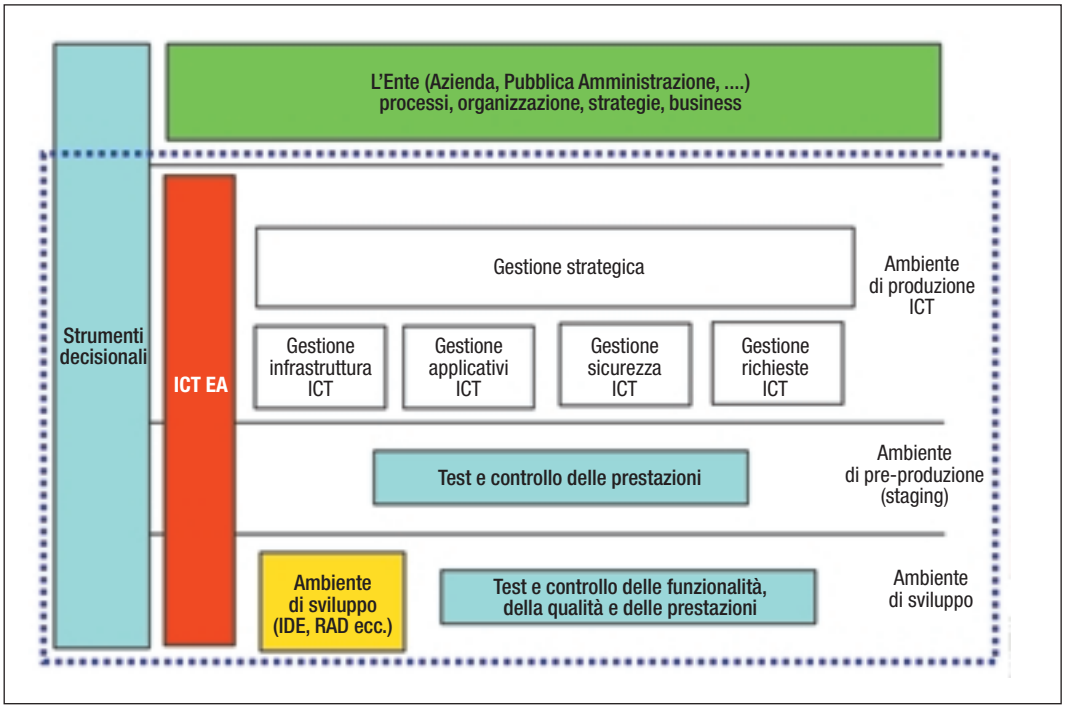


FIGURA 2
La visione concettuale dell'ICT Governance

A livello di *ambiente di sviluppo* gli elementi di governo includono i tipici strumenti per il controllo delle funzionalità, delle prestazioni e più in generale della qualità del software. Alcuni di questi strumenti sono inseriti negli ambienti "integrati" per lo sviluppo (nella figura indicati, come esempio di riferimento, con gli acronimi IDE, *Integrated Development Environment* e RAD, *Rapid Application Development*) altri sono specializzati per specifici controlli e funzionalità.
A livello di *ambiente di test e di pre-produzione*, il controllo funzionale e in parte prestazionale dell'ambiente di produzione viene testato su macchine diverse da quelle di produzione: la focalizzatone è sugli aspetti funzionali, di compatibilità e di interoperabiltà con gli altri moduli e sistemi dell'intero sistema ICT dell'Ente.
A livello di *ambiente di produzione* si possono individuare due principali gruppi di strumenti: il primo è l'insieme di strumenti per il controllo e la gestione dei sistemi e dei servizi ICT, delle loro prestazioni e della loro sicurezza; il secondo è l'insieme di strumenti per la gestione delle richieste delle utenze, siano esse di ordinaria amministrazione o strategiche, quali l'acquisto o la realizzazione di un nuovo applicativo. Tale secondo gruppo include gli strumenti di supporto decisionale per l'analisi del valore, l'analisi del rischio, la gestione del portafoglio applicativo, la gestione dei cambiamenti ecc.. Alcuni degli strumenti di supporto decisionale per l'ICT sono gli stessi usati per altri contesti, tipicamente gli strumenti per l'analisi del rischio, la gestione dei progetti, la stima del ritorno dell'investimento, la gestione dei cambiamenti e delle prestazioni del business.
Concettualmente e funzionalmente l'ICT governance è visto quindi come un ampio "ombrello" sotto il quale possono essere considerati ed inseriti innumerevoli processi e relativi strumenti, suddivisibili in due grandi aree:
- ❑ processi e strumenti per la *gestione operativa* di tutti gli ambienti di cui sopra, ma in particolar modo dell'ambiente di produzione;
- ❑ processi e strumenti per la *gestione strategica*, in altri termini per effettuare valutazioni e per prendere decisioni.

Facendo riferimento al solo ambito di produzione, i processi e gli strumenti operativi possono essere raggruppati, seguendo prevalentemente le "best practice" dell'ITIL[11], nelle seguenti aree di "gestione" (si veda per maggiori dettagli sottoparagrafo 5.1.):
- gestione delle infrastrutture ICT: hanno il compito di gestire e monitorare le risorse ICT, dalle reti ai server e ai PC/client e le loro prestazioni;
- gestione delle applicazioni;
- gestione della sicurezza;
- gestione dei servizi, a sua volta articolata in processi e strumenti per il supporto e per l'erogazione dei servizi. In questa area sono incluse importanti attività quali la gestione delle richieste ordinarie e straordinarie, il controllo dei livelli di servizio (Sla, Service Level Agreement), la gestione delle licenze dei vari software utilizzati.

Nei paragrafi seguenti verranno approfondite solo alcune di queste aree. Non si tratteranno in particolare gli ambiti di sviluppo, di staging e di pre-produzione, e per l'ambiente di produzione la pur fondamentale gestione della sicurezza.
Un ruolo determinante e indispensabile per l'ICT governance è svolto dall'architettura dei sistemi informatici, vista come il quadro di riferimento di tutte le risorse hardware, software e di rete.
Tale architettura, che nel seguito indicheremo con l'acronimo inglese di EA, *Enterprise Architecture*, è un prerequisito per poter realmente attuare un governo dell'informatica; a sua volta il governo impatta e condiziona lo sviluppo dell'EA, come verrà analizzato nel paragrafo 4.

3.1. L'organizzazione ed i processi dell'ICT

Il tipo, la struttura e il posizionamento dell'unità organizzativa che ha in carico la gestione dell'ICT dipende da molteplici fattori, sia strategici che organizzativi e tecnici: dall'organigramma dell'Ente, dalle sue dimensioni e dal-

[11] ITIL, *Information Technology Infrastructure Library*, sviluppato dall'Office of Government Commerce (www.ogc.gov.uk/) britannico, è un insieme di indicazioni e di linee guida per organizzare ed erogare al meglio (best practice) i Servizi Informatici. È da considerarsi un "modello" e non un metodo, poiché non fornisce indicazioni operative univoche e vincolanti: le sue norme di riferimento vanno adattate alle situazioni operative di ogni singolo Ente.

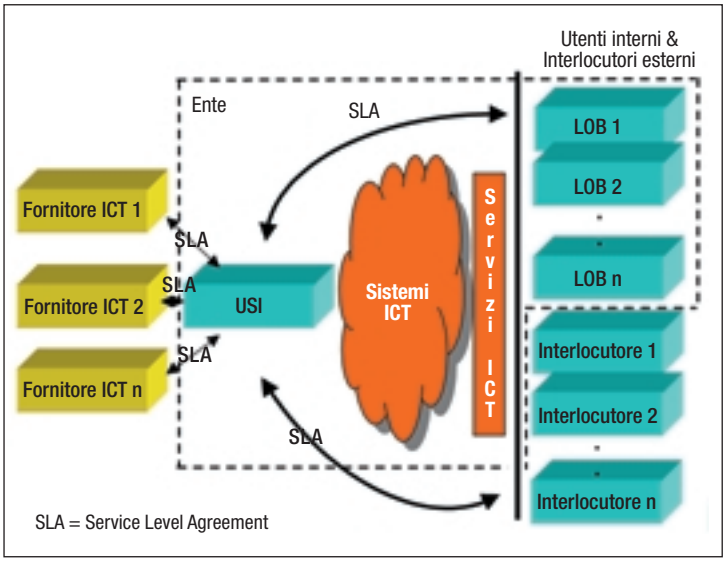


FIGURA 3
L'orientamento dell'USI ai servizi ICT

la sua articolazione sul territorio, dalla complessità e criticità dei sistemi informatici, dai processi aziendali in essere, dalle strategie di sviluppo, dai compiti assegnati a questa struttura (esercizio dei sistemi informatici, acquisto hardware, software e competenze, sviluppo software ecc.) e dalle funzioni o dai processi che sono eventualmente terziarizzati.
La terziarizzazione, spesso chiamata anche in italiano con il termine inglese di "outsourcing", è stato uno dei fenomeni che ha favorito la trasformazione dell'Unità organizzativa dei Sistemi Informatici (indicata per brevità nel seguito con l'acronimo *USI*) in un fornitore di servizi ICT ed in un centro di competenza ICT per i propri utenti. La terziarizzazione richiede un contratto tra l'Ente appaltante e la Società di outsurcing che fornisce i servizi, e la definizione di misure e logiche di controllo della qualità del servizio prestato, ossia la definizione e l'accordo sui livelli di servizio (SLA, *Service Level Agreement*) che il fornitore deve erogare. Questa necessità ha portato poi alla definizione di SLA tra la USI e le varie divisioni interne e i relativi utenti finali del sistema informatico. Con o senza un outsourcing, l'USI si è già trasformata o si trasformerà[12] in un fornitore di servizi ICT ai diversi dipartimenti/divisioni e ai relativi utenti, oltre che ai diversi interlocutori dell'Ente stesso (Figura 3): per far questo deve definire opportuni parametri e metriche per la valutazione dei livelli di servizio offerti e per come sono recepiti dalla sua utenza e attivare poi gli opportuni strumenti informatici che consentano tali misurazioni (per maggiori dettagli si veda il sottoparagrafo 5.1.1.).
La struttura organizzativa dell'USI dipende in primo luogo dai processi che deve supportare. I processi ICT sono stati standardizzati a livello internazionale da vari Enti: tali standard sono per lo più focalizzati allo sviluppo del software e alla sua gestione, oltre che complementari agli standard per la gestione della qualità totale. Sono standard più orientati alla definizione del "che cosa" piuttosto che del "come". Ed è per questo che l'ITIL, orientato a specificare come si deve gestire l'ICT, ha iniziato a riscuotere un forte interesse.
Gli standard che hanno trovato maggior rispondenza applicativa e che sono abbastanza diffusi e consolidati includono, oltre al già citato ITIL:

- ISO12207[13]: definisce i processi per l'intero ciclo di vita del software, suddividendoli in tre categorie: processi primari per lo sviluppo, processi secondari e processi organizzativi. Per ognuno di tali processi sono definiti obiettivi e responsabilità, la lista delle attività che li compongono e i tasks nei quali si articola ogni singola attività;
- ISO 15288, Systems Engineering - System life cycle processes: definisce il ciclo di vita di un generico "sistema", e quindi anche di uno informatizzato e i vari processi relativi alle varie fasi, dalla progettazione allo sviluppo, alla produzione, all'utilizzo e alla messa in servizi, fino al ritiro dal mercato;
- ISO15504, chiamata SPICE, *Software Process Improvement and Capability Evaluation*: è l'evoluzione dell'ISO 12207, ed aggiunge ad esso l'individuazione delle figure professionali coinvolte, l'elenco degli elementi in ingresso ed in uscita da ogni task con la descrizione del loro contenuto. Questo standard introduce logiche di valutazio-

[12] Tale processo è in parte avvenuto in Italia per le grandi aziende/enti, ed è in corso o lo sarà per le aziende/enti di medie e piccole dimensioni.

[13] ISO, *International Organization for Standardization* (www.iso.org).

ne della maturità dei processi e delle aree di miglioramento. È una normativa dettagliata e abbastanza complessa.
Altri standard de-jure e de-facto sui processi per la gestione del software e dei processi ICT, molti anche per la qualità, includono SEI-CMM, MIL-STD-498, EIA 632, EIA 731, ISO 1220.
Tutti gli standard citati sono orientati alla gestione del software e del prodotto, alla sua qualità e al suo ciclo di vita, ma specificano anche i processi tipici per la gestione dei sistemi, processi che sono significativi per l'USI.
Nell'informatica moderna lo sviluppo di software all'interno di una USI è molto ridotto rispetto al passato, data anche l'ampia disponibilità e diffusione di pacchetti software: nella maggior parte dei casi non solo lo sviluppo, ma anche la configurazione e la personalizzazione di un software è demandata al fornitore o ad un consulente.
Il modello dei processi USI è strettamente legato alle procedure organizzative interne ed esterne attuate. A grandi linee i processi si possono suddividere in:
a. processi orientati ai progetti: molti degli interventi richiesti alla USI sono di tipo progettuale, e parte di questi sono effettuati esternamente;
b. processi orientati alla erogazione e alla gestione dei servizi.
Una diversa ma importante distinzione dei processi ICT li distingue tra quelli che vedono coinvolto direttamente l'utente finale, che chiameremo per questo di *front-office* da quelli che non vedono l'utente direttamente coinvolto e che chiameremo di *back office*. I primi includono tutti i processi e le attività di esercizio ed erogazione dei servizi, la progettazione delle soluzioni con il loro sviluppo, installazione e test. I secondi includono la pianificazione, il budget ed il controllo, gli acquisti ed i rapporti coi fornitori, la gestione del personale interno all'USI, la definizione e gestione dell'architettura ICT, delle procedure e delle metodologie.
Una moderna USI orientata ai servizi ICT dovrebbe essere strutturata in centri di competenza per:
- la gestione delle infrastrutture hardware e software e di telecomunicazione e rete;
- la gestione degli applicativi;
- il monitoraggio ed il controllo sistematico e continuativo dei sistemi sia centralizzati che distribuiti e delle loro prestazioni; il monitoraggio è effettuato normalmente tramite la misura sistematica di indicatori chiamati KPI, *Key Performance Indicator*;
- il supporto giornaliero agli utenti tramite help-desk.

Ulteriori funzioni all'interno dell'USI riguardano le strategie, gli acquisti, gli aspetti legali, l'ottemperare (compliance) alle varie normative e disposizioni di legge, ed eventualmente lo sviluppo di software.
Le interazioni tra USI e gli utenti finali e le loro unità organizzative devono (o dovrebbero) essere regolati da opportuni SLA, *Service Level Agreement*, misurabili e verificabili: per un approfondimento si rimanda al sottoparagrafo 5.1.1.

4. L'ICT ENTERPRISE ARCHITECTURE

Come evidenziato nella figura 2, l'architettura dei sistemi ICT, (EA), è un mattone fondamentale per il governo dell'ICT. Ma che cosa si intende per EA? Architettura è un termine generico, usato nell'ambito ICT con diversi significati e con diverse valenze, dagli schemi di un circuito integrato, a quelli di una CPU, dalla scheda madre in un PC ad un intero dispositivo (server, PC, cellulare, router ecc.), dallo schema di un'intera rete trasmissiva a quella dell'intero sistema informativo. Indipendentemente dalle varie accezioni, il termine architettura implica sempre un approccio ed una logica di sistema, ossia di più elementi o parti tra loro interagenti, in modo che l'insieme delle funzionalità del sistema sia maggiore della somma delle funzionalità dei singoli componenti.
Non esiste una univoca o consolidata definizione di EA, si può tuttavia definirla come "l'insieme di principi, delle politiche e delle scelte tecniche per i sistemi ICT di un Ente, nell'ottica di standardizzazione interna sia tecnica sia organizzativa e di richieste di integrazione per l'operatività dell'Ente stesso". L'EA definisce gli standard interni all'Ente ma deve evolversi dinamicamente e flessibilmente in funzione dell'evoluzione

del business, dei processi, della tecnologia e del mercato ICT, in modo da:

- assicurare la compatibilità dei sistemi e degli applicativi;
- rendere gestibile la complessità e l'eterogeneità dei sistemi stessi;
- salvaguardare gli investimenti in ICT già fatti;
- permettere uno sviluppo controllato e congruente con gli obiettivi dell'Ente e con gli obiettivi di business per le Aziende o di servizi per le Pubbliche Amministrazioni.

È importante sottolineare come l'EA non sia solo un insieme di standard tecnici, un piano tecnologico o di gestione del portafoglio applicativo, ma debba includere le logiche ed i regolamenti-procedure per le interazioni tra i vari interlocutori interni ed esterni dell'Ente che interagiscono con il suo sistema informatico.

La figura 4 schematizza le relazioni logiche tra l'EA, il tipo di business ed i processi dell'Ente (il suo "business model"), l'organizzazione ed i processi dell'USI, i trend tecnologici e di mercato dell'ICT, le leggi e le norme da rispettare.

In maniera molto schematica, l'EA può essere considerata costituita da tre macro-livelli, Infrastrutture, Applicazioni e Informazioni (dati), che danno origine all'*architettura delle infrastrutture e all'architettura applicativa* (Figura 5).

L'architettura infrastrutturale include:

- ❑ le infrastrutture di comunicazione per i vari tipi di informazione;
- ❑ le piattaforme hardware e del software di ambiente per il funzionamento fisico e logico delle diverse unità, delle diverse piattaforme e dei diversi sottosistemi spesso periferici; tale software è chiamato middleware ed include:
  - sistemi operativi;
  - sistemi per la gestione dei file e dello storage;
  - le banche dati.

L'architettura applicativa è costituita dall'insieme dei programmi applicativi, di qualsiasi genere e operanti e/o cooperanti su diverse piattaforme, oltre che dalle informazioni multimediali che tali programmi trattano.

Anche per le architetture sono stati definiti degli standard internazionali di riferimento, alcuni molto noti, altri poco conosciuti anche dagli

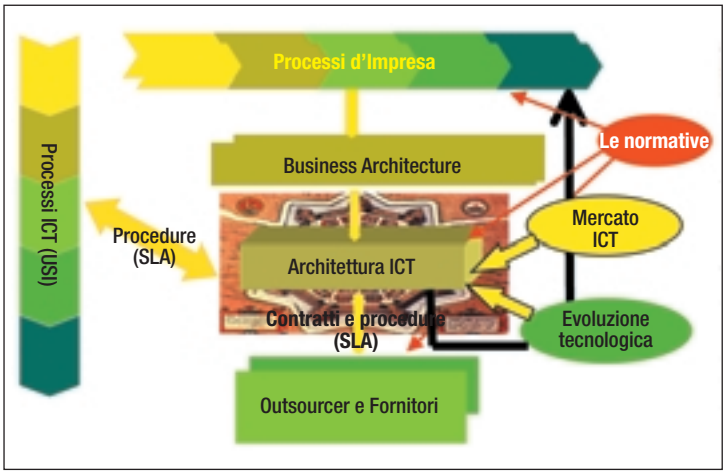


FIGURA 4

Il quadro di riferimento dell'Enterprise Architecture (EA)

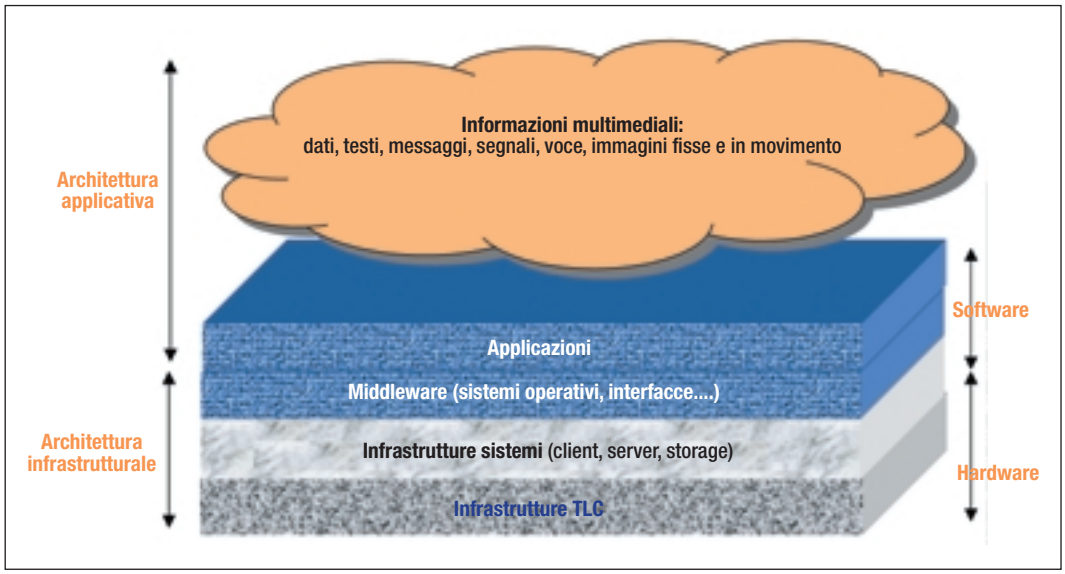


FIGURA 5

I macro componenti dell'architettura tecnica ICT

addetti ai lavori. Lo standard architetturale più noto è il modello OSI, *Open System Interconnection*, definito in ISO 7498, che ha strutturato funzionalmente i protocolli di comunicazione in 7 strati, da quello di linea a quello applicativo. La sua importanza e diffusione sono dovute al fatto che tale modello è stato adottato per l'inquadramento di tutti i protocolli realizzati con l'evolversi delle tecnologie, dalle LAN (*Local Area Network*) ai sistemi ATM (*Asynchronous Transfer Model*) ed ISDN (*Integrates Services Digital Network*) fino all'architettura Internet basata sul protocollo TCP/IP. La figura 6 confronta la struttura a livelli del modello OSI con quella TCP/IP tipica di Internet.

Più recenti modelli architetturali standard hanno ripreso la struttura a livelli OSI e hanno dettagliato ed articolato gli strati più alti del modello.

Sempre l'ISO ha emanato nel 1999 uno standard sull'EA "IS 15704: Requirements for Enterprise Reference Architecture and Methodologies, ISO TC 184/SC5/WG1", che risulta però troppo generico ed informale e non adatto ad essere utilizzato per effettive implementazioni.

Tra i modelli architetturali più significativi ed usati vi sono: il modello TAFIM, *Technical Architectural Framework for Information Management*, e il TOGAF, *The Open Group Architecture Framework*. Altri modelli consolidati, referenziati nella bibliografia, includono il *Zachman Framework*, lo statunitense FEAF, *Federal Enterprise Architecture Framework*, il GERAM[14], *Generalised Enterprise Reference Architecture and Methodology*, sviluppato da una task force congiunta dell'IFIP[15] e dell'IFAC[16], IEEE 1471[17] (2000) *Recommended Practice for Architectural Description of Software-Intensive Sistems*.

TAFIM è stato sviluppato da una agenzia[18] del Dipartimento della Difesa statunitense per guidare l'evoluzione dei propri sistemi ICT, incluse le interfacce con i sistemi d'arma. TAFIM ha l'obiettivo di creare un'architettura ICT totalmente flessibile ed interoperabile in un ambiente fortemente eterogeneo e di facilitare l'evoluzione dalle appli-

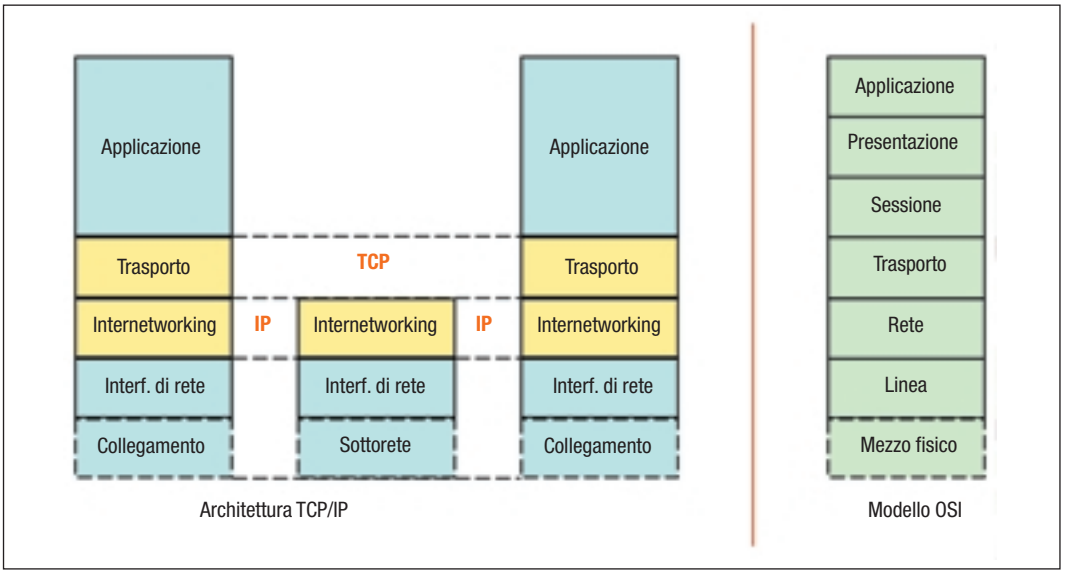


FIGURA 6
Architetture a strati: il confronto tra il modello OSI e quello TCP/IP

[14] GERAM, nella attuale versione 1.6.3, è inserito come Allegato nello standard ISO 15704 (2000).

[15] L'IFIP, *International Federation for Information Processing*, è la federazione mondiale, sotto l'egida dell'Unesco, di 48 associazioni nazionali e accademie: per l'Italia il "full member" è AICA. L'IFIP ha attivi 12 Comitati Tecnici suddivisi a loro volta in gruppi di lavoro, che organizzano conferenze, gestiscono incontri di lavoro e pubblicano documenti tecnici. Le attività sono strettamente tecniche scientifiche e vi partecipano esperti prevalentemente del mondo accademico.

[16] IFAC, *International Federation of ACcountants*, è la federazione mondiale che raggruppa alla data 163 associazioni nazionali dei revisori dei conti, per l'Italia il Consiglio Nazionale dei Dottori Commercialisti e Consiglio Nazionale dei Ragionieri e Periti Commerciali.

[17] IEEE, *Istitute of Electrical an Electronics Engineers* (www.ieee.org).

[18] L'agenzia è la DISA, *Defense Information Systems Agency*.

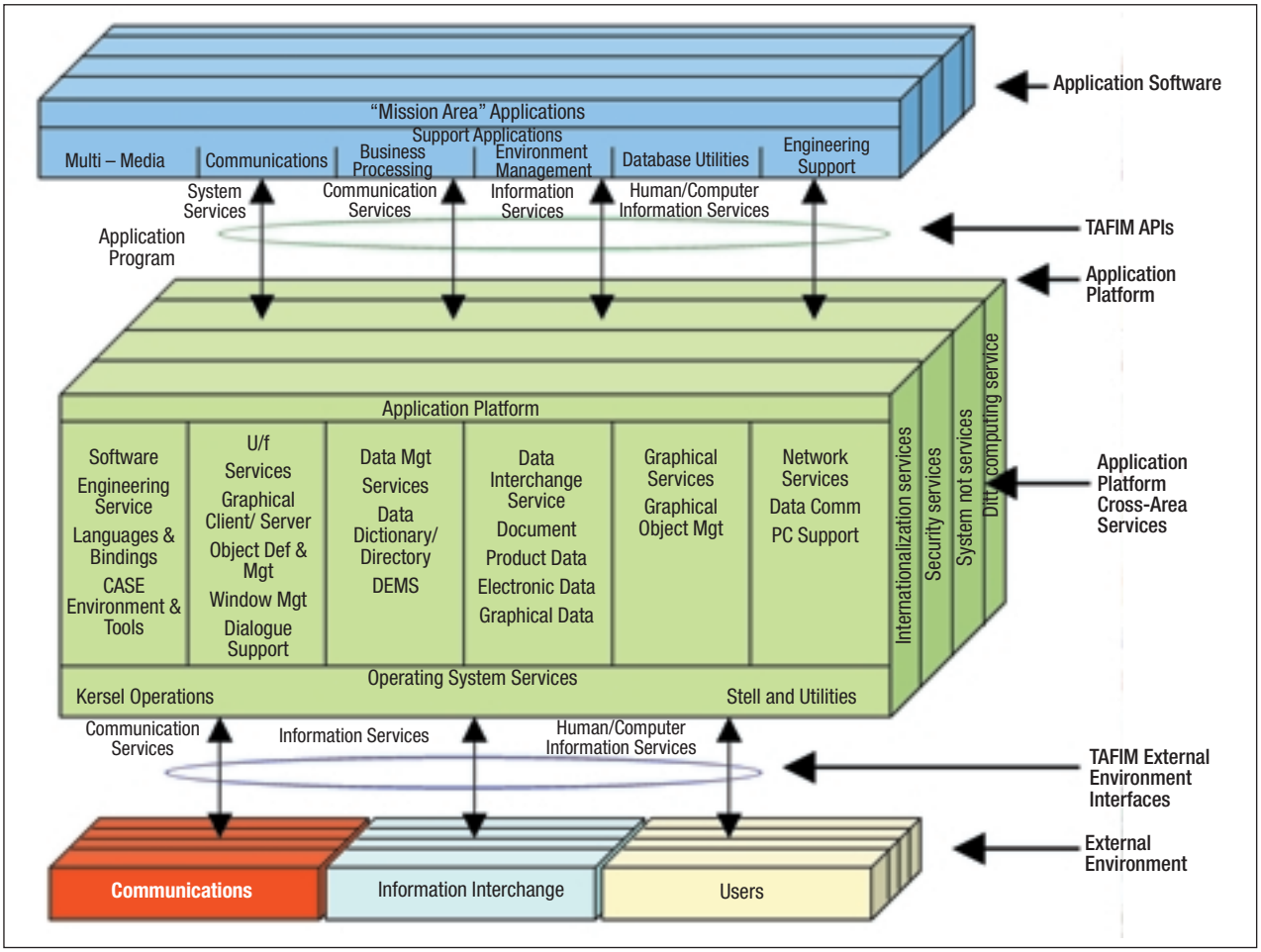


FIGURA 7
Modello di riferimento TAFIM, Technical Architectural Framework for Information Management
(Fonte: Tafim)

cazioni legacy ad applicazioni distribuite. Il modello di riferimento (Figura 7) definisce i servizi, i componenti e le relative interfacce da utilizzare per la progettazione, l'implementazione e l'installazione dei sistemi richiesti.

TOGAF è stato sviluppato dal Consorzio internazionale *The Open Group*: oggi è alla versione 8.1 e costituisce una quadro di riferimento (*framework*) per consentire la progettazione e la realizzazione della corretta EA informatica necessaria ed adatta a supportare il business. Non definisce quindi solo l'architettura ma anche la metodologia per costruirla, tenendo conto della specifica realtà dell'Ente/Azienda in termini di organizzazione, processi e attività. Per tale motivo TOGAF definisce e supporta quattro sotto insiemi architetturali: l'architettura dei processi di business, l'architettura applicativa, l'architettura dei dati e l'architettura infrastrutturale.

TOGAF si articola in quattro parti principali: l'introduzione, che inquadra i concetti base e la logica di approccio, l'*Architecture Development Method* (ADM), che dettaglia fase per fase[19] come realizzare l'architettura in

[19] L'ADM di Togaf prevede le seguenti 9 fasi: la prima, preliminare, introduce i concetti e le logiche di base. Dalla seconda in avanti le fasi sono indicate come A, B ecc.. La fase A (*Architectural Vision*) definisce la visione e lo scopo per cui si realizza la EA. La fase B (*Business Architecture*) definisce l'architetura di business attuale e quella prospettica (*target*), la fase C (*Information systems architecture*) sviluppa le architetture applicative e dei dati, la fase D (*Technology architecture*) riguarda lo sviluppo di tutta l'infrastruttura con le relative scelte tecnologiche, la fase E sviluppa l'intera strategia per l'EA, determinando che cosa acquisire, sviluppare o riusare e come implementare quanto definito nella precedente fase D. La fase F sviluppa il piano di migrazione dalla situazione attuale a quella prevista dalla EA, specificando le priorità dei diversi progetti. La fase G specifica come controllare e gestire l'intero sviluppo dell'EA mentre la fase H, l'ultima, considera come far evolvere l'EA al cambiare delle necessità e delle tecnologie.

funzione dell'organizzazione e dei requisiti di business specifici, l'Enterprise Continuum, il repository virtuale di tutti gli elementi necessari ed utili per costruire l'architettura, quali modelli, patterns, schemi architetturali ecc.. Di particolare importanza in questo repository vi sono: il TOGAF Foundation Architecture, il cuore dell'architettura che specifica servizi e funzioni e fornisce riferimenti ai principali standard industriali[20] e l'Integrated Information Infrastructure Reference Model, di ausilio per il flusso informativo tra sistemi ed ambienti diversi. L'ultima parte, chiamata TOGAF Resource Base, fornisce un insieme di strumenti di ausilio per l'ADM, quali linee guida, schemi, informazioni di background ecc.. La figura 8 schematizza il modello TOGAF.

Gli attuali modelli architetturali si basano sulla pila di protocolli standard TCP/IP per la comunicazione multimediale, su vari linguaggi ed ambienti di sviluppo per il software, tutti ormai basati sulla programmazione ad oggetti, su J2EE[21], o su .Net o su CORBA[22], per finire con i più recenti web services[23] e SOA[24]. Le moderne architetture ICT devono essere in grado di integrare e gestire i vari tipi di informazione, dai dati ai testi, dalla voce alle immagini fisse e in movimento, ora tutti digitalizzabili[25] e trattabili con differenti dispositivi per l'utente finale, fissi e mobili: PC, fax, palmari, telefoni, cellulari, dispositivi specializzati per specifiche funzioni. A livello architetturale, la multimedialità non solo comporta la gestione di dispositivi diversi, ma richiede che la gestione dell'informazione sia di tipo multimediale, garantendo la più ampia flessibilità nella reperibilità e nell'analisi dei dati di interesse dei processi dell'Ente.

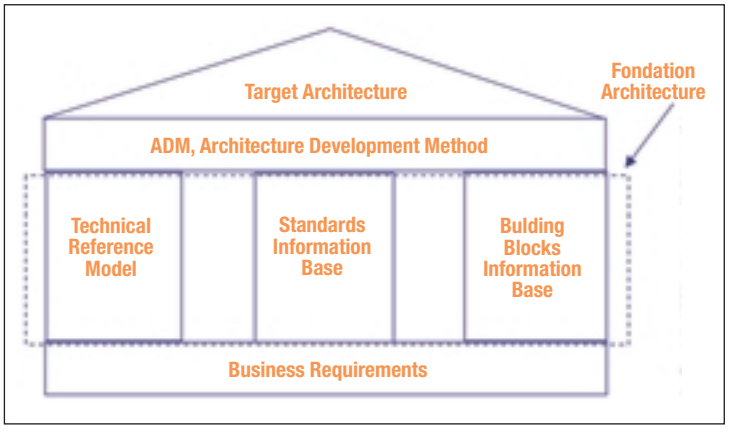


FIGURA 8
Modello di riferimentoTOGAF, The Open Group Architecture Framework
(Fonte: The Open Group)

La figura 9 mostra un esempio di macro componenti per l'architettura applicativa di una moderna azienda che si interfaccia ed interagisce anche a livello informatico con tutti i suoi interlocutori, dai clienti ai fornitori.

La figura evidenzia, anche se in maniera schematica e qualitativa, la valenza di back o front-end degli applicativi; al centro la figura mostra la banca dati che dovrebbe essere comune alla maggior parte delle applicazioni e costituire la banca dati della conoscenza dell'Ente stesso. In realtà, anche nelle più moderne architetture applicative, esistono diverse banche dati per ogni ambien-

[20] TOGAF fornisce una banca dati, chiamata *Standards Information Base* (SIB), contenente gli standard industriali che possono essere usati per definire uno specifico servizio o altri componenti dell'architettura.

[21] *Java2 Enterprise Edition* è la piattaforma di sviluppo con il linguaggio Java versione 2 (attuale standard) per l'ambiente "enterprise", ossia professionale e aziendale.

[22] CORBA, *Common Object Request Broker Architecture*, è lo standard sviluppato dall'OMG, *Object Management Group*, per far comunicare ed interagire componenti software indipendentemente dalla loro distribuzione nella rete e dal linguaggio di programmazione con cui sono stati sviluppati.

[23] Con il termine di *web services* si fa riferimento ad un insieme di interfacce programmabili che consentono un'interazione applicazione-applicazione nell'ambito web.

[24] SOA, *Service Oriented Architecture*, è un'architettura di riferimento standard basata sui web services ed indipendente dalle piattaforme hardware, dal software di base e dai linguaggi di programmazione. Consente l'effettiva interoperabilità tra applicazioni viste in rete come servizi. La SOA è alla base della nuova architettura per l'interoperabilità dei sistemi informatici delle Pubbliche Amministrazioni in Italia, chiamata SPC, *Sistema Pubblico di Cooperazione*. Lo sviluppo delle specifiche SOA è effettuato dal consorzio OASIS (*Organization for the Advancement of Structured Information Standards*) ed dal World Wide Web Consortium (W3C) cui si affianca l'organizzazione WS-I per migliorare l'interoperabilità tra le diverse implementazioni dei Web Service e per rendere più facile l'implementazione tramite una serie di "profili".

[25] Ossia codificabili in stringhe di bit.

te applicativo. È interessante sottolineare come le interazioni applicativo-applicativo, sono le prevalenti, ed in significava crescita rispetto a quelle applicativo utente-essere umano.
L'EA riguarda non solo i sistemi *in produzione* (chiamati anche "in esercizio"), ossia quelli che supportano gli applicativi ed i servizi informatici utilizzati dagli utenti, ma anche gli ambienti di *sviluppo* e di *test* e collaudo del software.
L'EA costituisce il piano regolatore dell'evoluzione del sistema informativo, e come tale deve definire non solo gli standard interni di tecnologie, prodotti e servizi, ma fornire anche gli archi temporali della loro validità.
Con *standard interno* si intende uno standard di tecnologia, prodotto e servizio contemplato nell'EA il cui *non uso* deve essere giustificato. In altri termini l'EA definisce i prodotti ed i servizi ICT che devono essere utilizzati: se un determinato ufficio ha esigenza di utilizzare per il proprio lavoro un sistema o un servizio non contemplato nell'EA, dovrà non solo farne esplicita richiesta al responsabile del sistema informatico, ma soprattutto dovrà esplicitare le motivazioni per cui non può o non è conveniente che utilizzi i sistemi previsti. Risulta evidente che se le motivazioni sono giustificate, il sistema o servizio richiesto potrà essere inserito tra gli standard interni dell'EA.

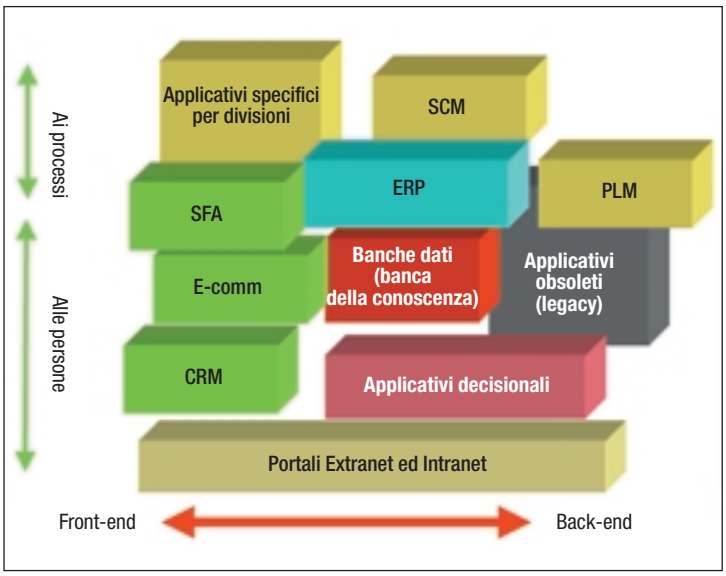


FIGURA 9
Esempio di macro-componenti dell'architettura applicativa

5. L'OPERATIVITÀ DELL'ICT GOVERNANCE

Come già evidenziato nei paragrafi precedenti, l'ICT governance include, o dovrebbe includere, vari strumenti e processi sia per la gestione operativa che per la gestione strategica dell'ICT, visto nell'ottica di tecnologia abilitante ai diversi processi e attività dell'Ente.
Alcuni strumenti di ausilio decisionale sono usati anche per talune attività operative, e alcune parti della gestione operativa sconfinano e/o si sovrappongono a quella strategica. La stessa suddivisione tra gestione operativa e strategica è puramente indicativa e tale confine può essere spostato a seconda del livello di delega e della volontà decisionale dell'Alta Direzione verso il Responsabile dei sistemi informatici.
Nel seguito verrà considerato solo l'ambito di produzione ed in tale contesto i più importanti processi e strumenti operativi per l'ICT Governance possono essere raggruppati in due aree: la prima relativa alla gestione operativa dei sistemi e dei servizi, che include la gestione delle prestazioni e degli utenti del sistema informatico, la seconda relativa alla gestione del portafoglio applicativo. La prima area include una ampia serie di processi e di servizi, ed è prevalentemente orientata al controllo dei sistemi ICT, inclusi gli applicativi ed i servizi che essi offrono all'utenza, con il primario obiettivo di soddisfare le richieste di tale utenza in termini di livello di servizio e di ridurre i rischi di non soddisfarli, rischi che avrebbero impatto sulle attività dell'Ente stesso.
La seconda area, pur inquadrata nell'operatività del governo dell'ICT, ha una valenza strategica: la gestione del portafoglio applicativo (che non deve essere confusa con la gestione degli applicativi, inclusa nella prima area) è un'insieme di processi e di attività decisionali, riguardanti l'acquisizione di nuovi pacchetti applicativi e lo sviluppo di nuove applicazioni o la modifica significativa[26] di applicativi esistenti. Tali decisioni implicano l'attivazione di un progetto che ne verifichi la fattibilità, la

[26] La modifica "significativa" di un applicativo richiede un numero considerevole di risorse e non può rientrare nella gestione operativa che riguarda piccoli cambiamenti, o cambi di release, sul parco applicativo in essere.

convenienza anche economica, il rischio, l'impatto sugli altri applicativi e sui processi sia dell'USI che dell'intero Ente. Il responsabile dei sistemi informatici, e più in generale l'Alta Direzione, devono valutare nell'ottica del miglioramento dell'efficacia e dell'efficienza dell'intera struttura organizzativa, e valutare i costi ed il ritorno economico. Tutti questi tipi di analisi e di decisioni rientrano nella gestione del portafoglio applicativo.
Si tratta di una gestione a cavallo tra l'operativo e lo strategico: oltre alla valutazione e all'eventuale approvazione dei vari progetti, si deve porre sistematicamente il problema del modo migliore di utilizzare il budget ICT e tutte le risorse dell'USI, con attenzione ai costi e all'analisi del rischio e del valore.

5.1. La gestione operativa dei sistemi e dei servizi ICT

La gestione operativa dei sistemi e dei servizi informatici ha impatto su tutta l'organizzazione, ma in particolare su quella dell'ICT e sulla sua conduzione. I processi che riguardano tale gestione (si veda il sottoparagrafo 3.1.) devono essere chiaramente definiti e fatti conoscere, devono essere misurabili tramite opportune metriche e devono essere automatizzati quanto più possibile.
La gestione dei sistemi ICT è nata con la nascita dell'informatica stessa, ma la crescente pervasività e complessità dell'ICT la rendono più difficile e sofisticata. Fin dai suoi albori il mercato informatico ha sviluppato sistemi per il controllo delle funzionalità e delle prestazioni degli elaboratori, sistemi che man mano si sono evoluti e si stanno ora orientando al governo dell'intero ICT. Ma è bene partire non da un prodotto o da una soluzione tecnologica, ma dalla logica e dall'organizzazione che si vuole dare al governo ICT nello specifico contesto dell'Ente che lo utilizza. Occorre ritagliare il governo dell'ICT nell'ambito del più generale governo (la cosi detta governance) dell'Ente in questione, calandolo nella specifica realtà composta da attività, processi, obiettivi, storia, cultura e attitudine delle persone che vi operano. L'approccio e la logica da seguire per la gestione operativa dell'ICT non deve essere strettamente e/o solamente tecnica, ma deve puntare all'impatto sulle attività e sul modus operandi dell'Ente, e quindi al suo business.

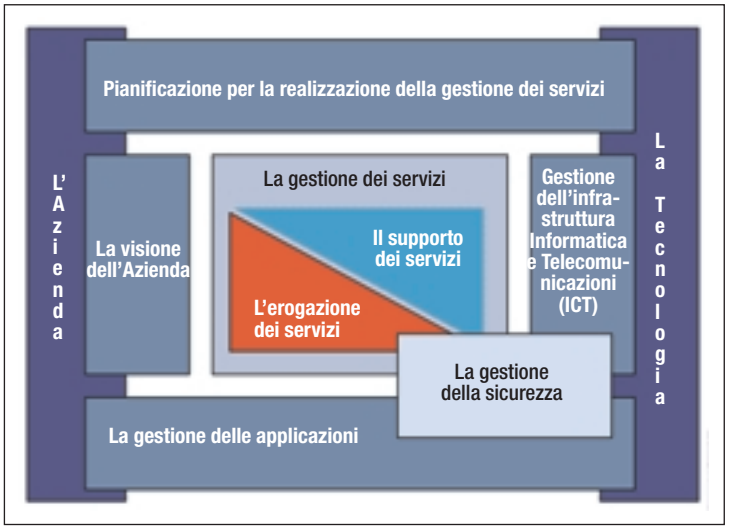


FIGURA 10
La struttura del modello ITIL, (Fonte: itSMF)

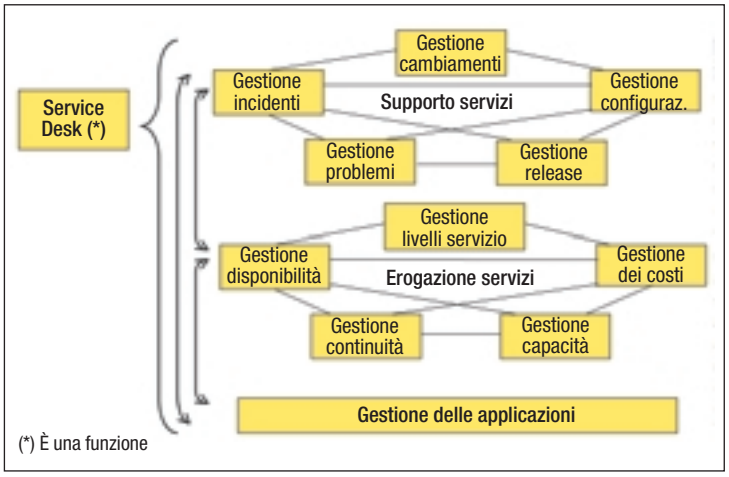


FIGURA 11
L'interazione tra i processi per la gestione ICT secondo ITIL (Fonte: itSMF)

Nella descrizione dei più significativi processi per la gestione operativa dell'ICT si farà riferimento alle logiche e allo schema del già citato ITIL, che include consolidate best practice a livello mondiale che si stanno diffondendo e consolidando anche nella realtà italiana. Come evidenziato nella figura 10, l'ITIL distingue i processi per l'erogazione del servizio ICT dai processi di supporto alla loro erogazione.
Nel seguito sono elencati i principali processi relativi alla erogazione del servizio, processi che sono tra loro strettamente correlati ed interdipendenti, così come lo sono con i processi di supporto. La figura 11 schematizza tale interdipendenza.

❑ **Gestione della Capacità (*Capacity Management*)**: il processo deve garantire la capacità di elaborazione informatica richiesta ad oggi e, in

maniera proattiva, quella prevedibile rispetto alle esigenze future, sempre in maniera conveniente e tempestiva. In tale ambito si attua il basilare monitoraggio del funzionamento delle prestazioni dei sistemi e degli applicativi, il perfezionamento delle attività per utilizzare le risorse in maniera efficiente, oltre alla sistematica preparazione del "Piano delle Capacità" (capacity planning) per poter rispondere efficacemente e prontamente alla crescita dei carichi ed alle punte di carico dei sistemi, garantendo i livelli di servizio concordati.

❑ **Gestione della Continuità (*Continuity Management*)**: questo processo deve garantire la capacità di fornire il livello di servizio concordato anche in casi di disastri, per supportare le esigenze minime dell'Ente. Assieme alla gestione della disponibilità, questo processo è strettamente correlato[27] alla gestione della sicurezza ICT. Le principali attività includono l'analisi e la gestione dei rischi ICT, il progetto e l'attuazione delle opportune contromisure, in particolare il piano e le procedure organizzative di Disaster Recovery.

❑ **Gestione della Disponibilità (*Availability Management*)**: questo processo è complementare a quello della continuità, e il suo scopo principale è la rilevazione e identificazione delle esigenze di disponibilità dei sistemi ICT da parte degli utenti. Così come per la gestione delle capacità, è basilare il monitoraggio sistematico e continuo della disponibilità dei sistemi e degli applicativi.

❑ **Gestione dei Livelli di Servizio (*SLA Management*)**: questo è il processo "chiave" per un reale orientamento ai servizi, è l'elemento che costituisce il salto dalla "tradizionale" gestione "tecnica" dei sistemi alla gestione dei servizi orientati al business o alle attività dell'Ente. Si approfondisce il tema nel sottoparagrafo 5.1.1.

❑ **Gestione Economico-finanziaria**: è l'insieme di attività per la determinazione dei Costi dei Servizi, del relativo Budget e del ricarico dei costi sugli utenti, o meglio sulle loro strutture organizzative (sussidiarie, divisioni, reparti, uffici ecc.). Inizia ora ad essere considerata anche la determinazione del valore (o almeno di una sua parte) che un determinato sistema, di norma una suite applicativa, genera per l'Ente: una stima di tale valore è effettuata nell'ambito della gestione del portafoglio applicativo e spesso viene anche condotta dalle singole unità organizzative che la utilizzano. Una importante attività, in molti casi non condotta dalle aziende/enti italiani, anche di grandi dimensioni, è l'attribuzione e la suddivisione dei costi per le diverse utenze e l'implementazione di un sistema che la supporti.

❑ **Gestione degli Incidenti (*Incident Management*)**: questo processo in primo luogo deve individuare ed identificare il tipo di incidente occorso, per poi attuare le opportune operazioni di ripristino, il tutto possibilmente automatizzato per poter riprendere la normale operatività nel minor tempo possibile e con il minimo impatto negativo. La gestione degli Incidenti deve assicurare il rispetto dei Service Level Agreement concordati, automatizzare i workflow per la segnalazione degli incidenti alla loro occorrenza e per attivare il personale e le procedure predefinite. Questo richiede una tassonomia degli incidenti, la definizione delle priorità e delle procedure di intervento ecc., oltre l'attuazione di un efficace sistema che tracci l'intero percorso dalla segnalazione dell'incidente alla sua risoluzione (tale sistema è chiamato trouble-ticketing, termine inglese spesso usato anche in italiano).

❑ **Gestione dei Problemi (*Problem Management*)**: questo processo è complementare alla gestione degli incidenti. L'incidente è l'effetto, ma quale è la sua causa? Occorre ricercare le cause dell'incidente per cercare di risolvere il problema, correggendo gli eventuali riscontri a livello infrastrutturale o a livello applicativo. È importante evidenziare come il più delle volte, per motivi di efficacia, si "tamponi" un incidente con soluzioni provvisorie o di fortuna, senza aver ben compreso le cause e averle eliminate definitivamente. La gestione dei problemi ha questo obiettivo primario ed è basilare creare un registro degli incidenti e dei problemi occorsi e conosciuti, in modo da poter disporre, nel tempo, di una banca dati della conoscenza su come comportarsi al ripetersi dello stesso incidente/problema. Anche questo processo deve essere guidato dalle SLA concordate con l'utenza.

[27] In molti casi tali attività sono incluse e gestite nell'ambito della sicurezza ICT.

❑ **Gestione delle Configurazioni (*Configuration Management*)**: la configurazione di un sistema e/o di un'applicazione assume nell'informatica odierna una valenza e una complessità ben maggior che nel passato, data anche la numerosità e la diversa profilatura degli utenti. Tale gestione si deve affiancare ed è complementare alla gestione delle versioni del software alla gestione dei cambiamenti, alla gestione delle licenze. La gestione delle configurazioni riveste inoltre un importante ruolo per la più generale gestione della sicurezza ICT. Il più delle volte la gestione della configurazione è effettuata localmente e per uno specifico hardware e software, con gli evidenti problemi in ambiti di grandi dimensioni ed eterogenei. In un'ottica di reale governo dell'ICT, tutta o gran parte della gestione delle configurazioni deve essere svolta centralmente ed automatizzata.

❑ **Gestione delle Release (*Release Management*)**: tutti i programmi software, di base ed applicativi, sono soggetti a continui aggiornamenti e a rilasci di nuove versioni, sia per i miglioramenti ed ampliamenti funzionali, sia per la correzione di errori (le così dette patch) ed il potenziamento della loro sicurezza intrinseca. Le centinaia o migliaia di programmi diversi che sono normalmente in produzione in un sistema informativo necessitano di una gestione automatizzata o semi-automatizzata per poter tempestivamente distribuire ed inserire le patch, così come gli aggiornamenti ed i cambi di versione. Per nuove versioni di un programma che portano a sostanziali cambiamenti delle funzionalità rispetto alla versione precedente, la gestione delle release utilizza i processi di controllo della gestione dei cambiamenti, delle configurazioni e delle licenze.

❑ **Gestione dei Cambiamenti (*Change Management*)**: in ITIL il processo di gestione dei cambiamenti è strettamente tecnico e non si deve confondere con il più generale, ma ben più complesso e pluri-disciplinare fenomeno di gestione di una organizzazione e delle persone che vi operano a fronte di cambiamenti (di politiche, di strumenti, di processi, di competenze ecc.) e dei relativi impatti. In questo contesto l'obiettivo è gestire i cambiamenti che vengono apportati all'infrastruttura ICT e alle applicazioni, generando il minimo impatto possibile sull'erogazione dei servizi agli utenti. La gestione dei cambiamenti è il processo di determinazione del futuro stato del sistema alla luce delle modifiche che si devono o si vogliono effettuare, della sua condizione di partenza e di tutti i passi intermedi che si dovranno compiere per raggiungere l'obiettivo.

❑ Il **Service Desk** è considerato dall'ITIL una funzione, non un processo. Il termine di "Service Desk" utilizzato da ITIL è sinonimo di termini, forse in Italia più diffusi ed usati, quali "help-desk", "contact center", "info center". L'obiettivo principale di un "service desk" è di massimizzare l'efficienza operativa nel fornire soluzioni alle domande operative degli utenti, migliorando la loro produttività. Tale funzione è spesso terziarizzata a società specializzate, ed include attività quali la gestione delle chiamate, la catalogazione ed analisi dei problemi, la soluzione dei problemi operativi, il tracciamento dello stato di avanzamento verso la soluzione del problema posto con l'eventuale coinvolgimento di altri esperti, l'interfacciamento, l'assegnazione e l'attivazione di questi ultimi, l'accumulo di informazioni sulle principali cause dei problemi e sulle modalità di risolverli, l'individuazione delle carenze hardware, software e di formazione e la gestione di almeno una parte dei livelli di servizio. Il service desk deve garantire con i clienti un'interazione multicanale (telefono, e-mail, web, sms ecc.). Le logiche e gli strumenti di un service desk possono essere inclusi in quelli della gestione dei clienti, tipicamente nei sistemi CRM. Sul mercato degli strumenti informatici di supporto e gestione al service desk esistono sia soluzioni specialistiche sia soluzioni integrate in sistemi CRM.

Per la gestione dei servizi ICT occorrono due insiemi di procedure, ossia di regole formalizzate e ben conosciute da chi dovrà seguirle:

a. *le procedure interne*, che regolano e codificano le attività interne alla struttura USI per erogare o far erogare al meglio i servizi ICT;

b. *le procedure esterne*, che regolano e codificano le interazioni tra l'USI e le varie Divisioni/Direzioni dell'Ente.

In entrambi i casi le procedure devono essere semplici ed efficaci e devono chiaramente

identificare chi fa che cosa (matrice ruoli-responsabilità). Le procedure devono essere opportunamente studiate in funzione del tipo di struttura organizzativa, delle figure professionali e delle eventuali Terze parti coinvolte. Affinché le procedure siano effettivamente seguite dai diversi interlocutori, è fondamentale che siano supportate da applicazioni in ambito intranet/extranet. L'uso di piattaforme collaborative può ulteriormente facilitare l'interazione tra i vari interlocutori, anche se normalmente utilizzate per interventi e progetti di media-lunga durata.
L'orientamento ai servizi è ormai la logica evolutiva nella gestione dei sistemi ICT e quindi più in generale del governo dell'ICT in quanto permette :

- di progettare, scegliere ed utilizzare le tecnologie ICT quali "tecnologie abilitanti", ossia di strumenti di supporto ai processi dell'Ente, facilitando l'allineamento tra il servizio offerto e le esigenze del business e dei processi che lo sottendono;
- di meglio controllare le prestazioni dei servizi informatici e di telecomunicazione;
- di facilitare l'interazione tra l'USI e le altre Direzioni/Divisioni grazie alle procedure esterne, riducendo il contenzioso tra l'USI e gli utenti tramite la definizione concordata di SLA e la loro sistematica misurazione;
- di facilitare, se e quando ritenuto opportuno, il passaggio all'insourcing/outsourcing dell'ICT.

5.1.1. I LIVELLI DI SERVIZIO

Nella logica di orientamento ai servizi, una USI o un fornitore di outsourcing stabiliscono un contratto coi loro clienti interni o esterni basato sulla definizione concordata di livelli di servizio misurabili che rappresentano le prestazioni e la qualità dei servizi da un lato erogati, dall'altro richiesti ed attesi. Tale accordo sui livelli di servizio è universalmente noto con la sigla SLA, *Service Level Agreement*.
L'erogazione dei servizi è effettuata in accordo con le SLA, che rappresentano il cuore del contratto di cui sopra e che permette di gestire i servizi al livello concordato e di assicurare, se misurate e controllate, la soddisfazione delle esigenze e dei requisiti degli utenti, soprattutto per quanto riguarda i tempi di risposta e per ciò che attiene l'effettiva usabilità dei servizi.
Le SLA contengono gli indicatori del livello di servizio erogato, chiamati KPI, *Key Performance Indicator*: indicatori che devono essere misurabili e che sono dettagliati servizio per servizio, con i valori di soglia minimo e massimo.
Per i servizi forniti da terze parti, come ad esempio quelli di accesso a reti per la trasmissione dati, l'USI deve specificare a livello di contratto con il fornitore le SLA ed i relativi KPI.
In entrambi casi è basilare che i KPI siano realmente misurabili con gli strumenti tecnici di gestione delle reti e dei sistemi ICT (*System and Network Management*) e che esista un opportuno ambiente applicativo che archivi e tracci sistematicamente nel tempo i KPI, per poter verificare, nell'arco temporale del contratto di servizio la rispondenza alle SLA concordate.
Nella definizione di SLA è necessario prevedere anche l'adozione di misure correttive che permettano di ridefinire gli obiettivi dinamicamente in corso d'opera, in modo tale da essere in grado di reagire correttamente ai cambiamenti dell'ambiente circostante. La disponibilità di piani di miglioramento (o di cambiamento) reattivo svolge un ruolo fondamentale nella gestione dei servizi più critici.
Per un medesimo servizio, possono essere definite SLA diverse in funzione di esigenze diverse, per tipologia di utenti, per la criticità delle applicazioni/sistemi (ERP, CRM, SCM, altri ambiti transazionali ecc.), per la gravità e l'urgenza del problema occorso.
Un esempio di definizione del livello di gravità da dettagliare nelle SLA è riportato nella tabella 1.
L'urgenza di intervento è un ulteriore parametro per le SLA: essa può essere indipendente dalla gravità di un problema, per esempio per il change management. Nella tabella 2, come esempio esplicativo, si sono considerati 3 livelli di urgenza, che implicano anche costi diversi, soprattutto se devono essere coinvolti enti terzi come per esempio, i fornitori.
Un importante parametro è il "rispetto dei tempi di intervento" (RTI), tipicamente usato per il *service desk* e per tutti i servizi che implichino attività di installazione, movimentazione, aggiunte e cambiamenti. Il sistema di gestione delle misure fa riferimento all'apertura di un ticket di intervento e traccia poi lo stato di avanzamento ed i vari

Gravità	Descrizione	Tempi target previsti da SLA
2	Il servizio non è disponibile.	• Per server critici: entro 4h • Per gli altri: entro 8 h
1	Il servizio non è disponibile per una parte degli utenti. Ci sono sensibili problemi prestazionali imputabili ai server. Ci sono ricorrenti problemi prestazionali imputabili ai server.	• Per server critici: entro 24 h • Per gli altri: entro 48 h
0	Il servizio è disponibile ma con un degrado funzionale e/o prestazionale tollerabile.	• Per server critici: entro 4 giorni lav. • Per gli altri: entro 7 giorni lav.

TABELLA 1
Un esempio di livelli di gravità del disservizio

Livello urgenza	Tipo urgenza	Esempi di valori di soglia per la tempestività di intervento
2	Molto urgente	Entro il giorno successivo alla chiamata nel 96-98% dei casi, entro 2 giorni nel restante 2-4%
1	Normale	Entro 4-5 giorni nel 96-98% dei casi, entro 7 h nel restante 2-4%
0	Poco urgente	Entro 7-8 giorni nel 96-98% dei casi, entro 10 giorni nel restante 2-4%

TABELLA 2
Un esempio di livelli d'urgenza dell'intervento

tempi spesi nelle diverse fasi necessarie per il suo completamento dell'intervento.
Possono essere definiti obiettivi differenziati in relazione ai diversi tipi di attività svolta e al tipo di utenza. Tipici dati elementari da rilevare includono il numero degli interventi effettuati, la data e l'ora di inizio e di chiusura dell'intervento.
L'informativa sulla soddisfazione delle SLA può essere fornita periodicamente tramite opportuni rapporti e con la disponibilità *on-demand* da parte dell'utente che abbia gli opportuni diritti di accesso. I rapporti possono avere diverse periodicità in funzione anche dei diversi servizi: giornalieri, settimanali, mensili, trimestrali o semestrali.
Nella maggior parte dei casi SLA e KPI sono ben definiti nei contratti di outsourcing, ma non lo sono nella prassi dei rapporti tra la USI e le altre strutture dell'Ente. SLA e KPI, ma più in generale qualsiasi procedura tra USI e le altre strutture organizzative dell'Ente, sono spesso considerati una burocrazia inutile e dispendiosa, ma, se gestititi con gli idonei strumenti informatici e con il livello di flessibilità e di dettaglio opportuni, sono la condizione necessaria per un effettivo governo dell'ICT all'interno dell'Ente.

6. CONCLUSIONI

Dato un sistema informativo, il suo governo può avere diversi livelli di maturità. In assenza di processi riconosciuti e formalizzati, parlando di "governo" si fa riferimento al fatto che ogni singolo sistema è controllato localmente su base individuale e caso per caso: l'intervento avviene solo all'occorrenza di malfunzionamenti e lasciato all'iniziativa del singolo operatore. Da questo livello "zero" si passa a livelli crescenti di governo, con procedure sempre più formalizzate e con l'utilizzo di strumenti informatici progressivamente più sofisticati ed integrati.
La classificazione proposta da COBIT[28] si articola nei seguenti livelli crescenti:
• *iniziale*: l'azienda è cosciente che le istanze di ICT Governance esistono e devono essere affrontate. Non ci sono, tuttavia, processi

[28] COBIT, *Control Objectives for Information and related Technology*, è un *IT Governance framework*, sviluppato da ISACA, *Information Systems Audit and Control Association & Foundation*, ed arrivato ora alla versione 4.0. Cobit dispone di una serie di tool di supporto ed è orientato all'impostazione strategica dell'EA e per questo da molti è considerato complementare al più operativo ITIL.

standardizzati ma approcci estemporanei attuati su base individuale o caso per caso;
• *ripetibile*: complessivamente sono comprese e presenti le principali istanze dell'*ICT Governance* sia a livello strategico sia a livello operativo. Le attività e gli indicatori di performance sono in una fase di sviluppo, in particolare per i processi di pianificazione, l'erogazione e il monitoraggio dei servizi;
• *definito*: è ben compresa ed accettata la necessità di agire con riferimento all'*ICT Governance*. Si è sviluppato un insieme di base di indicatori, dove i collegamenti tra le misure dei risultati e parametri delle prestazioni sono definiti, documentati e integrati nella pianificazione strategica-operativa e nei processi di monitoraggio;
• *gestito*: sussiste piena comprensione delle istanze di *ICT Governance* a tutti i livelli. Le responsabilità sono definite e monitorate mediante accordi sui livelli di servizio. Le responsabilità sono chiare e la proprietà dei processi è fissata. I processi ICT sono allineati all'azienda e alla strategia ICT;
• *ottimizzato*: c'è una comprensione avanzata e proiettata in avanti dei problemi e delle soluzioni di *ICT Governance*. I processi sono stati rifiniti al livello delle migliori pratiche e basate sui risultati del miglioramento continuo e del modello di maturità di altre organizzazioni.

In Italia, il governo dell'ICT si classifica nella parte alta di questa scala solo per le poche grandi aziende operanti anche livello internazionale e per alcune grandi Pubbliche Amministrazioni, sia locali che centrali.
Le medie e le piccole aziende e gli Enti di non grandi dimensioni sono in una fase di "comprensione" del problema e vari aspetti del governo, al di là di alcuni strettamente tecnici, non sono ancora ben percepiti e compresi dai Responsabili di alto livello. È un problema in primo luogo di cultura manageriale.
Negli USA e nell'Europa del Nord il governo dell'ICT è ad uno stadio ben più avanzato rispetto alla situazione Italiana e fornisce significativi ritorni in termini di visibilità e controllo in tempo reale su tutte le attività, strategiche e operative, sui servizi e sui progetti ICT. I processi risultano più trasparenti e aumenta l'affidabilità e la responsabilità dell'Ente nel suo complesso. È perseguito razionalmente l'allineamento continuo con gli obiettivi strategici e operativi dell'azienda e viene meglio bilanciata la domanda con l'offerta delle risorse ICT effettivamente disponibili. Il tutto porta ad un effettivo miglioramento dell'efficienza e della qualità nell'intera organizzazione, con la riduzione delle "isole" tra le funzioni e di processi, oltre che dei colli di bottiglia tra l'USI e l'Ente.
Facendo una sintesi delle varie ricerche e indagini di mercato a livello i internazionale, si possono quantificare i seguenti principali benefici:
• riduzione dei progetti ICT incompleti del 50%-70%;
• riduzione dei costi legati al personale ICT del 15-25%;
• introduzione di soluzioni ICT più rapidamente del 10-20%;
• incremento della produttività del 20-30%.

Tutti gli Enti intuiscono e sentono ormai la necessità che la USI governi l'ICT in maniera flessibile, economica e proattiva. Indipendentemente dalle diverse modalità di risposta e quindi di governo, questa percezione si traduce in un innalzamento del ruolo dell'ICT e dei suoi responsabili. Le metodologie e gli strumenti ci sono, e sempre più sofisticati, ma il vero problema non è tecnico, ma organizzativo e culturale. Occorre adeguare a queste logiche l'intera struttura organizzativa, a partire dal settore ICT stesso.

Bibliografia

[1] A cura del ClubTI di Milano: *Il valore del business per l'IT*. ISEDI Editore, 2000.

[2] Pasini, Marzotto, Perego: *La misurazione delle prestazioni dei sistemi informativi aziendali*. Egea, 2005.

[3] *EITO, European Information Technology Observatory*. Rapporto annuale sull'ICT in Europa.

[4] Il Rapporto annuale Cnel-FTI sull'ICT in Italia: *L'ICT trasforma la società. X Rapporto sulla tecnologia dell'informazione e Primo Rapporto sulla società dell'Informazione in Italia*. Franco Angeli, 2005.

[5] Bozzetti, Casotti, Pozzi (a cura di): *Crimine virtuale, minaccia reale. ICT Security: politiche e strumenti di prevenzione*. Franco Angeli, 2004.

[6] Scott W.G., ed altri: *Manuale di Management*. Il Sole 24ore Editore, 2003.

[7] Bracchi, Motta: P*rocessi aziendali e sistemi informativi*. Franco Angeli, 1997.

[8] Weill Peter, Ross Jeanne: *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. Harvard Business School Press, 2004.

[9] Wim Van Grembergen (Editor): *Strategies for Information Technology Governance*. IGP, 2003

[10] Jaap Bloem, Menno van Doorn, Piyush Mittal: *Making IT Governance Work in a Sarbanes-Oxley World. Wiley*, 2005.

[11] Kaplan Robert S., Cooper Robin: *Cost & Effect*. Harvard Business School Press, 1998.

[12] Kaplan Robert S., Norton David P.: *The Balanced Scorecard*. Harvard Business School Press.

[13] Porter Michael: *Competitive Advantage*. Macmillan 1985.

[14] Hammer Michael, Champy James: *Reengineering the Corporation*. Harper 1993.

[15] Bernus P., Mertins K., Schmidt G.: *Handbook on Architectures of Information Systems*. Springer-Verlag., 1998.

[16] Vernadat F.B.: *Enterprise Modelling and Integration: principles and applications*. Chapman & Hall, 1996.

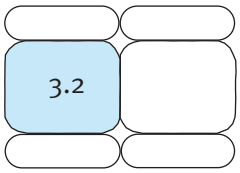
[17] Spewak Steven H.: *Enterprise Architecture Planning : Developing a Blueprint for Data, Applications, and Technology*. Wiley, 2003.

MARCO BOZZETTI si occupa di ICT da più di 35 anni e ha operato con responsabilità crescenti presso primarie imprese di produzione, quali Olivetti e Italtel, e di consulenza, quali Arthur Andersen; è stato anche responsabile dei sistemi informativi dell'intero Gruppo ENI.
Attualmente opera come consulente in GeaLab ed in Gea Consulenti Associati. E' stato uno dei primi ad occuparsi di internetworking e di sicurezza ICT e fu ideatore di EITO, European IT Observatory, e di OCI, Osservatorio Criminalità ICT in Italia, pubblicato periodicamente dall'FTI, Forum delle tecnologie dell'Informazione. È Vice e Past President di FIDAInform, socio fondatore e componente del Comitato Scientifico dell'FTI, componente del Consiglio del Terziario Innovativo di Assolombarda, Past President e membro del Consiglio Direttivo del ClubTI di Milano. Ha pubblicato articoli e libri sull'evoluzione tecnologica, la sicurezza, gli scenari e gli impatti dell'ICT.
E-mail: marco.bozzetti@gealab.it

REVERSE ENGINEERING DI SISTEMI SOFTWARE
LIMITI E POTENZIALITÀ

Filippo Ricca
Paolo Tonella

Il reverse engineering è un processo di analisi mirato all'identificazione delle componenti e relazioni di un sistema software. Da alcuni esperti è considerato un potente mezzo, mentre da altri un processo non realizzabile in pratica. Chi ha ragione? In questo articolo vengono sottolineati i limiti e le potenzialità del reverse engineering. Prima di concludere con due storie di successo a livello industriale, l'articolo propone anche una possibile classificazione dei tool usati nel processo di reverse engineering.



1. INTRODUZIONE

Col passare degli anni, modifica dopo modifica, i sistemi software invecchiano e la loro struttura tende inevitabilmente a degenerare. Spesso i manager e gli ingegneri del software sono costretti a dover gestire un sistema molto grande, vitale per l'organizzazione, costato molto e al tempo stesso difficile da comprendere e mantenere. A peggiorare le cose è il fatto che solitamente la documentazione non esiste e se esiste non è aggiornata. Questo punto di arrivo, che ha come conseguenza quella di rendere molto costosi e difficili tutti gli interventi di manutenzione, costringe spesso i manager a una scelta difficile: riscrivere il sistema da zero oppure ristrutturarlo completamente.
Le tecniche di *reverse engineering* possono essere utili nel caso in cui viene scelta questa seconda soluzione. Il *reverse engineering* [2], in italiano ingegneria inversa, è un processo di analisi di un oggetto o dispositivo (un componente elettronico, un aeroplano, un motore, un software ecc.) che ha come obbiettivo unicamente la comprensione. Solitamente il passo successivo del *reverse engineering* è quello di costruire un nuovo oggetto/dispositivo avente caratteristiche e funzionalità simili a quello analizzato.
Il *reverse engineering* è stato usato spesso dalle forze armate al fine di copiare la tecnologia bellica dei nemici [7]. In ambito militare, celebri esempi di *reverse engineering* sono il missile *Scud Mod A*, il bombardiere russo *Tupolev Tu-4* e il personal computer *Agatha*. La Corea del nord in tempi recenti si è dotata di missili a lunga gittata denominati Scud Mod A. Questi missili sono stati costruiti seguendo un processo di ingegneria inversa a partire dai missili sovietici Scud BS. Durante la seconda guerra mondiali alcuni bombardieri americani B-29 diretti in Giappone furono costretti ad atterrare in Unione Sovietica per problemi tecnici. In quel tempo i Sovietici non erano dotati di aerei così avanzati e sfruttando questo caso fortuito decisero di copiare la tecnologia dei B-29; nacquero così i Tupolev Tu-4. Il personal computer Agatha, vera punta di diamante dei sovietici per diversi anni, è stato creato a partire dal computer americano Apple II.

In ambito informatico lo scopo del *reverse engineering* è abbastanza diverso rispetto a quello militare. Solitamente l'ingegneria inversa è utilizzata dai progettisti per aumentare il grado di conoscenza di un sistema software quando questo deve essere sottoposto ad una operazione di modifica. Nei casi reali non è raro che i progettisti siano costretti ad eseguire operazioni di manutenzione del codice senza avere una conoscenza approfondita dello stesso. Questa eventualità si ha tutte le volte che sul software agiscono progettisti o ingegneri che non hanno partecipato alle fasi di sviluppo – caso assai frequente visto che alcuni sistemi hanno oramai più di 30 anni – oppure quando occorre modificare il proprio codice dopo che è passato un po' di tempo.
Le tecniche e i *tool* di *reverse engineering* forniscono i mezzi per generare un'adeguata documentazione del codice; in particolare sono in grado di produrre documenti mai esistiti o recuperare quelli che si sono persi con il passare degli anni. Il risultato di questo processo è un insieme di rappresentazioni alternative del sistema, spesso grafiche, che aiutano il progettista nella fase di manutenzione ed evoluzione del codice.

2. MANUTENZIONE DEL SOFTWARE

Il ciclo di vita di un sistema software inizia con l'analisi del dominio e dei requisiti, continua con la fase di sviluppo (progettazione e codifica) e termina con la dismissione – momento in cui viene stabilito che il sistema software non è più utile per l'organizzazione. Dopo il rilascio, cioè quando il sistema diventa fruibile da parte degli utenti, il software entra in una fase molto delicata che solitamente dura molto, la manutenzione.
Per manutenzione si intende il complesso di operazioni necessarie a:
- conservare il sistema software funzionante ed efficiente con il passare degli anni;
- mantenere le funzionalità del sistema continuamente aggiornate.

Dopo avere condotto diversi studi empirici sull'evoluzione/manutenzione del software, Lehman and Belady [17] hanno concluso che ci sono delle "leggi" che valgono per tutti i sistemi software.

Leggi sull'evoluzione del software
- Un programma che è utilizzato in un ambiente reale deve necessariamente essere modificato o diviene progressivamente meno utile in quell'ambiente.
- Man mano che un software evolve, la sua struttura diviene sempre più complessa. Risorse extra devono essere impiegate per preservare e semplificare la struttura.
- Le funzionalità offerte dal sistema devono essere continuamente aggiornate ed estese per mantenere gli utenti soddisfatti.
- La qualità di un sistema tende a peggiorare modifica dopo modifica. Risorse extra devono essere impiegate al fine di contenere questo degrado irreversibile della struttura.

Nell'riquadro sono riportati i loro risultati più significativi.
La manutenzione è una fase molto critica e ancora oggi, nonostante il miglioramento delle tecniche di sviluppo e dei processi di produzione, rappresenta per tutte le aziende una spesa molto cospicua. Sommerville [23] la stima in un valore compreso tra il 50% e il 75% della spesa totale di produzione del software. Mantenere ed evolvere un sistema software esistente è difficile perché bisogna tenere presenti diversi aspetti: il veloce "turnover" degli sviluppatori, la continua crescita dei sistemi software in termini di grandezza e complessità e l'evoluzione della tecnologia.
I costi sono molto alti perché la manutenzione è composta di una serie di attività estremamente complesse:
- ***comprensione del codice***: consiste in un'analisi approfondita del software (o solo di una parte). Prima di effettuare una modifica occorre, infatti, capire come funziona il codice e dove apportarla. Questa è sicuramente l'attività più complessa tra tutte; è estremamente complicato comprendere il proprio codice e peggio ancora capire il software scritto da altri;
- ***analisi di impatto***: serve a capire quali funzionalità vanno aggiornate in modo da renderle consistenti con la modifica da implementare;
- ***modifica del codice***: consiste nella fase di codifica del cambiamento;
- ***validazione***: consiste nel verificare che la modifica non abbia introdotto nuovi errori.

La manutenzione, oltre ai costi diretti, implica anche un insieme di costi indiretti. Il più evidente in ambito industriale è lo sbilanciamento nell'allocazione delle risorse verso la fase di manutenzione. Quello che accade in realtà è che i programmatori sono impegnati, quasi sempre, a mantenere codice già rilasciato dedicando poco tempo allo sviluppo di nuovi progetti. Il risultato è che le risorse effettive impegnate per lo sviluppo di nuovi progetti sono spesso insufficienti.

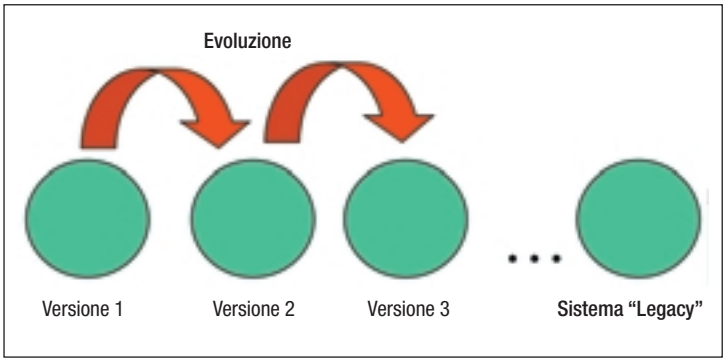


FIGURA 1
Le modifiche ripetute degradano la struttura interna del software. Spesso il risultato è un sistema legacy

Un altro aspetto che rende questa fase molto delicata è il fatto che una manutenzione non strutturata e caotica porta al rapido degrado del sistema, facendo diventare le successive modifiche sempre più complesse. Un sistema software raggiunge lo stato di *legacy* (Figura 1) quando subisce nel tempo diverse modifiche e la fase di manutenzione diventa sempre più complessa e costosa.

3. SISTEMI "LEGACY"

Un sistema *legacy* [1] è un software che è stato mantenuto in vita per molti anni e che ha le seguenti caratteristiche:

- ❑ è vitale per un'organizzazione ed è pesantemente utilizzato. Per esempio deve essere operativo 24 h su 24 e 7 giorni su 7;
- ❑ su di esso l'organizzazione ha investito molto, sia in termini economici che di tempo;
- ❑ non può essere dismesso facilmente in quanto è l'unico *repository* di conoscenza. Le procedure dell'azienda (le cosiddette *business rules*) non sono registrate in nessun altro posto e sono contenute solo nel codice;
- ❑ solitamente è un codice enorme costituito da milioni di linee;
- ❑ è scritto in un linguaggio di vecchia generazione (per esempio: Cobol, Assembler, PL/1, RPG ecc.) ed è progettato secondo vecchie concezioni (per esempio: programmazione non strutturata, salti incondizionati ecc.). Inoltre spesso è eseguito su piattaforme obsolete;
- ❑ solitamente utilizza un database obsoleto sempre che non faccia uso di un *file system* (file ad accesso sequenziale, accesso diretto ecc.);
- ❑ solitamente è un'applicazione monolitica. Presentazione, logica applicativa ed accesso ai dati sono fusi assieme in un unico blocco (le componenti non sono separate né logicamente né fisicamente);
- ❑ è difficile da comprendere, mantenere ed espandere in quanto solitamente la documentazione non esiste e se esiste non è aggiornata con le modifiche che sono state, nel corso degli anni, apportate al software (codice e documentazione non sono allineati).

Molti software che furono sviluppati negli anni '70 e '80, in linguaggi come Cobol, Assembler, PL/1, RPG e che ancora oggi sono utilizzati in diversi ambiti, possono essere considerati a pieno titolo dei sistemi *legacy*. Di fatto i *legacy* sono ancora molto comuni [18] e sono tuttora ampiamente utilizzati. Questo accade soprattutto nella Pubblica amministrazione, in quanto la loro dismissione non può avvenire facilmente perché non giustificabile economicamente. Sebbene l'espressione sistema *legacy* è spesso associata a sistemi scritti in linguaggi di programmazione obsoleti non è raro trovare sistemi recenti che presentano problemi simili [3]. È importante sottolineare che le tecniche di *reverse engineering*, originariamente pensate per i sistemi *legacy*, si sono rivelate estremamente utili anche in supporto alla comprensione di sistemi software scritti secondo approcci moderni (per esempio, codice orientato agli oggetti o applicazioni Web).

4. APPROCCI POSSIBILI PER TRATTARE I SISTEMI LEGACY

Il momento più critico per un'organizzazione si ha quando il sistema software che è stato utilizzato con successo per anni è diventato, con il passare del tempo, inefficiente o troppo costoso da mantenere. Una soluzione è quella di rimpiazzare il vecchio sistema con uno moderno più semplice da mantenere e più efficiente. Questa soluzione è raramente scelta dai manager perché presenta elevati rischi per l'organizzazione e al tempo stesso spesso ne raddoppia i costi (l'organizzazione dovrà investire nel nuovo sistema e contemporaneamente mantenere funzionante quello vecchio). Costruire da zero un nuovo sistema che riproduce ed estende le funzionalità di uno vecchio è

un'operazione complessa, soggetta a molti errori e che solitamente prende molto più tempo di quello preventivato. Molti progetti statunitensi di questo tipo iniziati nel biennio 1992/93 si rivelarono un autentico disastro; nel 1995 quasi l'80% erano falliti, nel senso che erano stati ridimensionati, posposti o addirittura cancellati [18]. Altra complicazione è il fatto che non è possibile avere la certezza che il nuovo sistema, una volta sviluppato, riprodurrà fedelmente parte delle funzionalità di quello vecchio.
Il ***legacy dilemma*** consiste proprio in questo: *è più conveniente ristrutturare completamente il vecchio sistema e continuare ad usarlo oppure accantonarlo e riscriverne uno nuovo che riproduce ed estende le funzionalità del vecchio*?
Di fatto tra i due estremi, detti sostituzione a taglio netto (o riscrittura) e migrazione totale (o ristrutturazione totale), esistono anche soluzioni intermedie. In generale, esistono diversi approcci possibili per trattare i sistemi *legacy* [18]:

- ***manutenzione del sistema legacy nella sua forma originale***: spesso questa soluzione non è praticabile in quanto troppo costosa;
- ***sostituzione a taglio netto o riscrittura***: consiste nell'accantonare il *legacy* e procedere con una riscrittura da zero del nuovo sistema. Questa è sicuramente la soluzione che presenta più rischi;
- ***wrapping***: si tratta di aggiungere al *legacy system* un nuovo livello software che nasconde l'implementazione effettiva delle funzionalità e presenta il vecchio codice sotto una nuova forma. Il *wrapping* può essere realizzato a livello di interfaccia utente o a livello di sistema. Nel primo caso la vecchia applicazione appare all'utente completamente trasformata, anche se di fatto non lo è, e con un interfaccia grafica moderna. Nel secondo caso il nuovo livello software presenta il sistema *legacy* attraverso un'interfaccia software ben definita, tipicamente ad oggetti, che permette la connessione del vecchio software con i nuovi moduli che costituiscono l'ambiente distribuito (Figura 2). In questo modo la vecchia applicazione appare sotto forma di uno o più oggetti, del tutto simili a tutti gli altri, ed in grado di operare nello stesso ambiente, accettare messaggi dagli altri moduli e rispondere alle richieste [18];
- ***migrazione parziale***: solo una parte del sistema, quella che crea più problemi o quella più obsoleta, è trasformata. Solitamente è mantenuta in vita la parte del *legacy* che implementa le *business rules*, mentre sono migrate le interfacce utente e i dati. La differenza tra migrazione parziale e *wrapping* è nella prospettiva di evoluzione del sistema. Nel primo caso sono trasformate alcune componenti software per facilitare gli interventi futuri di manutenzione sul sistema. Nel secondo caso l'implementazione effettiva delle funzionalità è nascosta da un nuovo strato software nella speranza di non dover più intervenire sul sistema *legacy*;
- ***migrazione totale***: la migrazione totale consiste nel trasformare completamente il sistema *legacy* in uno nuovo con l'intenzione di migliorare la manutenibilità. Questo processo può presentare grossi rischi e spesso richiede molti anni; pertanto la migrazione totale deve avvenire gradualmente e incrementalmente, in modo da mantenere sempre operativo il sistema.

Tipicamente le paure e le incertezze legate agli approcci più estremi fanno preferire le soluzioni intermedie. *Wrapping* e migrazione parziale raramente danno luogo alla soluzione migliore, ma hanno il pregio di non esporre l'organizzazione a grossi rischi e soprattutto di costare meno.

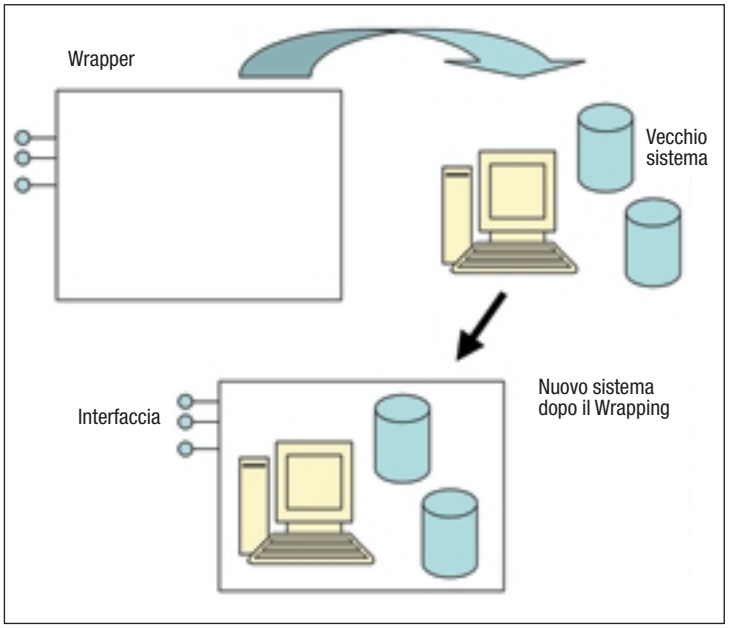


FIGURA 2
Operazione di wrapping a livello di sistema [18]

Per questo motivo le soluzioni intermedie sono scelte più frequentemente.

5. REVERSE ENGINEERING

Per molti anni l'ingegneria del software si è concentrata soprattutto sullo sviluppo di nuove applicazioni trascurando in parte la fase di manutenzione. Ultimamente però, con il crescere dell'importanza del software e soprattutto con la proliferazione dei sistemi *legacy*, questa fase è stata notevolmente rivalutata sia nel mondo della ricerca che in quello dell'industria.
Come già accennato, uno dei momenti più critici della manutenzione è rappresentato dalla comprensione del codice, fase nella quale il programmatore cerca di capire la struttura interna del software (o parte del software) e il suo funzionamento prima di effettuare le dovute modifiche.
È proprio nella fase di comprensione che risultano particolarmente utili le tecniche di *reverse engineering*. Queste tecniche forniscono i mezzi per recuperare le informazioni perse con il passare degli anni o che non sono mai esistite. L'obbiettivo è quello di costruire progressivamente dei modelli mentali del sistema in modo tale da migliorare la comprensione dello stesso. Mentre questa operazione non è complessa per piccoli sistemi software, dove lettura ed ispezioni del codice sono spesso sufficienti, diventa estremamente problematica nel caso di sistemi *legacy* a causa della loro grandezza e complessità.
Dato un sistema legacy ci sono diversi approcci per cercare di ricostruire le funzionalità del software e la loro interazione con i dati:

- ***lettura della documentazione esistente e del codice***: è difficile usare questo approccio quando la documentazione è datata, incompleta o inesistente. Inoltre la lettura del codice diventa improponibile quando le linee di codice sono nell'ordine del milione (si dice che questa tecnica non scala con il crescere del sistema);
- ***intervista agli utenti e agli sviluppatori***: questo è un buon metodo ma è applicabile solo raramente. È infatti difficile trovare gli sviluppatori originali che hanno partecipato allo sviluppo del codice;
- ***utilizzo di tools*** e delle tecniche proprie del *reverse engineering* per generare rappresentazioni ad alto livello del codice: vedremo nel paragrafo 7 una lista di *tools* di *reverse engineering*.

In letteratura esistono due definizioni diverse di *reverse engineering*, una "forte" e una cosiddetta "debole" [22]:

- ***definizione forte*** di *reverse engineering*: è un processo che a partire dal codice *legacy* permette di estrarre le specifiche formali del sistema. Le informazioni di design del sistema vengono derivate dal codice come passo intermedio (Figura 3) e le specifiche formali estratte possono essere usate per creare una nuova implementazione del sistema. In questa definizione ci sono tre assunzioni implicite:
  - il processo è automatico;
  - le specifiche sono ad un buon livello di astrazione in modo tale che il sistema possa essere re-implementato in un altro linguaggio o secondo concezioni diverse;
  - il tempo e lo sforzo per derivare le specifiche è inferiore rispetto a quello richiesto per costruire il sistema da zero;
- ***definizione debole*** di *reverse engineering*: è un processo automatico o semi-automatico (ovvero assistito dall'utente) che a partire dal codice *legacy* permette di derivare una base di conoscenza del sistema. La base di conoscenza è costituita da rappresentazioni alternative del codice *legacy*, spesso ad un livello più astratto e grafico, che mettono in risalto alcune proprietà e caratteristiche del sistema stesso.

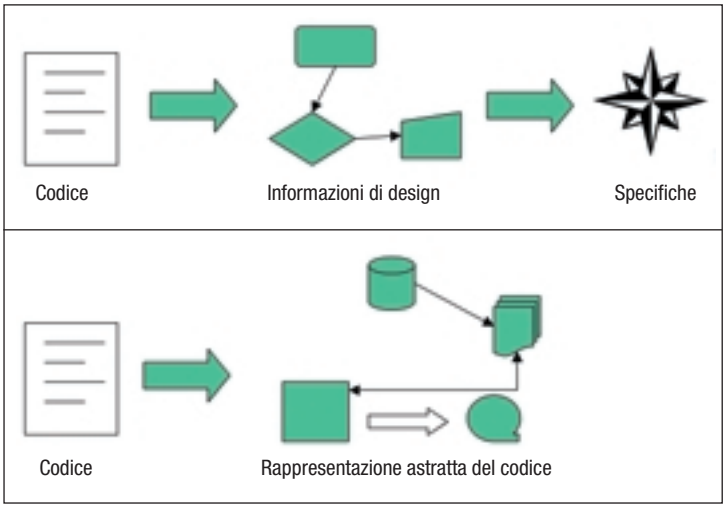


FIGURA 3
Definizione forte (in alto) e debole (in basso) di reverse engineering

Questa seconda definizione è molto meno ambiziosa rispetto alla precedente. Nella definizione debole non viene richiesto come risultato un insieme di specifiche formali del sistema ma solo una rappresentazione astratta del codice. Altra differenza fondamentale è che il processo non deve essere completamente automatico bensì assistito dall'utente.

6. SUCCESSO O FALLIMENTO?

Rispondere alla domanda se il *reverse engineering* sia stato e sia tuttora un successo è estremamente difficile. La risposta chiaramente dipende dal risultato che ci aspettiamo dal *reverse engineering* e quindi dalla definizione che scegliamo (forte o debole).
Se scegliamo la definizione forte allora dovremmo concludere che il *reverse engineering* è stato ed è un fallimento. Lo stato dell'arte è molto lontano dagli obbiettivi prefissati da questa definizione. I *tool* commerciali ed accademici non sono in grado di recuperare automaticamente dal codice le specifiche formali. Inoltre anche limitandosi alle informazioni di design non esistono *tool* in grado di recuperare dal codice delle rappresentazioni astratte usabili in modo completamente automatico. Le rappresentazioni astratte che si ottengono dai *tool* di *reverse engineering* devono poi, per essere utilizzabili, venire raffinate manualmente. Inoltre i *tool* devono essere "guidati" nel processo di recupero delle informazioni: l'intervento umano è indispensabile per avere dei risultati accettabili.
Per quanto concerne la terza assunzione della definizione forte, quella relativa ai tempi di sviluppo, risulta molto difficile verificarla e dipende da sistema a sistema. La definizione forte presenta inoltre un altro grosso problema, relativo alla seguente domanda: *è possibile estrarre specifiche formali dal codice in modo completamente automatico*? In generale non è infatti decidibile l'equivalenza tra due rappresentazioni formali alternative di un sistema, quali codice e specifiche.
Se invece viene scelta la definizione debole allora la risposta alla domanda iniziale è completamente diversa: il *reverse engineering* è stato ed è un successo. Esistono, infatti, in commercio e nel mondo accademico innumerevoli *tool* che sono in grado di recuperare in modo semi-automatico dal codice, rappresentazioni astratte (grafiche e testuali) molto utili nella fase di comprensione del software.
Al momento le ricerche si stanno concentrando principalmente su tecniche e *tool* di estrazione di informazioni di design a diversi livelli di astrazione.

7. TOOL DI REVERSE ENGINEERING: COMMERCIALI ED ACCADEMICI

In commercio e nel mondo accademico esistono diversi programmi che sono classificati come *tool* di *reverse engineering*. Questa sezione propone una possibile classificazione, necessariamente non esaustiva, di questi tools. Una lista più completa può essere trovata su Internet all'indirizzo [8].
Tra i *tool* più utilizzati per scopi di comprensione del codice troviamo:
1. ***pretty printer***: **f**ormattano il codice in una forma più leggibile. Questi programmi si rivelano particolarmente utili tutte le volte che il codice è stato scritto con convenzioni di formattazione obsolete o particolarmente difficili da capire. DMS [10], un *tool* commerciale di trasformazione automatica del codice che dispone tra le varie funzionalità anche quella di ri-formattare il codice, è stato utilizzato per analizzare il codice C vincitore di un edizione del IOCCC, una competizione internazionale che premia il codice C più "offuscato" (Figura 4);
2. ***visualizzatori di codice*** (*code viewer*): producono viste alternative del software. Alcune sono testuali altre grafiche. A questa categoria appartengono i generatori di diagrammi di flusso (per esempio, il *tool* commerciale *Code Visual to Flowchart* V3.5 [11]) e i navigatori di codice (per esempio, il *tool Source-Navigator* [12]), che permettono di "navigare" con facilità all'interno del codice grazie all'ausilio di particolari viste ad alto livello (per esempio, la vista gerarchica delle classi nei sistemi ad oggetti). Un *tool freeware* che rientra in questa categoria è il *CodeCrawler* (riquadro a p. 59) [16]. Questo *tool* implementa e visualizza il concetto di viste polimetriche, ovvero viste del software arricchite con informazioni semantiche (me-

```
#include                                        <math.h>
#include                                      <sys/time.h>
#include                                      <X11/Xlib.h>
#include                                     <X11/keysym.h>
                                              double L,o,P
                                             ,_=dt,T,Z,D=1,d,
                                             s[999],E,h=8,I,
                                             J,K,w[999],M,m,O
                                             ,n[999],j=33e-3,i=
                                             1E3,r,t,u,v,W,S=
                                             74.5,l=221,X=7.26,
                                             a,B,A=32.2,c,F,H;
                                              int N,q,C,y,p,U;
                                             Window z;char f[52]
                                             ;GC k;main(){Display*e=
XOpenDisplay(0);z=RootWindow(e,0);for(XSetForeground(e,k=XCreateGC(e,z,0,0),BlackPixel(e,0))
;scanf("%lf%lf%lf",y+n,w+y,y+s)+1;y++);XSelectInput(e,z=XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0)),KeyPressMask);for(XMapWindow(e,z);;T=sin(O)){struct timeval G={0,dt*1e6}
;K=cos(j);N=1e4;M+=H*_;Z=D*K;F+=_*P;r=E*K;W=cos(O);m=K*W;H=K*T;O+=D*_*F/K+d/K*E*_;B=
sin(j);a=B*T*D-E*W;XClearWindow(e,z);t=T*E+D*B*W;j+=d*_*D-_*F*E;P=W*E*B-T*D;for(o+=(l=D*W+E
*T*B,E*d/K*B+v+B/K*F*D)*_;p<y;){T=p[s]+i;E=c-p[w];D=n[p]-L;K=D*m-B*T-H*E;if(p[n]+w[p]+p[s
]==0|K<fabs(W=T*r-l*E+D*P)|fabs(D=t*D+Z*T-a*E)>K)N=1e4;else{q=W/K*4E2+2e2;C=2E2+4e2/K
*D;N-1E4&&XDrawLine(e,z,k,N,U,q,C);N=q;U=C;}++p;}L+=_*(X*t+P*M+m*l);T=X*X+l*l+M*M;
XDrawString(e,z,k,20,380,f,17);D=v/l*15;i+=(B*l-M*r-X*Z)*_;for(;XPending(e);u*=CS!=N){
XEvent z;XNextEvent(e,&z);
++*((N=XLookupKeysym
(&z.xkey,0))-IT?
N-LT?UP-N?&E:&
J:&u:&h):--*(
DN-N?N-DT?N==
RT?&u:&W:&h:&J
);}m=15*F/l;
c+=(l=M/l,l*H
+l*M+a*X)*_;H
=A*r+v*X-F*l+(
E=.1+X*4.9/l,t
=T*m/32-l*T/24
)/S;K=F*M+(
h*1e4/l-(T+
E*5*T*E)/3e2
)/S-X*d-B*A;
a=2.63/l*d;
X+=(d*l-T/S
*(.19*E+a
*.64+J/1e3
)-M*v+A*
Z)*_;l+=
K*_;W=d;
sprintf(f,
"%5d %3d"
"%7d",p=l
/1.7,(C=9E3+
O*57.3)%0550,(int)i);d+=T*(.45-14/l*
X-a*130-J*.14)*_/125e2+F*_*v;P=(T*(47
*l-m*52+E*94*D-t*.38+u*.21*E)/1e2+W*
179*v)/2312;select(p=0,0,0,0,&G);v-=(
W*F-T*(.63*m-l*.086+m*E*19-D*25-.11*u
)/107e2)*_;D=cos(o);E=sin(o);}}
```

```
#include <sys/time.h>
#include <X11/Xlib.h>
#include <X11/keysym.h>
double L, o, P, _ = dt, T, Z, D = 1, d, s[999], E, h = 8, I, J, K, w[999],
       M, m, O, n[999], j = 3.3e-2, i = 1e3, r, t, u, v, W, S = 7.45e1,
       l = 221, X = 7.26, a, B, A = 3.22e1, c, F, H;
int N, q, C, y, p, U;
Window z;
char f[52];
GC k;
main()
{
  Display * e = XOpenDisplay(0);
  z = RootWindow(e, 0);
  for (XSetForeground(e, k = XCreateGC(e, z, 0, 0), BlackPixel(e, 0));
  scanf("%lf%lf%lf", y + n, w + y, y + s) + 1; y++);
  XSelectInput(e, z = XCreateSimpleWindow(e, z, 0, 0, 400, 400,
               0, 0, WhitePixel(e, 0)), KeyPressMask);
  for (XMapWindow(e, z);; T = sin(O))
    {
      struct timeval G = { 0, dt * 1e6 };
      K = cos(j);
      N = 1e4;
      M += H * _;
      Z = D * K;
      F += _ * P;
      r = E * K;
      W = cos(O);
      m = K * W;
      H = K * T;
      O += D * _ * F / K + d / K * E * _;
      B = sin(j);
      a = B * T * D - E * W;
      XClearWindow(e, z);
      t = T * E + D * B * W;
      j += d * _ * D - _ * F * E;
      P = W * E * B - T * D;
      for (o += (l = D * W + E * T * B, E * d / K * B + v + B / K * F * D) * _; p < y;)
        {
          T = p[s] + i;
          E = c - p[w];
          D = n[p] - L;
          K = D * m - B * T - H * E;
          if (p[n] + w[p] + p[s] == 0 | K < fabs(W = T * r - l * E + D * P) | fabs(D = t * D + Z * T - a * E) > K)
            N = 1e4;
          else
            {
              q = W / K * 4e2 + 2e2;
              C = 2e2 + 4e2 / K * D;
              N - 1e4 && XDrawLine(e, z, k, N, U, q, C);
              N = q;
              U = C;
            }
          ++p;
        }
      L += _ * (X * t + P * M + m * l);
      T = X * X + l * l + M * M;
      XDrawString(e, z, k, 20, 380, f, 17);
      D = v / l * 15;
      i += (B * l - M * r - X * Z) * _;
      for (; XPending(e); u *= CS != N)
        {
          XEvent z;
          XNextEvent(e, & z);
          ++ * ((N = XLookupKeysym(& z.xkey, 0)) - IT ? N - LT ? UP - N ? & E : & J : & u : & h);
          -- * (DN - N ? N - DT ? N == RT ? & u : & W : & h : & J);
        }
      m = 15 * F / l;
      c += (l = M / l, l * H + l * M + a * X) * _;
      H = A * r + v * X - F * l + (E = 1e-1 + X * 4.9 / l, t = T * m / 32 - l * T / 24) / S;
      K = F * M + (h * 1e4 / l - (T + E * 5 * T * E) / 3e2) / S - X * d - B * A;
      a = 2.63 / l * d;
      X += (d * l - T / S * (1.9e-1 * E + a * 6.4e-1 + J / 1e3) - M * v + A * Z) * _;
      l += K * _;
      W = d;
      sprintf(f, "%5d %3d"
              "%7d", p = l / 1.7, (C = 9e3 + O * 5.73e1) % 0550, (int) i);
      d += T * (4.5e-1 - 14 / l * X - a * 130 - J * 1.4e-1) * _ / 1.25e4 + F * _ * v;
      P = (T * (47 * l - m * 52 + E * 94 * D - t * 3.8e-1 + u * 2.1e-1 * E) / 1e2 + W * 179 * v) / 2312;
      select(p = 0, 0, 0, 0, & G);
      v -= (W * F - T * (6.3e-1 * m - l * 8.6e-2 + m * E * 19 - D * 25 - 1.1e-1 * u) / 1.07e4) * _;
      D = cos(o);
      E = sin(o);
    }
}
```

FIGURA 4

Codice C vincitore di un'edizione del IOCCC prima e dopo il pretty printing (solo una porzione)

Due storie di successo in ambito industriale

Tra i tanti *tool* di *reverse engineering* quelli che forse hanno avuto un maggior impatto a livello industriale sono stati *Codecrawler* [16] e *Rigi* [9].

Il *Codecrawler* è un *tool open-source* di visualizzazione del codice che permette diversi tipi di viste grafiche. Questo tool, scritto completamente in Smalltalk, è stato usato con successo in diversi progetti industriali di *reverse engineering*. La tabella seguente (presa da [3]) fornisce una lista di sistemi (industriali e non) che sono stati analizzati usando questo *tool*. Per questioni di riservatezza industriale gli autori non hanno potuto fornire descrizioni dettagliate dei sistemi analizzati (per questo motivo nella tabella i sistemi industriali sono indicati in modo generico con S1, S2, S3 ecc.).

UNA SELEZIONE DI SISTEMI ANALIZZATI CON CODECRAWLER

Sistema	Linguaggio	Linee di codice	Classi
S1 (Network Switch)	C++	1.200.000	2300
S2 (Network Switch)	C++/Java	140.000	400
S3 (Multimedia)	Smalltalk	600.000	2500
S4 (Payroll)	Cobol	40.000	-
Squeak (Multimedia Env.)	Smalltalk	300.000	1800
JBoss	Java	300.000	4900
S5 (Logistics)	C++	120.000	300

Come si può vedere dalla tabella il *tool* è stato utilizzato per analizzare sistemi scritti in linguaggi differenti (ad oggetti ed imperativi) e di dimensioni molto diverse tra loro. Gli autori sostengono che il *tool* non ha problemi di scalabilità e che il tempo medio impiegato per l'analisi dei sistemi inseriti nella tabella è stato per ognuno meno di una settimana di lavoro. Il risultato tipico prodotto per ogni sistema analizzato è stato un *report* contenente le viste polimetriche più significative e una lista di possibili problemi individuati (classi/moduli troppo estesi, metodi troppo lunghi, attributi o metodi non usati, *dead code* ecc.). Per i sistemi di dimensione media sono stati proposti anche interventi di *restructuring* e *re-engineering*. In un sistema, è stato proposto, per ridurre la complessità eccessiva della gerarchia della classi, l'utilizzo del design pattern "*template method*" [4]. I progettisti di questi sistemi, spesso inizialmente scettici, hanno giudicato l'informazione estratta dal *CodeCrawler* come estremamente rilevante [3] utilizzandola, in certi casi, per scopi di documentazione o come base per futuri lavori di ristrutturazione. La conclusione finale dei progettisti del *CodeCrawler* [3] è che le viste polimetriche non sostituiscono completamente l'ispezione e la lettura del codice ma aiutano nella fase di comprensione del codice.

L'altro *tool* che vogliamo menzionare è *Rigi* [9], un *tool* altamente personalizzabile che permette di ricavare alcune viste e *report* (decomposizione del sistema basata sui tipi, struttura a file, architettura del sistema, analisi delle librerie condivise ecc.) dal codice in modo interattivo o non interattivo. Gli autori sostengono che per sistemi molto grandi è necessario lasciare all'utente la possibilità di manipolare le viste in modo non-interattivo, ovvero facendo uso di *script*. *Rigi* supporta un linguaggio di *scripting* basato su Tcl [21] ed un esteso insieme di comandi con i quali è possibile produrre astrazioni personalizzate del sistema e *report*. Il *tool* può analizzare diversi linguaggi di programmazione. Anche Rigi è stato applicato con successo a diversi sistemi commerciali. Uno tra questi è presentato in [25], dove un sistema industriale di buone dimensioni (700.000 linee di codice) scritto in C e C++ è stato completamente analizzato con *Rigi* con l'intenzione di ottenere alcune informazioni sulla sua struttura (viste grafiche e *report* testuali) e potenziali anomalie. Anche in questo caso gli autori, per questioni di riservatezza industriale, non hanno potuto rivelare il dominio dell'applicazione e il nome del sistema. Durante la fase di *reverse engineering* gli autori hanno sfruttato la possibilità offerta da *Rigi* di personalizzare le varie analisi mediante la scrittura di *script*. La prima decomposizione ottenuta è stata quella a livello architetturale. Lo scopo di questa vista è quello di suddividere il sistema in sottosistemi, capire le funzionalità offerte da ogni modulo e soprattutto determinare il livello di accoppiamento tra le varie sottoparti. Da questa analisi gli autori hanno ricavato alcune informazioni importanti: il sistema è composto di sette sottosistemi principali molto connessi (cioè l'accoppiamento è alto) di cui cinque contenenti un interfaccia grafica (GUI). Gli sviluppatori originali hanno giudicato questa vista molto utile trovando nel loro sistema relazioni inaspettate tra i vari moduli. Un'altra vista grafica molto apprezzata è stata quella gerarchica a livello di directory, sottodirectory e file sorgenti. Tra i vari *report* utilizzati nella fase di comprensione del codice quelli più utilizzati sono stati: la gerarchia delle superclassi (relazione di ereditarietà tra classi), le variabili globali (relazione tra funzioni e variabili globali), i sottosistemi (relazione tra sottosistemi e artefatti contenuti) e le dipendenze tra i vari artefatti. Gli autori in [25] hanno concluso il loro lavoro di analisi con alcune considerazioni. Dato che il numero di routine e variabili globali è alto comparato con quello delle classi, il sistema sembra non aderire completamente al paradigma *Object Oriented*, ovvero il sistema è scritto in C++ ma segue uno stile imperativo (questa affermazione è confermata anche da altre analisi). Visto che sono presenti 412.315 linee di commento su 700.000 linee globali di codice, gli autori hanno concluso che il sistema è ben documentato.

triche del software). Nella figura 5 viene mostrata la vista "complessità del sistema" prodotta dal *CodeCrawler* usando come input un sistema medio scritto in Smalltalk. Le metriche usate sono il numero di attributi (larghezza dei nodi), il numero di metodi (lunghezza dei nodi) e il numero di linee di codice (colore) per ogni classe del sistema. Dalla figura 5 è possibile capire che il sistema presenta alcune classi con un numero

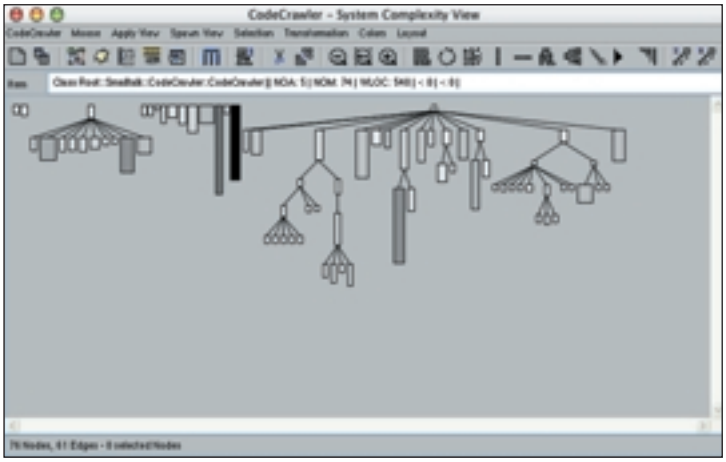


FIGURA 5

Vista "complessità del sistema" prodotta da CodeCrawler

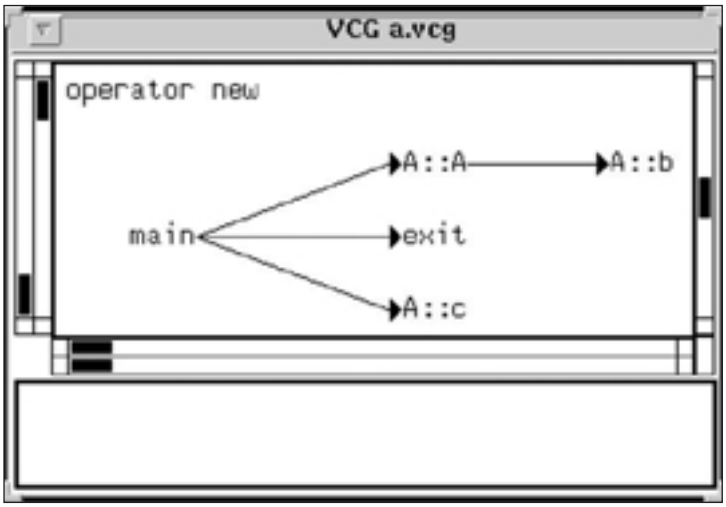


FIGURA 6

Grafo delle chiamate. Il main chiama il costruttore della classe 'A', 'exit' e il metodo 'c' della classe 'A'. A sua volta il costruttore della classe A chiama il metodo b

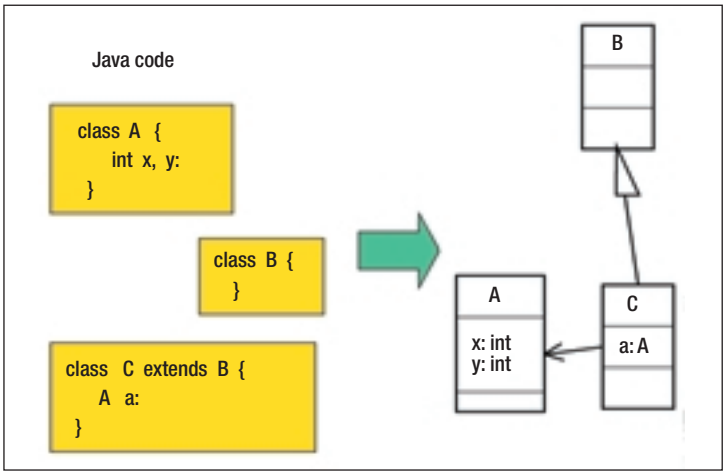


FIGURA 7

Estrazione del diagramma delle classi UML da codice Java

eccessivo di metodi e molte linee di codice e che quindi andrebbero ristrutturate;

3. ***generatori di diagrammi***: analizzano il codice sorgente e producono in output dei diagrammi che rappresentano alcune proprietà del sistema. Tra i diagrammi generati troviamo il "data flow", il grafo delle chiamate (Figura 6), il grafo delle dipendenze tra file e il grafo delle relazioni di ereditarietà tra le varie classi di un sistema ad oggetti. Un *tool* commerciale che rientra in questa categoria e produce in output diversi tipi di diagrammi, tra i quali il grafo delle chiamate e il grafo delle dipendenze tra file, è *Imagix 4D* [13];

4. ***tool di analisi***: recuperano dal codice sorgente un insieme di metriche utili ai fini della comprensione. Alcune metriche sono semplici, come le linee di codice o il numero di metodi/funzioni per file, altre sono più complesse come la coesione e l'accoppiamento tra moduli;

5. ***tool di recupero del design***: recuperano le relazioni interne tra le varie componenti software (per esempio, relazioni tra classi nei sistemi ad oggetti; Figura 7) e producono in output un diagramma che rappresenta, in un qualche formalismo, il design del sistema analizzato. Esempi di *tool* che ricadono in questa categoria sono *Rational Rose* [14] ed *EclipseUML* [15]. Questi *tool* estraggono diagrammi UML da codice Java;

6. ***tool di generazione ed analisi delle tracce di esecuzione***: l'uso di informazione dinamica, cioè informazione raccolta durante l'esecuzione del software, è stata utilizzata spesso nel contesto del *reverse engineering* ad integrazione delle tecniche statiche (cioè quelle tecniche che si basano solo sull'analisi del codice). Le tecniche statiche sono conservative, nel senso che ogni possibile comportamento del sistema sotto analisi è rappresentato nei risultati, ma meno potenti di quelle dinamiche (problema dell'indecidibilità di alcune asserzioni a livello statico). Le tecniche di analisi dinamica hanno tra i vantaggi la garanzia dell'esistenza di almeno un caso in cui il comportamento del sistema è quello descritto dall'analisi. Tra gli svantaggi troviamo la parzialità delle analisi (non tutti i comportamenti del sistema sono rappresentati nei risultati) e il limite in termini di scalabilità ed interpretazione dei risultati;

7. ***tool di pattern matching***: ricercano, all'interno del codice, i cosiddetti pattern, ovvero soluzioni software consolidate per un problema ricorrente. I pattern possono essere a diversi livelli di astrazione: a livello di design (design pattern [5]), a livello di architettura (per esempio, l'architettura *client-server*) e a livello concettuale. Il riconoscimento in modo automatico di alcuni *pattern* a livello sintattico o meglio ancora di alcuni concetti a livello semantico [20] semplifica la fase di comprensione del codice.

8. LIMITI E POTENZIALITÀ

Bisogna essere realistici a riguardo dei risultati che si possono ottenere da un'operazione di *reverse engineering*. Sia in ambito industriale che in quello accademico è risaputo che il *reverse engineering* è un compito laborioso, difficile e dai costi molto elevati. Inoltre il *reverse engineering* non è mai fine a se stesso; dopo la fase di *reverse engineering* spesso seguono fasi altrettanto difficili e laboriose come il *re-structuring* o il *re-engineering* (vedere il riquadro).
Il limite più grande di questo approccio è che una parte sostanziale di lavoro deve essere svolta manualmente. I *tool* sono utili, ma vanno "guidati" nel processo di recupero delle informazioni, ed inoltre i risultati ottenuti non sono quasi mai immediatamente usabili; per essere utili devono essere raffinati manualmente.
I risultati prodotti in modo automatico devono essere, da un lato, integrati, infatti, non tutte le informazioni utili possono essere ricavate dal codice o dalla sua esecuzione (*informazione mancante*) e, dall'altro, filtrati (*sovrabbondanza di informazione*). Capita spesso che la quantità di informazioni prodotte da un *tool* di *reverse engineering* sia ingestibile.
Per esempio, nel caso dei *tool* che estraggono diagrammi UML spesso l'informazione mancante è molta:

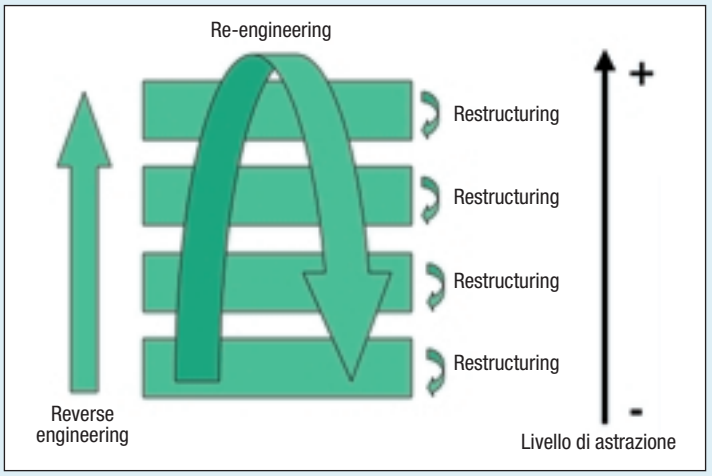
- una volta implementate, aggregazione e associazione sono indistinguibili, per cui il *tool* non può scegliere tra le due (solitamente quando non è possibile decidere il *tool* fa la scelta più conservativa, ovvero opta per le associazioni);
- la molteplicità non può essere recuperata dal codice (staticamente è indecidibile determinare il numero di entità coinvolte in una data relazione);

Forward engineering, restructuring e re-engineering

Sono termini strettamente connessi al *reverse engineering* [2].

Forward engineering è il tradizionale processo di sviluppo del software. Inizia con l'analisi dei requisiti e termina con l'implementazione del sistema. In un certo senso il *reverse engineering* può essere considerato l'operazione inversa. Si analizza il codice sorgente con l'intenzione di derivare una rappresentazione più astratta del sistema di partenza.

Restructuring è una trasformazione di un programma da una rappresentazione ad un'altra che preserva il comportamento esterno del sistema, ovvero le funzionalità. I *refactoring* proposti da Fowler [4] ricadono in questa categoria. È importante sottolineare che la trasformazione avviene allo stesso livello di astrazione, ovvero tra codice e codice oppure tra design e design. Un esempio di *restructuring* a livello di codice è la conversione di un programma che fa uso di salti incondizionati (codice a "spaghetti") in uno strutturato senza i comandi di "goto". Un esempio di *restructuring* a livello di design è invece la sostituzione di una struttura dati con un'altra (per esempio: sostituire il *file system* con un DBMS).



Relazione tra i vari termini

Re-engineering è un operazione composta da due attività (si veda la Figura): analisi/comprensione del sistema (*reverse engineering*) e ricostruzione dello stesso in una nuova forma (*forward engineering*). Il processo di *re-engineering* può includere, a differenza del *restructuring*, delle modifiche rispetto ai requisiti del sistema originale.
Un'operazione di *re-engineering* può essere eseguita per diverse ragioni. La migrazione di sistemi *legacy*, il riuso o la sicurezza del codice, la migrazione verso linguaggi più evoluti (per esempio: da C a Java) oppure la migrazione di sistemi sul Web.

• non è possibile ricavare i nomi delle relazioni, i ruoli, i vincoli e le proprietà;
• quando vengono utilizzati i contenitori debolmente tipati (per esempio, in Java 1.4 i *Vector* o le *LinkedList*) le classi effettive degli oggetti contenuti non possono essere recuperate staticamente.

L'altro difetto dei *tools* di estrazione del design è la sovrabbondanza di informazione. Spesso i diagrammi recuperati sono troppo grandi e quindi poco usabili. In un sistema ad oggetti medio, nell'ordine delle 20 mila linee di codice, è abbastanza comune avere 50-100 classi. In casi tipici come questo, il diagramma delle classi estratto dal codice è difficile da leggere per un umano le cui abilità cognitive sono attorno alle 10 entità come limite approssimativo.

Esistono due possibili soluzioni per cercare di risolvere questo problema. Il filtraggio, operazione guidata dall'utente che esclude dal diagramma generato l'informazione irrilevante, e le viste multiple [24]. Con le viste multiple l'utente suddivide il sistema in viste e, durante la fase di *reverse engineering*, decide quali elementi (per esempio in un sistema ad oggetti: classi, metodi, campi ecc.) appartengono a quale vista.

Le tecniche di *reverse engineering*, originariamente pensate per i sistemi *legacy*, si sono rivelate potenzialmente utili anche in supporto alla comprensione di sistemi software scritti secondo approcci più moderni. In particolare le tecniche di *reverse engineering* sono potenzialmente applicabili ai sistemi *open source* [19] che solitamente sono rilasciati senza o con poca documentazione. Un'altra potenzialità è nell'ambito dei processi agili (per esempio, *Extreme Programming* [6]). Nei processi agili si riconosce la centralità del codice; nel ciclo di sviluppo non sono previste esplicitamente né la fase di analisi né la fase di design. Questo fa sì che non vi sia necessariamente della documentazione ad accompagnare il codice sorgente.

9. CONCLUSIONI

I sistemi *legacy* non sono dei dinosauri in via di estinzione, bensì dei sistemi molto difficili da trattare con i quali dobbiamo convivere. Il *reverse engineering* di questi sistemi è un compito laborioso, difficile e molto costoso. I *tool* possono aiutare in questo processo anche se però bisogna essere realistici sui risultati che si possono ottenere. Se consideriamo come valida la definizione forte di *reverse*, ovvero processo di recupero delle specifiche in modo automatico, allora dovremmo concludere che il *reverse engineering* ha fallito lo scopo. Lo stato dell'arte è molto lontano da questi risultati e non si sa nemmeno se in futuro questi potranno essere raggiunti. Se invece indeboliamo la definizione e ci accontentiamo di recuperare viste astratte in modo semi-automatico, ovvero consideriamo il *reverse engineering* come un processo di estrazione cooperativa (*tool* automatici e programmatori), allora potremmo concludere che l'ingegneria inversa è stata ed è un successo.

Bibliografia

[1] Bennett K.: *Legacy systems: coping with stress*. IEEE Software, Vol. 12, Issue 1, Jan. 1995, p. 19-23.

[2] Chikofsky E.J., Cross J.H.: *Reverse Engineering and Design Recovery: A Taxonomy*. IEEE Software, IEEE Computer Society, January 1990, p. 13-17.

[3] Ducasse S., Girba T., Lanza M., Demeyer S.: Moose - A Collaborative and Extensible Reengineering Environment. *Capitolo del libro Tools for Software Maintenance and Reengineering, RCOST /Software Technology Series*. Franco Angeli, 2005, p. 55-71.

[4] Fowler M.: Refactoring: *Improving the Design of Existing Code*. Addison-Wesley Professional, 1999.

[5] Gamma E., Helm R., Johnson R., Vissides J.: *Design patterns: elements of reusable object-oriented software*. Addison Wesley, October 1994.

[6] Ghezzi C., Monga M.: Extreme programming: programmazione estrema o revisionismo estremista?. *Mondo Digitale*, n. 4, dicembre 2002.

[7] http://en.wikipedia.org/wiki/Reverse_engineering

[8] http://smallwiki.unibe.ch/codecrawler/anon-exhaustivelistofsoftwarevisualizationtools/

[9] http://www.rigi.csc.uvic.ca.

[10] http://www.semdesigns.com/

[11] http://www.fatesoft.com/s2f/

[12] http://sourcenav.sourceforge.net/

[13] http://www.imagix.com/products/screens.html

[14] http://www-306.ibm.com/software/awdtools/developer/rosexde/

[15] http://www.omondo.com/

[16] Lanza M.: *CodeCrawler - Lessons Learned in Building a Software Visualization Tool*. CSMR 2003, 7-th European Conference on Software Maintenance and Reengineering, IEEE Computer Society, p. 409-418.

[17] Lehman M.M., Belady L.: *Program evolution – Processes of software change*. London, Academic Press, 1985.

[18] Massari A., Mecella M.: Il trattamento dei legacy system. *Capitolo 2 del libro Sistemi informativi*, Vol. 5. Franco Angeli, 2001.

[19] Meo A.R.: Software libero e Open Source. *Mondo Digitale*, n. 2, giugno 2002.

[20] Gold N.: *Hypothesis-Based Concept Assignment to Support Software Maintenance*. ICSM 2001, 17-th IEEE International Conference on Software Maintenance, IEEE Computer Society, p. 545-454.

[21] Ousterhout J.K.: *An introduction to Tcl and Tk*. Addison-Wesley, 1994.

[22] Quilici A.: *Reverse engineering of legacy systems: a path toward success*. ICSE 1995, 17-th International Conference on Software engineering, Seattle, Washington, United States, p. 333-336.

[23] Sommerville I.: *Software engineering*. Addison Wesley, 2000.

[24] Spinellis D.: *Drawing UML Diagrams with UMLGraph*. Disponibile all'indirizzo: http://www.spinellis.gr/sw/umlgraph/.

[25] Wong K., Tilley S.R., Müller H.A., Storey M.-A.D.: *Structural redocumentation: A case study*. IEEE Software, January 1995, p. 46-54.

FILIPPO RICCA è un ricercatore presso l'ITC-irst, Centro per la Ricerca Scientifica e Tecnologica, di Trento. Si è laureato e ha ottenuto il titolo di dottore di ricerca presso l'Università di Genova (Dipartimento di Informatica a Scienze dell'Informazione). Dal 1999 fa parte del gruppo di ricerca in Software Engineering dell'ITC-irst. I suoi interessi di ricerca attuali includono il reverse engineering, gli studi empirici, le applicazioni Web, il testing e le trasformazioni automatiche di codice.
E-mail: ricca@itc.it

PAOLO TONELLA è un ricercatore senior presso l'ITC-irst, Centro per la Ricerca Scientifica e Tecnologica, di Trento. Si è laureato e ha ottenuto il titolo di dottore di ricerca presso l'Università degli Studi di Padova. Dal 1994 fa parte del gruppo di ricerca in Software Engineering dell'ITC-irst. È autore del libro "Reverse Engineering of Object Oriented Code", pubblicato da Springer nel 2005. I suoi interessi di ricerca attuali includono il reverse engineering, la programmazione orientata agli aspetti, gli studi empirici, le applicazioni Web ed il testing.
E-mail: tonella@itc.it

ICT E INNOVAZIONE D'IMPRESA
Casi di successo

Rubrica a cura di
Roberto Bellini, Chiara Francalanci

La rubrica *ICT e Innovazione d'Impresa* vuole promuovere la diffusione di una maggiore sensibilità sul contributo che le tecnologie ICT possono fornire a livello di innovazione di prodotto, di innovazione di processo e di innovazione di management. La rubrica è dedicata all'analisi e all'approfondimento sistematico di singoli casi in cui l'innovazione ICT ha avuto un ruolo critico rispetto al successo nel business, se si tratta di un'impresa, o al miglioramento radicale del livello di servizio e di diffusione di servizi, se si tratta di una organizzazione pubblica.

Il ruolo dell'informazione nel settore dei trasporti: il caso Tarasconi Trasporti

Eugenio Capra

1. Introduzione

L'informazione ha sempre rivestito un ruolo estremamente importante in tutte le attività dell'uomo, ben prima dell'avvento dell'informatica e delle moderne tecnologie ICT. L'informazione sta alla base del coordinamento e della gestione di quasi tutti i processi produttivi ed è l'elemento sopra al quale può essere costruita la conoscenza, che è data dall'analisi e dalla rielaborazione delle informazioni. La conoscenza a sua volta abilita l'evoluzione dei processi, in quanto permette di "non perder memoria" di quanto fatto da altri attori, in altri contesti geografici e temporali.
In tutte le attività organizzate dell'uomo sono sempre stati presenti strumenti di gestione della conoscenza, anche se molto spesso impliciti e non formalizzati: riunioni, discussioni, scoperte e competenze acquisite e tramandate oralmente, documenti scritti su carta, manuali, formulari ecc..
Il termine *Knowledge Management* viene coniato nel 1986 da Karl Wiig (si veda [1, 3]) durante una conferenza allestita dall'Organizzazione Internazionale dei Lavoratori delle Nazioni Unite. Indica un insieme di metodologie e processi finalizzati a gestire in modo ottimale le conoscenze aziendali critiche, necessarie per conseguire specifici obiettivi di miglioramento di singole attività nel breve termine o per sostenere il vantaggio competitivo dell'azienda nel medio-lungo termine [2]. È evidente come la tecnologia possa costituire un supporto fondamentale per lo sviluppo del Knowledge Management, sia per l'intensificazione e il potenziamento dei flussi di comunicazioni che rende possibili, sia per lo sviluppo della capacità di memorizzazione, ricerca ed elaborazione dei dati.
È interessante studiare come la tecnologia venga utilizzata a supporto della gestione della conoscenza in quei settori tradizionalmente più legati alla materialità delle Operations e in cui il ruolo dell'informatica è, anche se solo in apparenza, secondario. In questo articolo viene presentato il caso di Tarasconi Trasporti S.r.l [4], azienda operante nel settore del trasporto intermodale di merci pericolose, ma molto attenta ai vantaggi potenziali derivanti dall'ICT. L'informazione e la conoscenza costituiscono un fattore strategico all'interno del settore, e Tarasconi Trasporti sfrutta le nuove tecnologie dell'informazione per far sì che l'informazione diventi una leva per consolidare la propria posizione di leadership.
La presentazione è strutturata come segue: il Paragrafo 2 descrive a livello generale l'azienda esaminata e la contestualizza all'interno del settore di riferimento; il Paragrafo 3 discute il ruolo che l'informazione e la conoscenza hanno in Tarasconi Trasporti; il Paragrafo 4 presenta i

sistemi di Knowledge Management impiegati e le tecnologie di supporto adottate; infine, il Paragrafo 5 conclude la trattazione.

2. Tarasconi Trasporti e il settore del trasporto intermodale

Tradizionalmente in Italia il trasporto merci su strada ricopre un particolare importanza per motivi sia storici che strutturali, tra cui la capillarità della rete viaria e la quasi totale assenza di reti fluviali interne, molto più utilizzate in altri stati europei. Con l'intensificarsi delle relazioni commerciali con gli altri stati europei, favorite dall'allargamento dei confini dell'Unione Europea, a livello internazionale sempre più attenzione è posta alla realizzazione di nuovi e importanti assi di comunicazioni fra i vari stati.
In questo contesto negli ultimi anni ha assunto particolare rilevanza strategica il trasporto intermodale, vale a dire una pianificazione del percorso delle merci che combina tutti i mezzi a disposizione: strada, rotaia, nave. Il trasporto intermodale permette di ridurre i tempi di trasporto complessivi ottimizzando i percorsi e di aumentarne l'efficienza, sfruttando le economie di scala che derivano dall'utilizzo del trasporto via treno e via nave. Tale efficienza inoltre si riflette indirettamente sulla riduzione del traffico stradale e dell'inquinamento complessivo a parità di merce trasportata.
Tarasconi Trasporti S.r.l, nata come ditta individuale e artigiana nel 1955, è stato uno dei primi operatori a puntare sull'intermodalità fin dal 1988, ed è oggi un'azienda leader nel trasporto intermodale di merci pericolose.
Potendo contare su di un parco attrezzature composto da più di 100 unità, con cisterne e tank container specifici per il trasporto di prodotti chimici, petrolchimici e reflui industriali, Tarasconi Trasporti ha un fatturato che supera i 5 milioni di euro annui. L'azienda riveste un ruolo di rilievo anche a livello europeo, infatti più del 35% del suo fatturato deriva dai servizi prestati fuori dai confini nazionali. Il ruolo strategico che l'intermodalità costituisce per Tarasconi Trasporti è dimostrato dal fatto che il 95% dei servizi prestati viene effettuato combinando l'impiego della strada a quello della rotaia e del mare, riuscendo ad affidare a subvettori meno dell'1% delle commesse.
Il settore dei trasporti in generale è molto competitivo, in quanto il mercato tende a dare molta importanza al costo del servizio. La forte competizione sul prezzo si affianca ad un andamento crescente dei costi del trasporto, difficilmente influenzabile in quanto strettamente legato al costo del greggio.
Per avere successo nel settore è quindi necessario puntare anche sulla leva della differenziazione del prodotto, curando l'immagine aziendale e offrendo servizi a valore aggiunto ai propri clienti. I fattori premianti principali riconosciuti dal mercato sono:

- la rapidità di consegna;
- la tracciabilità della merce;
- la sicurezza del trasporto.

La rapidità di consegna è sostanzialmente legata alla disponibilità di mezzi e ad un'accurata pianificazione del tragitto. È importante ottimizzare i viaggi compiuti dai mezzi, per esempio minimizzando i viaggi a vuoto effettuati per riportare i mezzi al deposito o per trasportare gli autisti. La tracciabilità della merce costituisce un servizio a valore aggiunto per molti clienti, che desiderano sapere in tempo reale dove si trova il carico e quanto tempo è necessario alla consegna. Questo aspetto è enfatizzato dal fatto che Tarasconi Trasporti è specializzata nel trasporto di merci pericolose, per cui lo smarrimento o il danneggiamento del carico è considerato un evento critico non solo per il valore economico, ma anche per l'impatto a livello di immagine. La sicurezza in questa particolare tipologia di trasporti è molto importante per la relazione con le autorità locali, con le assicurazioni e per l'immagine dell'azienda in generale. I continui investimenti di Tarasconi Trasporti in attrezzature di sicurezza per le cisterne, l'attenzione alla selezione e alla formazione del personale insieme ad una cultura aziendale molto orientata alla qualità ha fatto sì che gli incidenti che coinvolgono il prodotto siano praticamente nulli, mentre il numero di incidenti che coinvolgono le motrici non è mai superiore a 1-2 all'anno.

3. Il ruolo dell'informazione e della conoscenza in Tarasconi Trasporti

La conoscenza e la gestione dell'informazione in generale in aziende come la Tarasconi Trasporti possono diventare critiche per poter differenziare la propria offerta secondo le dimensioni di qualità illustrate nella sezione prececente.

Tralasciando le forme di conoscenza tacita e implicita, che pur rivestendo un ruolo importante nella conduzione dell'azienda sono difficilmente valorizzabili, due tipologie di conoscenza appaiono particolarmente critiche in Tarasconi Trasporti:

- la conoscenza esatta dello stato e della posizione delle proprie unità operative (conoscenza "interna");
- la conoscenza del mondo dei trasporti intesa in senso più generale come risorsa condivisa all'interno del settore (conoscenza "esterna").

La conoscenza "interna" delle proprie unità operative è essenziale per ottenere efficienza e per poter offrire ai propri clienti servizi a valore aggiunto, come la tracciabilità della merce. Sapere esattamente in ogni istante dove si trovano i proprio camion, con quali carichi a bordo, con quale autista in servizio e magari sapere anche se in zona ci sono altri autisti che possono sostituire quelli alla guida è un tipo di informazione estremamente preziosa, che opportunamente elaborata può generare conoscenza utilizzabile per pianificare al meglio le tratte e ridurre al minimo i tempi di percorrenza. Questo tipo di conoscenza diventa ancora più importante in presenza di un modello di business fortemente orientato alla multimodalità, dove i mezzi da gestire sono di tipologie profondamente diverse e i tempi rischiano di essere meno controllabili.

Conoscere lo stato di tutte le proprie attrezzature, la data dell'ultima revisione e le eventuali anomalie e avarie riscontrate è un tipo di informazione che può permettere di pianificare in modo estremamente efficiente gli interventi di manutenzione sia ordinaria che straordinaria. Questo si traduce in alti livelli di sicurezza e in costi minori per aderire agli standard europei (ADR, Accordo europeo sul trasporto di merci pericolose).

Tutte queste informazioni prese singolarmente sono ben codificate e documentate (per esempio, su ogni camion è presente una bolla con il dettaglio del carico trasportato, così come per ogni motrice viene redatta una scheda che riporta gli interventi di manutenzione effettuati), questo tipo di conoscenza è cioè esplicito e formalizzato. Il problema è queste informazioni sono "sparse" sulle varie unità operative che per la natura stessa di un'azienda di trasporti sono a loro volta sparse sul territorio, e non è semplice averne una visione d'insieme.

La tecnologia si rivela un fattore abilitante proprio per la sua capacità di aggregare tutte le informazioni rendendole facilmente consultabili e di elaborarle in modo unitario. Tarasconi Trasporti si avvale dello strumento Voyager, descritto nel prossimo paragrafo, per gestire questo tipo di informazioni.

Accanto alla conoscenza "interna" vi è poi una conoscenza "esterna" relativa a tutto quanto gravita intorno al mondo dei trasporti: notizie, normative, informazioni sui concorrenti o sui potenziali partner, informazioni sul traffico o sulla viabilità ecc.. Si tratta di tutte quelle informazioni che, a differenza della posizione e dello stato delle proprie unità, sono poco formalizzate e scarsamente documentate, in quanto non sono direttamente controllate dall'azienda. Sono informazioni d'interesse più ampio, di cui possono usufruire tutte le aziende del settore nonché i potenziali clienti. Lo strumento ideale per raccogliere e diffondere questo tipo di conoscenza è un portale internet dedicato coordinato da un team editoriale specializzato. A questo scopo Tarasconi Trasporti, tramite la controllata IRIX, ha sviluppato e gestisce il portale Felixia, che sarà descritto nel prossimo paragrafo.

4. Sistemi di Knowledge Management a supporto del business

Tarasconi Trasporti è un'azienda molto attenta all'evolversi delle nuove tecnologie ICT e da anni promuove progetti per favorire l'innovazione e applicare queste tecnologie all'interno del proprio business di riferimento. Molti di questi progetti sono svolti in collaborazione con università ed enti di ricerca. Particolare attenzione è stata rivolta ai sistemi di supporto alla gestione della conoscenza, intesa come rielaborazione di dati ed informazioni elementari. In questo paragrafo saranno descritti due sistemi di Knowledge Management adottati da Tarasconi Trasporti: Voyager, una piattaforma gestionale volta alla gestione della conoscenza "interna", e Felixia, un portale di riferimento per il settore dei trasporti destinato alla gestione della conoscenza "esterna" all'azienda.

4.1. Voyager

Voyager è un applicativo che unisce le caratteristiche dei software gestionali di tipo ERP (*Enterprise Resource Planning*) e degli strumenti più specifici per il Knowledge Management, infatti aggre-

ga la conoscenza di tipo "interna" (si veda il paragrafo precedente) sparsa all'interno dell'azienda e la utilizza per ottimizzarne i processi operativi.
Voyager comprende un'ampia serie di funzionalità e fornisce un supporto completo al processo di gestione del trasporto: dalla gestione della fatturazione a quella della manutenzione di veicoli ed attrezzature, dalla programmazione delle consegne al tracciamento dei veicoli su strada mediante sistema GPS, dalla gestione statistica degli ordini all'abbinamento degli ordini alle attrezzature.
In particolare, le funzionalità principali offerte dal programma sono:

- gestione dell'ordine;
- gestione del trasporto;
- manutenzione;
- fatturazione;
- scheduling.

La funzione di "gestione dell'ordine" prevede la possibilità di inserire gli ordini di trasporto in un data base, strutturati secondo data di carico, data dell'ordine e committente. Il software applicativo permette la conferma automatica dell'ordine via fax e abilita una gestione statistica degli ordini.
Per quanto riguarda il trasporto, Voyager permette di conoscere in tempo reale l'abbinamento degli ordini alle attrezzature di trasporto, la registrazione di ogni singolo evento durante i viaggi, la posizione di ogni mezzo della flotta, divisi per zona geografica, e l'indicazione dello stato dell'attrezzatura. Questo è reso possibile dalla presenza a bordo dei veicoli di ricevitori GPS che inviano tutte le informazioni alla centrale di controllo tramite la rete GPRS. In questo modo il cliente può conoscere in ogni momento la localizzazione della propria merce semplicemente telefonando alla centrale operativa. Attualmente non è previsto il servizio di localizzazione lungo le tratte ferroviarie o navali, in quanto i vettori utilizzati non sono gestiti direttamente da Tarasconi Trasporti, ma da società terze.
Tutte le informazioni rilevate vengono salvate in memoria ed utilizzate per elaborazioni statistiche relative ad attrezzature, autisti, stabilimenti e località. Per esempio, il sistema calcola automaticamente indici di utilizzo delle attrezzature, i chilometri percorsi da ogni mezzo, l'affidabilità degli autisti ed altri indici utili per analizzare le prestazioni del processo di trasporto. Queste informazioni sono preziose per il management dell'azienda, che le può utilizzare per stabilire i bonus da erogare agli autisti, per capire il grado di saturazione dei propri mezzi, per monitorare in modo preciso i propri costi ed in generale per verificare l'efficacia delle strategie adottate.
L'applicativo permette inoltre di programmare la manutenzione di tutte le attrezzature, registrando le date in cui vengono eseguite, e di gestire le statistiche delle manutenzioni effettuate. Ogni volta che viene eseguito un intervento viene registrata manualmente la data in un apposito database, in modo da tener traccia delle revisioni effettuate per ogni mezzo.
Nella fase di fatturazione il software gestisce l'emissione e la stampa delle fatture e delle note di credito. Una volta inseriti i dati relativi alla bolla di trasporto il sistema genera automaticamente la fattura e la invia direttamente via fax al cliente.
Tutte queste attività sono supportate da un tool di scheduling delle attività, con il quale Voyager registra le revisioni delle attrezzature, le schede di tutti i veicoli (ADR, Accordo europeo sul trasporto di merci pericolose), i collaudi delle cisterne e degli accessori e i dati relativi alle patenti degli autisti. Il sistema permette quindi di pianificare le date dei diversi interventi e di inviare le dovute comunicazioni al management o al personale competente in modo che possano provvedere alla realizzazione di queste attività.
Voyager è stato sviluppato *ad hoc* nel 1993 da IRIX S.r.l [5], società controllata da Tarasconi Trasporti, basandosi sulla piattaforma Microsoft Access®. Da allora però il sistema è stato continuamente migliorato e modificato, anche in considerazione dell'importanza che assume nel processo di gestione dei trasporti. Attualmente è in corso lo sviluppo di una nuova versione basata su Visual Basic 6 ®, al fine di rispondere al meglio alla continua espansione e all'ammodernamento dell'azienda.
Nell'opinione del management di Tarsconi Trasporti S.r.l, il sistema Voyager ha avuto nel corso degli anni un impatto enorme sulla conduzione dell'azienda, aiutandola a guadagnare una quota sempre maggiore di mercato, ed ha costituito un forte differenziale competitivo rispetto ai suoi concorrenti.

4.2. Felixia

Felixia [6] è un motore di ricerca per il mondo dei trasporti (Figura 1). L'obiettivo del sistema consiste nel fornire agli operatori del settore

servizi informativi e interattivi personalizzati. Attraverso Felixia l'utente è in grado di effettuare ricerche libere, consultare categorie di ricerca predefinite e reperire informazioni e link a risorse di rete dedicate al mondo dei trasporti (per esempio, informazioni sulla viabilità e sul traffico). Scopo di Felixia è catalizzare e rendere facilmente fruibile tutta la conoscenza "esterna" alla Tarasconi Trasporti, ma di forte interesse per il settore dei trasporti.
Felixia è stato realizzato nel 2000 ed è gestito da IRIX S.r.l., si appoggia ad una base dati di circa 2.000 record e ad oggi conta una media di circa 7.000 visitatori/mese e 700 imprese iscritte al servizio di mailing list, a testimonianza del successo dell'iniziativa. È stato completamente aggiornato nel 2002, quando da semplice motore di ricerca si è trasformato in portale specializzato, risorsa Internet di riferimento per gli operatori del settore trasporti e logistica.
I principali servizi offerti da Felixia sono:

- ❑ indirizzario e base dati di tutte le aziende del settore;
- ❑ collegamenti a risorse specifiche (per esempio, informazioni sulla viabilità);
- ❑ rassegna stampa virtuale;
- ❑ mailing list;
- ❑ calendario eventi del settore;
- ❑ sensibilizzazione alle tematiche del settore, informazione su progetti in corso, opportunità formative ecc..

La banca dati di Felixia contiene tutte le informazioni di contatto delle aziende fornitrici di servizi, come indirizzo, ragione sociale, categoria/e di trasporto e il sito internet in modo da renderle reperibili al pubblico. In questo modo Felixia funge da "vetrina" per gli operatori del settore e da punto di incontro virtuale tra domanda e offerta.
Il database viene aggiornato sulla base delle richieste di registrazione inviate dagli utenti e sulla base di ricerche svolte dal comitato editoriale per reperire link a risorse informative di pubblica utilità. In particolare, le aziende che vogliono comparire sul portale compilano un modulo di richiesta direttamente online. Le richieste vengono esaminate dal comitato editoriale che verifica la correttezza delle informazioni fornite; se i controlli danno esito positivo l'azienda viene aggiunta agli altri contatti e diventa reperibile tramite questo motore di ricerca. Questo servizio offerto da Felixia è totalmente gratuito, ma oltre a questo c'è la possibilità di sponsorizzazioni

FIGURA 1
Homepage di Felixia (www.felixia.it)

tramite banner che compaiono nel sito o tramite l'indicizzazione, che permette alle aziende che vogliono usufruirne di comparire ai primi posti nelle ricerche effettuate dagli utenti.
Con il rinnovamento di Felixia avvenuto nel 2002 all'elenco delle aziende del settore sono stati aggiunti servizi informativi altamente specializzati: rassegna stampa, calendario eventi, collegamenti a risorse Web specifiche, collegamenti a istituzioni ed enti di ricerca, illustrazione delle principali normative.
Felixia press review è un servizio di rassegna stampa virtuale specializzata nel settore dei trasporti e della logistica. Offre una recensione completa della stampa di settore, italiana ed europea, e delle principali risorse informative disponibili in rete. La consultazione degli articoli è gratuita e può avvenire tramite la navigazione delle rubriche tematiche oppure tramite la funzione di ricerca per parola chiave. L'utente può segnalare degli articoli da pubblicare in *Felixia press review* compilando e inviando online un apposita richiesta di registrazione. Inoltre il servizio permette di ricevere direttamente nella casella di posta elettronica le notizie aggiornate sugli argomenti di interesse maggiore iscrivendosi alla mailing list. Oltre al servizio di rassegna stampa è possibile la consultazione on-line delle maggiori riviste del settore, come *Lo spedizioniere doganale magazine, Logistica delle sostanze pericolose, Logistica Management e Professione camionista*.

Il *calendario eventi* permette di informare le imprese del settore logistico riguardo a notizie, eventi, iniziative, ma offre anche la possibilità di informare le aziende sulle principali normative nazionali e comunitarie inerenti il trasporto, così da fare chiarezza nel complesso di norme che regolano il mondo della logistica.
Un'altra caratteristica del portale è infatti il ruolo svolto per sensibilizzare gli operatori alle problematiche internazionali del settore e ai nuovi indirizzi della politica nazionale ed europea dei trasporti, per lo più incentrati sullo sviluppo dell'intermodalità, la valorizzazione del trasporto ferroviario e navale, il trasferimento delle merci dalla strada ad altre vie di trasporto, l'innovazione tecnologica e l'informatizzazione.
Felixia dà visibilità alle attività di enti nazionali e internazionali, istituti di ricerca, scuole e università, valorizzando i progetti nazionali e transnazionali concernenti lo scambio informatico di dati tra strutture portuali e interportuali, con l'obiettivo di monitorare il trasporto merci.
Vengono inoltre forniti i link ai siti di riferimento dei progetti attualmente in corso e dei corsi di formazione disponibili.
In conclusione, Felixia costituisce un punto di riferimento per il settore dei trasporti, sia perché offre visibilità alle aziende registrate sia per il suo ruolo di aggregazione, organizzazione e distribuzione della conoscenza. Tarasconi Trasporti S.r.l ha un chiaro interesse strategico a coordinare e dirigere questa struttura, che le permette di consolidare la sua immagine e il suo ruolo di leader all'interno del settore.

5. Conclusioni

In questo articolo si è evidenziato il ruolo strategico che l'informazione e la conoscenza, nelle sue varie forme, posso rivestire all'interno di un settore ancora tradizionalmente poco legato all'informatica come quello dei trasporti.
I sistemi informatici di gestione e di Knowledge Management utilizzati da Tarasconi Trasporti nell'opinione del management dell'azienda hanno sicuramente contribuito e contribuiscono a consolidare la posizione di leadership all'interno del settore e del mercato di riferimento.
In sintesi, l'applicativo Voyager ha permesso di:
- ridurre i costi e i tempi di gestione;
- ridurre gli errori e aumentare la sicurezza;
- pianificare ed automatizzare molti processi ripetitivi, come la manutenzione delle attrezzature.

Lo sviluppo di Felixia ha invece permesso a Tarasconi Trasporti di:
- ampliare il suo campo di azione;
- consolidare la sua posizione di leader tra le aziende del settore;
- migliorare la sua immagine attraverso i banner pubblicitari.

In generale, il caso Tarasconi dimostra ancora una volta come l'ICT non sia solo un costo, ma come l'impiego della tecnologia insieme ad un'adeguata cultura aziendale e organizzazione dei processi possa avere un impatto molto positivo sulle performance di un'azienda.

Ringraziamenti

Un particolare ringraziamento va a Katia e Maurizio Tarasconi per la disponibilità a rilasciare informazioni sulla loro azienda. Si ringraziano inoltre la prof. Chiara Francalanci per i preziosi consigli e gli studenti Alessandro Caleffi e Alessandro Cassi per l'aiuto fornito nell'analisi delle informazioni ricevute.

Bibliografia

[1] Wiig K.: *Knowledge Management Foundations: Thinking about Thinking – How Organizations Create, Represent and Use Knowledge*. Schema Press, 1993.

[2] Minghetti M.: Nel labirinto del Knowledge Management. *HAMLET, AIDP*, n. 18-1, 2000, p. 4.

[3] Sito internet: www.kriig.com

[4] Sito internet: www.tarasconi.it

[5] Sito internet: www.irixweb.com

[6] Sito internet: www.felixia.it

Eugenio Capra si è laureato in Ingegneria Elettronica nel 2003 presso il Politecnico di Milano, ha quindi lavorato come business analyst per McKinsey & Co. fino al 2005. Da marzo 2005 sta frequentando il Dottorato di Ricerca in Ingegneria dell'Informazione presso il Politecnico di Milano. Le sue attività di ricerca principali riguardano i costi di sviluppo e manutenzione del software in ambiente open source e l'impatto dell'IT sui processi di business. Collabora con la Fondazione Politecnico di Milano e svolge attività di consulenza nel settore finanziario e dell'IT.
E-mail: capra@elet.polimi.it

ICT E DIRITTO

Rubrica a cura di

Antonio Piva, David D'Agostini

Scopo di questa rubrica è di illustrare al lettore, in brevi articoli, le tematiche giuridiche più significative del settore ICT: dalla tutela del *domain name* al *copyright* nella rete, dalle licenze software alla *privacy* nell'era digitale. Ogni numero tratterà un argomento, inquadrandolo nel contesto normativo e focalizzandone gli aspetti di informatica giuridica.

La Posta Elettronica Certificata

Maurizio Blancuzzi, David D'Agostini, Antonio Piva

1. Introduzione

Dal 1° gennaio 2006 è in vigore il Codice dell'amministrazione digitale (approvato con d.lgs. 82/05), provvedimento, che accorpa la normativa sull'utilizzo delle tecnologie informatiche nella pubblica amministrazione, destinato a definire i caratteri futuri dell'impiego intensivo delle nuove tecnologie[1]. Dal testo del decreto emerge il "diritto all'uso delle tecnologie", ovvero il diritto riconosciuto a cittadini e imprese di richiedere e ottenere l'uso della telematica nelle comunicazioni con le pubbliche amministrazioni.

La via maestra attraverso la quale si esplica tale diritto è rappresentata dalla posta elettronica, soprattutto se certificata, ovvero se conferisce prova della comunicazione come finora è avvenuto con la raccomandata postale con ricevuta.

Infatti la posta elettronica o e-mail (abbreviazione di electronic mail) risulta ormai lo strumento di comunicazione elettronica più utilizzato per lo scambio di comunicazioni via Internet, il cui principale vantaggio è l'immediatezza di trasmissione; i messaggi possono includere testo, immagini, audio, video o qualsiasi tipo di file. La *Posta Elettronica Certificata* (PEC) è un sistema di posta elettronica nel quale è fornita al mittente documentazione elettronica, con valenza legale, attestante l'invio e la consegna di documenti informatici.

In particolare il menzionato Codice agli articoli 45 e 48 dispone che il documento informatico trasmesso per via telematica si intende spedito dal mittente se inviato al proprio gestore e si intende consegnato al destinatario se reso disponibile all'indirizzo elettronico da questi dichiarato, nella casella di posta elettronica del destinatario messa a disposizione dal gestore. La trasmissione telematica di comunicazioni che necessitano di una ricevuta di invio e di una ricevuta di consegna avviene mediante la posta elettronica certificata ai sensi del decreto del Presidente della Repubblica 11 febbraio 2005, n. 68; la trasmissione del documento informatico per via telematica, effettuata nei casi consentiti dalla legge in ottemperanza a tale normativa, equivale alla notificazione per mezzo della posta.

La garanzia dell'invio e della ricezione di documenti elettronici tra Pubbliche Amministrazioni, cittadini e imprese si avvia quindi a diventare realtà: con il DPR n.68/05 l'e-mail certificata acquista infatti valore legale, grazie al fatto che la trasmissione del messaggio e la ricezione da parte del destinatario sono attestate dai Gestori di posta elettronica certificata. Tale decreto (*Regolamento recante le disposizioni per l'utilizzo della posta elettronica certificata*) stabilisce le procedure tecniche per la certificazione dell'avvenuto invio e consegna di posta tramite e-mail.

La PEC, quindi, consente l'invio di messaggi la cui trasmissione è valida agli effetti di legge; a

1 Le innovazioni del codice dell'amministrazione digitale sono stati oggetto di trattazione nella presente rubrica del numero di dicembre 2005 di Mondo Digitale.

tal fine, dev'essere utilizzato l'indirizzo espressamente dichiarato dai cittadini ai fini di ciascun procedimento con le pubbliche amministrazioni o di ogni singolo rapporto intrattenuto tra privati e pubbliche amministrazioni. Le imprese, nei rapporti tra loro intercorrenti, possono dichiarare l'esplicita volontà di accettare l'invio di PEC mediante indicazione nell'atto di iscrizione al registro delle imprese.
L'Utilizzo della PEC, al posto della tradizionale posta mediante raccomandata con ricevuta di ritorno cartacea, è reso possibile attraverso le ricevute elettroniche rilasciate dai gestori di posta che sono da questi sottoscritte mediante una firma elettronica avanzata generata in automatico dal sistema di posta e basata su chiavi asimmetriche a coppia, una pubblica e l'altra privata, assicurando così la provenienza, l'integrità e l'autenticità del messaggio di PEC[2].
L'elenco pubblico dei gestori di Posta Elettronica Certificata, previsto dall'art. 14 del DPR 11 febbraio 2005, n. 68 tenuto dal CNIPA (Centro nazionale per l'informatica nella Pubblica Amministrazione) è reso disponibile attraverso la rete Internet[3]. I gestori di PEC, per essere inclusi nell'elenco pubblico gestito e controllato dal CNIPA, devono possedere determinati requisiti sia di adeguatezza del personale, di sicurezza e di esperienza nell'erogazione di servizi analoghi, che per quanto riguarda la natura giuridica della società e il suo capitale sociale. Sono inoltre tenuti a garantire la riservatezza, la sicurezza (anche per quanto concerne i virus informatici) e l'integrità nel tempo delle informazioni di certificazione conservandole per trenta mesi.

2. Il funzionamento della PEC

Il processo di invio e ricezione del messaggio di PEC è il seguente: il mittente spedisce il messaggio al suo gestore di PEC (punto di accesso) che gli inoltra la ricevuta di accettazione e contemporaneamente invia il messaggio al gestore del destinatario (punto di destinazione) assicurando l'interoperabilità dei servizi offerti.
Come per una raccomandata con ricevuta di ritorno un'e-mail certificata si ritiene "ricevuta" dal destinatario se consegnata nella casella di posta elettronica (un tanto è comprovato dalla ricevuta che invia il gestore di posta di quest'ultimo) indipendentemente dal fatto che sia stato o meno letta.
Nella figura 1 viene mostrato il flusso informatico del processo di invio e ricezione di un massaggio di PEC includente le ricevute previste dal Decreto Ministeriale, contenente le "Regole tecniche per la formazione, la trasmissione e la validazione, anche temporale, della PEC" (tutti i requisiti tecnico-funzionali che devono essere rispettati dalle piattaforme utilizzate per erogare il servizio), pubblicato nella *Gazzetta Ufficiale* del 15 novembre 2005, n. 266.
La PEC, quindi, è un sistema di posta elettronica nel quale è fornita al mittente documentazione elettronica, con valenza legale, attestante l'invio e la consegna di documenti informatici. "Certificare" l'invio e la ricezione (i due momenti fondamentali nella trasmissione dei documenti informatici) significa fornire al mittente, da parte del proprio gestore di posta, una ricevuta che attesta l'avvenuta spedizione del messaggio e dell'eventuale allegata documentazione. Allo stesso modo, quando il messaggio perviene al destinatario, il gestore invia al mittente la ricevuta di avvenuta (o mancata) consegna con precisa indicazione temporale. Se il mittente smarrisce le ricevute, la traccia informatica delle operazioni svolte, conservata per legge per un periodo di 30 mesi, consente la riproduzione, con lo stesso valore giuridico, delle ricevute stesse.
La valenza legale di tutte le operazioni è garantita dall'utilizzo delle firme elettroniche: su tutte le tipologie di messaggi PEC ed in particolare sulle buste di trasporto e sulle ricevute vengono apposte in modo automatico delle firme digitali da parte dei gestori al fine di assicurare l'integrità e l'autenticità del messaggio.
Il CNIPA ha i compiti di effettuare le attività di vigilanza e controllo assegnategli dalla norma e, con un apposito Centro di competenza, di supportare le PA ai fini dell'introduzione della PEC nei procedimenti amministrativi.

2.1. Formato dei messaggi generati dal sistema

Il sistema di PEC genera i messaggi (ricevute, avvisi e buste) in formato MIME. I messaggi sono composti da una parte di testo descrittivo, per l'utente, e da una serie di allegati (messag-

[2] La tematica della firma elettronica stata trattata nella presente rubrica del numero di marzo 2005 di Mondo Digitale.

[3] Si veda in www.cnipa.gov.it l'elenco pubblico dei gestori di Posta Elettronica Certificata.

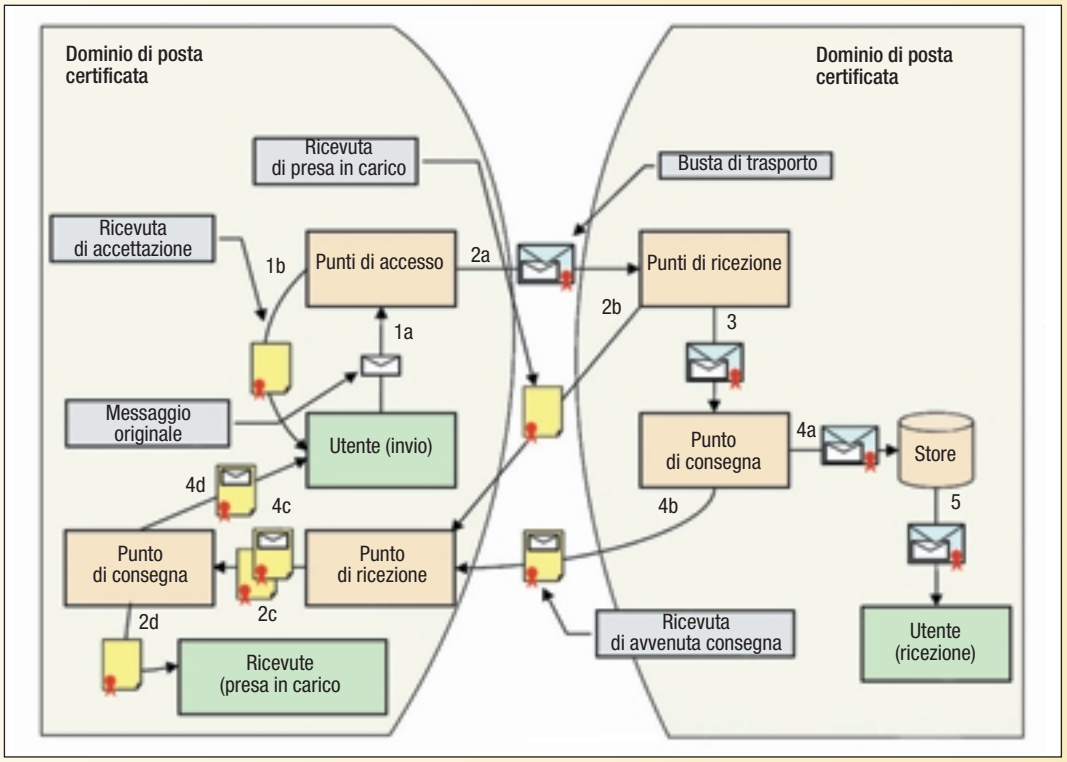


FIGURA 1

Busta di trasporto corretta e valida con consegna avente esito positivo:

1a - L'utente invia una e-mail al destinatario attraverso il Punto di Accesso del Gestore; 1b - Viene restituita all'utente una Ricevuta di Accettazione; 2a - Viene inoltrata la Busta di Trasporto e inoltrata al Punto di Ricezione del Gestore destinatario; 2b - Il Punto di Ricezione genera la Ricevuta di Presa in Carico che viene inoltrata al Punto di Ricezione del Gestore mittente; 2c - Il Punto di Ricezione del mittente inoltra la Ricevuta di Presa in Carico al Punto di Consegna: 2d - Il Punto di Consegna salva la Ricevuta di Presa in Carico nello store delle ricevute del Gestore; 3 - Il Punto di Ricezione inoltra la Busta di Trasporto al Punto di Consegna; 4a - Il Punto di Consegna inserisce la Busta di Trasporto nella mailbox del destinatario; 4b - Viene inviata una ricevuta di Avvenuta Consegna al Punto di Ricezione del Gestore mittente; 4c - Il Punto di Ricezione inoltra la Ricevuta al Punto di Consegna; 4d - Il Punto di Consegna salva la Ricevuta di Avvenuta Consegna nella mailbox del mittente; 5 - L'utente destinatario ha a disposizione la e-mail inviata da mittente

gio originale, dati di certificazione ecc.) variabili a seconda della tipologia del messaggio.
Il messaggio (composto dall'insieme delle parti descritte nelle specifiche sezioni del presente allegato) è inserito in una struttura S/MIME v3, firmata con la chiave privata del gestore di posta certificata. Il certificato associato alla chiave usata per la firma è incluso in tale struttura e il formato S/MIME usato per la firma dei messaggi generati dal sistema è il "multipart/signed" (formato .p7s).
Tutti i messaggi generati dal sistema di posta certificata sono identificabili per la presenza di un header specifico.
Durante le fasi di trattamento del messaggio presso i punti di accesso, ricezione e consegna, il sistema mantiene traccia delle operazioni svolte. Tutte le attività sono memorizzate su un registro riportante i dati significativi dell'operazione:

- il codice identificativo univoco assegnato al messaggio originale (Message-ID);
- la data e l'ora dell'evento;
- il mittente del messaggio originale;
- i destinatari del messaggio originale;
- l'oggetto del messaggio originale;
- il tipo di evento (accettazione, ricezione, consegna, emissione ricevute, errore ecc.);
- il codice identificativo (Message-ID) dei messaggi correlati generati (ricevute, errori ecc.);
- il gestore mittente.

Viene garantita la possibilità di reperire, a richiesta, le informazioni contenute nei log.

2.2. Punto di accesso

Il punto di accesso consente all'utente di accedere ai servizi di posta certificata resi disponi-

bili dal proprio gestore. La possibilità da parte di un utente di accedere ai servizi di PEC prevede necessariamente l'autenticazione dello stesso da parte al sistema. Alla ricezione di un messaggio originale, il punto di accesso:

- effettua dei controlli formali sul messaggio in ingresso;
- genera una ricevuta di accettazione;
- imbusta il messaggio originale in una busta di trasporto.

La ricevuta di accettazione indica al mittente che il suo messaggio è stato accettato dal sistema e certifica la data e l'ora dell'evento. All'interno della ricevuta è presente un testo leggibile dall'utente, un allegato XML con i dati di certificazione in formato elaborabile ed eventuali altri allegati per funzionalità aggiuntive offerte dal gestore.
Il punto di accesso, utilizzando i dati dell'indice dei gestori di posta certificata, effettua un controllo per ogni destinatario del messaggio originale per verificare se appartengono all'infrastruttura di posta certificata o sono utenti esterni (esempio, posta Internet). Tale controllo è realizzato verificando l'esistenza dei domini dei destinatari presenti all'interno dell'indice dei gestori. La ricevuta di accettazione (ed i relativi dati di certificazione) riporta quindi la tipologia dei vari destinatari per informare il mittente del differente flusso seguito dai due gruppi di messaggi (utenti di posta certificata, utenti esterni).

2.3. Controlli formali sui messaggi in ingresso

Al momento dell'accettazione del messaggio il punto di accesso deve garantirne la correttezza formale effettuando delle verifiche sui campi ("From","To"...) o sui dati di instradamento.
Qualora il messaggio non superi i controlli, il punto di accesso non dovrà accettare il messaggio all'interno del sistema di posta certificata emettendo il relativo avviso di non accettazione.
Qualora il punto di accesso non possa provvedere all'inoltro del messaggio, a causa del mancato superamento dei controlli formali, viene recapitato al mittente uno specifico avviso di non accettazione. Il corpo del messaggio di questa ricevuta è composto da un testo che costituisce la vera e propria ricevuta in formato leggibile.

2.4. Busta di trasporto

La busta di trasporto consiste in un messaggio generato dal punto di accesso e contiene il messaggio originale ed i dati di certificazione.
Il corpo della busta di trasporto è composto da un testo che costituisce la parte immediatamente leggibile dal destinatario del messaggio di posta.
All'interno della busta di trasporto è inserito in allegato l'intero messaggio originale immodificato in formato conforme alla RFC 2822 completo di header, corpo ed eventuali allegati. Nella stessa busta di trasporto è inoltre incluso un allegato XML che specifica i dati di certificazione già riportati nel testo ed informazioni aggiuntive sul tipo di messaggio e tipo di ricevuta richiesta.

2.5. Punto di ricezione

Il punto di ricezione permette lo scambio di messaggi di posta certificata tra diversi gestori di posta certificata, ed è anche il punto attraverso il quale messaggi di posta elettronica ordinaria possono essere inseriti nel circuito della posta certificata.
Lo scambio di messaggi tra diversi gestori avviene tramite una transazione basata sul protocollo SMTP come definito dalla RFC 2821.
Il punto di ricezione, a fronte dell'arrivo di un messaggio, effettua una serie di controlli ed operazioni quali la verifica della correttezza/natura del messaggio in ingresso, l'integrità e la correttezza delle busta di trasporto o delle ricevute. La ricevuta di presa in carico è emessa dal gestore ricevente il messaggio, nei confronti del gestore mittente. Il suo fine è quello di consentire il tracciamento del messaggio nel passaggio tra un gestore ed un altro.

2.6. Ricevuta di presa in carico

Allo scambio di messaggi di posta certificata corretti tra differenti gestori di posta certificata, il gestore ricevente emette una ricevuta di presa in carico nei confronti del gestore mittente. Le ricevute di presa in carico emesse sono relative ai destinatari ai quali è indirizzato il messaggio in ingresso. All'interno dei dati di certificazione della singola ricevuta di presa in carico sono elencati i destinatari a cui la stessa fa riferimento.
L'indirizzo per l'invio delle ricevute al gestore mittente è ricavato dall'indice dei gestori di posta certificata durante l'interrogazione necessaria per il controllo del soggetto che ha emesso la firma nella verifica del messaggio in ingresso.

2.7. Busta di anomalia

Quando un messaggio evidenzia un errore o viene riconosciuto come messaggio non di posta elettronica certificata, il gestore effettua la propagazione verso il destinatario utilizzando un'apposita busta di anomalia per evidenziare al destinatario detta anomalia.

2.8. Avvisi relativi alla rilevazione di virus informatici

Sia il punto di accesso che quello di destinazione effettuano controlli relativamente alla presenza di virus informatici. Qualora il gestore del mittente (destinatario) riceva messaggi con virus informatici è tenuto a non accettarli, informando tempestivamente il mittente (gestore del mittente) dell'impossibilità di dar corso alla trasmissione ed emettendo un chiaro *avviso di non accettazione (mancata consegna) per virus informatico*.

2.9. Punto di consegna

All'arrivo del messaggio presso il punto di consegna, il sistema del gestore di posta elettronica certificata ne verifica la tipologia e stabilisce se deve inviare una ricevuta al mittente. La ricevuta di avvenuta consegna è emessa dopo che il messaggio è stato consegnato nella casella di posta del destinatario. La ricevuta di avvenuta consegna indica al mittente che il suo messaggio è stato effettivamente consegnato al destinatario specificato e certifica la data e l'ora dell'evento tramite un testo leggibile dall'utente ed un allegato XML con i dati di certificazione. Infatti le ricevute di avvenuta consegna sono costituite da un messaggio di posta elettronica inviato al mittente che riporta la data e l'ora di avvenuta consegna, i dati del mittente e del destinatario e l'oggetto. Se il messaggio pervenuto al punto di consegna non fosse recapitabile alla casella di destinazione, il punto di consegna emette un avviso di mancata consegna (Figura 2). Nel caso si verifichi un errore nella fase di consegna del messaggio, il sistema genera un avviso di mancata consegna da restituire al mittente con l'indicazione dell'errore riscontrato.

2.10. La provenienza PEC

La posta elettronica certificata garantisce la provenienza del messaggio attraverso l'assicurazione dell'inalterabilità dell'indirizzo associato alla casella di posta elettronica certificata dalla quale si effettua l'invio del messaggio; proprio questa particolarità del servizio

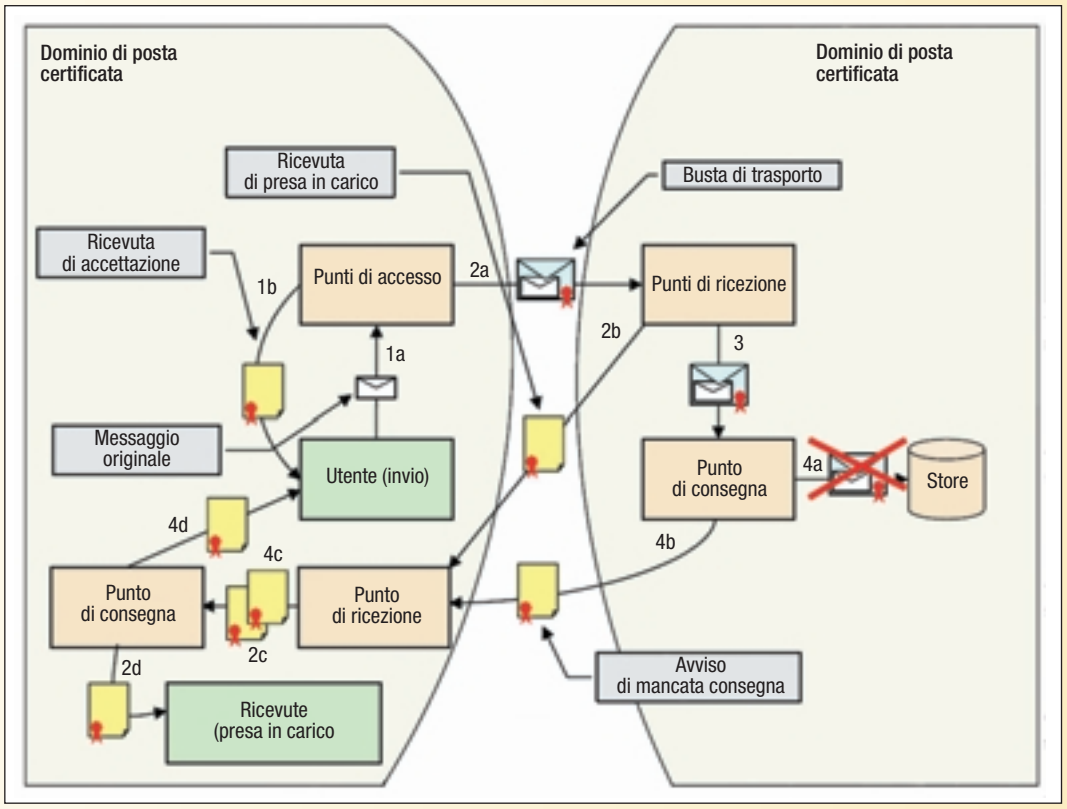


FIGURA 2
Busta di trasporto corretta e valida con consegna avente errore di consegna

PEC, risulta essere un valido deterrente contro il fenomeno dello SPAM. Inoltre vi è la garanzia dell'identità del mittente attraverso l'associazione fra il titolare del servizio e la relativa casella di posta elettronica certificata in quanto il soggetto che intende richiedere un servizio di PEC deve presentare al Gestore (iscritto nell'indice tenuto dal CNIPA), oltre alla richiesta di attivazione del servizio, anche un documento che attesti la sua identità diventando quindi titolare del servizio.

3. Conclusioni

Con lo stesso valore legale della tradizionale raccomandata con avviso di ricevimento, il servizio di PEC consente di effettuare l'invio di documenti informatici avendo la garanzia di "certificazione" dell'invio e dell'avvenuta (o mancata) consegna (scambiato tra il Gestore di PEC del mittente e quello del destinatario), della provenienza, della data e l'ora di ricezione e spedizione, l'integrità del contenuto del messaggio e degli allegati. Il servizio ha, pertanto, tutti i requisiti della raccomandata con A/R cui si aggiungono notevoli vantaggi sia in termini di tempo che di costi:

- semplicità ed economicità di trasmissione, inoltro e riproduzione;
- semplicità ed economicità di archiviazione e ricerca;
- facilità di invio multiplo, cioè a più destinatari contemporaneamente, con costi estremamente più bassi rispetto a quelli dei mezzi tradizionali;
- velocità della comunicazione in quanto non è necessaria la presenza del destinatario per completare la consegna;
- possibilità di consultazione ed uso anche da postazioni diverse da quella del proprio ufficio o abitazione (basta un qualsiasi PC connesso ad Internet e un normale browser web), ed in qualunque momento grazie alla persistenza del messaggio nella casella di posta elettronica;
- diversamente dalla raccomandata, nella ricevuta di avvenuta consegna sono presenti anche i contenuti del messaggio originale.

Il Ministero per l'innovazione e le tecnologie ha stimato 31 milioni i messaggi di posta elettronica inviati tra P.A. e soggetti esterni, 18 € il risparmio ottenuto per ogni e-mail rispetto alla lettera cartacea che, secondo il Ministero, potranno portare risparmi di circa 360 milioni di euro già da quest'anno. Pertanto, il riconoscimento giuridico delle comunicazioni per via telematica anche attraverso la PEC, oltre ad ottemperare il diritto, per i cittadini e le imprese, all'uso delle tecnologie ed a ricevere qualsiasi comunicazione pubblica per e-mail, come previsto nel Codice dell'amministrazione Digitale, pone le basi affinché la P.A. digitale funzioni meglio garantendo efficacia ed efficienza.

Bibliografia

[1] Codice dell'amministrazione digitale (D.lgs. 82/2005).

[2] Decreto del Presidente della Repubblica 11 febbraio 2005, n. 68.

[3] Decreto Ministeriale 2 novembre 2005 - Ministro per l'Innovazione e le Tecnologie.

[4] www.innovazione.gov.it

[5] www.cnipa.gov.it

Antonio Piva laureato in Scienze dell'Informazione, Vice Presidente dell' ALSI (Associazione Nazionale Laureati in Scienze dell'Informazione ed Informatica) e Presidente della commissione di informatica giuridica.
Docente a contratto di diritto dell'informatica all'Università di Udine.
Consulente sistemi informatici e Governo Elettronico nella PA locale, valutatore di sistemi di qualità ISO9000 ed ispettore AICA ECDL.
E-mail: antonio_piva@libero.it

David D'Agostini avvocato, ha conseguito il master in informatica giuridica e diritto delle nuove tecnologie, fornisce consulenza e assistenza giudiziale e stragiudiziale in materia di *software, privacy* e sicurezza, contratti informatici, *e-commerce*, nomi a dominio, computer crimes, firma digitale. Ha rapporti di partnership con società del settore ITC nel Triveneto.
Collabora all'attività di ricerca scientifica dell'Università di Udine e di associazioni culturali.
E-mail: david.dagostini@adriacom.it

Maurizio Blancuzzi laureato in Scienze dell'Informazione. Responsabile e-government Regione autonoma FVG. Membro della commissione informatica giuridica dell'ALSI (Associazione Nazionale Laureati in Scienze dell'Informazione e Informatica). Ha svolto anche attività didattica e di consulenza in materia di ICT ed informatica giuridica.
E-mail: maublanc@katamail.com

DENTRO LA SCATOLA

Rubrica a cura di

Fabio A. Schreiber

Dopo aver affrontato negli scorsi anni due argomenti fondanti dell'Informatica – il modo di codificare l'informazione digitale e la concreta possibilità di risolvere problemi mediante gli elaboratori elettronici – con questa terza serie andiamo ad esplorare "*Come parlano i calcolatori*". La teoria dei linguaggi e la creazione di linguaggi di programmazione hanno accompagnato di pari passo l'evolversi delle architetture di calcolo e di gestione dei dati, permettendo lo sviluppo di applicazioni sempre più complesse, svincolando il programmatore dall'architettura dei sistemi e consentendogli quindi di concentrarsi sull'essenza del problema da risolvere.
Lo sviluppo dell'Informatica distribuita ha comportato la nascita, accanto ai linguaggi per l'interazione tra programmatore e calcolatore, anche di linguaggi per far parlare i calcolatori tra di loro – i protocolli di comunicazione. Inoltre, la necessità di garantire la sicurezza e la privatezza delle comunicazioni, ha spinto allo sviluppo di tecniche per "non farsi capire" da terzi, di quì l'applicazione diffusa della crittografia.
Di questo e di altro parleranno le monografie quest'anno, come sempre affidate alla penna (dovrei dire tastiera!) di autori che uniscono una grande autorevolezza scientifica e professionale ad una notevole capacità divulgativa.

La compilazione: concetti e sviluppi tecnologici

Stefano Crespi Reghizzi

1. La compilazione

Quali sono i programmi più eseguiti e meno conosciuti dall'utente? Con buona probabilità la risposta è: i compilatori. Introdotto all'alba dell'informatica, il termine "compilazione" sta per la traduzione d'un testo *sorgente*, scritto in linguaggio artificiale, in un testo *pozzo*[1], scritto in un altro linguaggio.
Cercheremo di descrivere i compilatori o traduttori nei loro aspetti concettuali, tecnici e applicativi, e nell'evoluzione concomitante a quella delle architetture di calcolo e della Rete.
I linguaggi progettati nei cinquant'anni passati sono migliaia, quelli tuttora utilizzati poche decine. I linguaggi, a seconda della destinazione, si classificano in: programmazione degli elaboratori (per esempio, C++), interrogazione di basi dati (per esempio, SQL), descrizione dei documenti (per esempio, XML), progetto dei sistemi digitali (per esempio, VHDL), controllo di robot ecc..
Poiché sono i linguaggi programmativi quelli che stanno al centro della ricerca sulle tecniche di compilazione, di cui beneficiano anche le altre categorie di linguaggi, non è limitativo parlare della compilazione dei soli linguaggi di programmazione.
Con un balzo indietro di 50 anni, immaginiamo la pena dei programmatori che scrivevano le applicazioni nel linguaggio assemblatore, scegliendo i codici operativi, i registri e gli indirizzi dei dati. All'ingrandirsi degli applicativi, le difficoltà crebbero, e, ad aggravare la situazione, per ogni nuovo elaboratore, il lavoro era completamente da rifare: mancava totalmente la *portabilità* dei programmi.
È storia nota l'invenzione del FORTRAN, il primo progetto industriale d'un linguaggio e del suo compilatore. Altrettanto nota, credo, è la straordinaria sinergia che si ebbe negli anni 1950-60 tra la linguistica, la nascente teoria degli automi e dei linguaggi formali, e l'ingegneria dei compilatori.
Il buon esito del compilatore FORTRAN della

1 La denominazione "testo pozzo" qui usata sembra più appropriata di quella tradizionale "testo oggetto": infatti il flusso della traduzione corre, come un corso d'acqua, dalla sorgente al pozzo.

IBM, fece fare un balzo alle applicazioni scientifiche, che poterono essere scritte in un linguaggio di alto livello, e mostrò la via a altri linguaggi importanti, come Algol 60, COBOL e Lisp.

2. Struttura del compilatore

La struttura di un compilatore degli anni 1960 è ancora attuale, al livello delle parti principali. Vi sono più passate, ognuna elabora un testo (o meglio una *rappresentazione intermedia* IR) e produce un'altro testo per la passata successiva. La prima passata legge il *testo sorgente*, scritto nel linguaggio da tradurre; l'ultima produce il *testo pozzo*, nel linguaggio della macchina.
Ogni passata, tranne l'ultima, produce dunque una IR, che è l'ingresso della passata successiva. La granularità delle IR cambia da una passata all'altra, dai costrutti del linguaggio sorgente fino alle istruzioni macchina.
In questo schema, l'ordinamento tipico delle passate è il seguente:

- il *parsificatore* (o *analizzatore lessicale-sintattico*) verifica che il testo sorgente sia lessicalmente e sintatticamente corretto, e lo trasforma in una IR, che agevola le passate successive. La IR tipica è l'*albero sintattico astratto*, AST, che mostra le inclusioni testuali tra i costrutti del programma sorgente. Gli elementi atomici della AST sono gli identificatori, le funzioni, gli operatori, le costanti ecc., ossia i *lessemi* del linguaggio.
- L'*analizzatore semantico* controlla sulla IR del programma che le regole semantiche del linguaggio sorgente siano rispettate: per esempio, le regole di conformità di tipo tra gli operandi delle espressioni aritmetiche.
- Il *generatore del codice* sceglie le istruzioni macchina.
- *Ottimizzatori*. Altre passate, poste a monte o a valle del generatore di codice, sono necessarie per migliorare l'efficienza del codice prodotto. Le ottimizzazioni sono la parte più complessa e qualificante del compilatore.

Tale schema viene denominato *traduttore guidato dalla sintassi* (SDT[2]), poiché la prima attività è l'analisi sintattica, che si basa sulla grammatica formale (o sintassi) del linguaggio sorgente. Per meglio comprendere, conviene guardare da vicino l'organizzazione dei primi due stadi.

2.1. Teorie formali nel progetto del compilatore

La decomposizione in passate permette di dominare la complessità del progetto e di sfruttare in ciascuna passata certi metodi specializzati, che ora saranno tratteggiati.
Dall'invenzione del FORTRAN, tutti i linguaggi sono stati definiti mediante una *grammatica* formale, cioè mediante delle regole sintattiche quali per esempio:

frase_condizionale → **if** espressione_booleana **then** istruzione **else** istruzione

Le regole sono dette libere dal contesto (*context-free o type 2*) dal linguista Noam Chomsky, e BNF (*Backus Naur Form*) dagli informatici del progetto Algol 60. I termini come frase_condizionale sono le classi sintattiche; la classe programma comprende tutti i testi sorgente che obbediscono alla grammatica.
Sotto la sintassi sta il lessico. Come in italiano un periodo è una serie di parole, così un programma è una sequenza di lessemi (costanti, identificatori, operatori, commenti, parole chiave come **if** ecc.).
La grammatica comprende al livello inferiore le regole lessicali, esse stesse scritte sotto forma di regole libere dal contesto o più spesso di espressioni regolari. Per esempio una costante intera è definita dall'espressione regolare 0 | (1 ... 9)(0 ... 9) *.
La prima operazione del parsificatore è l'analisi lessicale (*scanning*), che segmenta il testo sorgente nei lessemi. Essa opera tramite dei semplici automi a stati finiti che scandiscono il file sorgente e riconoscono le classi lessicali.
Successivamente il parsificatore esamina il programma, ora rappresentato come stringa di lessemi, per verificare le regole sintattiche. L'algoritmo è un automa che, oltre agli stati interni, possiede una memoria ausiliaria con modalità di accesso LIFO (ossia una pila).
Gli algoritmi lessicali e sintattici, studiati dagli anni 1960, sono efficienti: il tempo di calcolo è lineare rispetto alla lunghezza del programma, per gli algoritmi più diffusi, gli analizzatori ascendenti a spostamento e riduzione e quelli

[2] Syntax directed translator.

discendenti del tipo predittivo, che realizzano un automa deterministico. I due casi si differenziano rispetto all'ordine in cui l'albero sintattico è costruito: rispettivamente in ordine ascendente dai lessemi alla classe sintattica programma, o nell'ordine inverso, dalla radice alle foglie dell'albero, detto discendente.
In pratica il parsificatore deve anche costruire l'albero sintattico astratto AST del programma e dare indicazioni diagnostiche in caso di errore.
Certo l'esito della parsificazione è soltanto il primo esame, perché un programma, pur se sintatticamente corretto, può violare la semantica del linguaggio. Anche la distinzione tra correttezza sintattica e semantica, come quella tra lessico e sintassi, è un retaggio della prima teoria di Chomsky.
Più precisamente gli errori presenti in un programma sono classificabili come lessicali, sintattici e semantici. Un'altra classificazione ortogonale distingue gli errori *statici* da quelli *dinamici*, dove l'aggettivo "statico" indica un'attività svolta durante la traduzione, mentre "dinamico" si riferisce a quanto sarà fatto durante l'esecuzione del programma. Per esempio, l'assegnamento d'una costante del tipo stringa a una variabile del tipo integer è un errore semantico statico; l'uso di una variabile, come indice d'un vettore, la quale fuoriesca dalle dimensioni del vettore stesso, è un errore semantico, ma dinamico.
La seconda passata verifica la correttezza semantica statica del programma. Le regole di tipizzazione e di visibilità ("scope") sono quelle proprie del linguaggio sorgente. L'analizzatore o valutatore semantico è un algoritmo ricorsivo che visita lo AST, controllando, per ogni costrutto (per esempio, un assegnamento) l'osservanza delle regole semantiche.
A differenza di quelle sintattiche, le regole semantiche non sono generalmente coperte da un modello formale, anche se non sono mancati i tentativi di formalizzarle.
Nei compilatori dei linguaggi correnti, la semantica è talvolta specificata in qualche notazione semiformale, come le grammatiche con attributi (proposte da D. Knuth nel 1968).
Superato il controllo semantico, il testo è tradotto in una IR, che quasi sempre include la tabella dei simboli, concettualmente simile a una base dati che descrive le proprietà di tutti gli enti (variabili, costanti, procedure o metodi, classi ecc.) presenti nel programma. Ma non esiste uno standard, e ogni compilatore adotta la propria IR.

3. Fronte e retro

Gli analizzatori sintattico e semantico non dipendono dall'architettura del processore, ma soltanto dalle specifiche del linguaggio sorgente. Tale parte del compilatore è detta *fronte* o *tronco* comune. I metodi di progetto del tronco sono assestati e descritti in molti testi sui compilatori (per esempio, [1, 2]).
Sul tronco si innestano, come le branche d'un albero, altre passate dipendenti dalla macchina, note come *retro* (back-end), tra le quali spicca il generatore di codice.
La divisione tra fronte e retro facilita il progetto del compilatore, sia separando gli ambiti di competenza dei progettisti, sia rendendo possibile il riuso di una o dell'altra parte. Così il tronco d'un compilatore per il C potrà essere riutilizzato, sostituendo il generatore per la macchina A con quello per la macchina B. Viceversa due tronchi per il linguaggio C e C++ possono stare a monte dello stesso generatore di codice. Naturalmente, affinché il riuso sia possibile, le interfacce IR devono essere unificate. Ciò di norma avviene all'interno d'una piattaforma di compilazione, progettata da un'industria o organizzazione, per una gamma di linguaggi sorgente e di architetture; mentre l'interoperatività di compilatori di diversa origine è ostacolata dall'assenza di standard.

4. Meta-compilazione

Ben presto infastidì i progettisti la ripetizione del progetto del parsificatore, al mutare del linguaggio sorgente: di qui l'idea di produrre il parsificatore con un programma generatore, parametrizzato con le regole lessicali e sintattiche del linguaggio sorgente. Il generatore legge le regole (i meta-dati) e produce il programma del parsificatore specifico. Molti compilatori sono stati così prodotti, grazie alla disponibilità di buoni strumenti, dal classico *lex/yacc* (*flex/bison* nella versione libera di GNU) che produce un parsificatore ascendente in C, al più recente ANTLR, che genera un parsificatore a discesa ricorsiva in Java.
Perché non generare automaticamente anche l'analizzatore semantico, che è assai più complesso del parsificatore? L'idea ebbe comprensibilmente molti fautori, e alcuni strumenti (parame-

trizzati da una specifica semiformale delle azioni semantiche nota come *grammatiche con attributi*) ottennero qualche diffusione, per poi finire in disarmo. Due fattori giocano a sfavore dei metacompilatori semantici: la mancanza di una teoria formalizzata, semplice e condivisa; e il fatto che le tecniche di programmazione orientate agli oggetti già permettono un buon riuso di quella parte del compilatore che tratta le proprietà delle entità dichiarate nel programma sorgente.
L'uso di librerie di classi parametriche rispetto al linguaggio sorgente o all'architettura pozzo è frequente nelle piattaforme di compilazione sulle quali ritorneremo più avanti.

5. Generazione del codice

La più ovvia dipendenza dalla macchina viene dall'architettura dell'insieme di istruzioni (ISA). Fino agli anni 1980 il generatore di codice era scritto a mano da uno specialista del processore, abile nello sfruttare ogni istruzione. Ma la crescente complessità delle istruzioni delle macchine CISC[3] e la difficile manutenzione del generatore, stimolarono le ricerche sulla produzione automatica dei generatori. Ebbero successo i *generatori di generatori di codice*, impostati mediante la ricerca di forme (vedasi per esempio, [3]) sull'albero della IR. In breve, ogni istruzione macchina (per esempio, un'addizione tra un registro e una cella di memoria) è descritta formalmente come un *pattern*, ossia un frammento. La IR del programma deve essere ricoperta (tiling) con le forme corrispondenti ai pattern della macchina. Poiché molte ricoperture sono possibili, l'algoritmo deve orientare la scelta con opportune euristiche verso la copertura di costo minimo, dove il costo è il tempo di esecuzione del codice macchina.
Occorre dire che la ricerca della ricopertura ottimale dell'albero IR risulta molto più facile se l'architettura è del tipo RISC[4], nel qual caso è agevole scrivere a mano il generatore di codice.

6. Analisi statica di flusso

Per rendere eseguibile il codice, manca ancora un aspetto essenziale: la scelta dei registri del processore; infatti il passo precedente, per non affrontare in un colpo solo tutte le difficoltà, ha generato delle istruzioni che usano un numero illimitato di registri. Ma un cattivo uso dei registri provoca inutili copiature di dati, nonché rallentamenti dovuti agli accessi alla memoria. I processori moderni dispongono di tanti registri, che se ben sfruttati permettono di limitare gli accessi alla memoria.
L'assegnazione dei registri alle variabili è un problema di conflitto nell'uso di risorse scarse; poiché la ricerca della soluzione ottima è complessa, si deve ricorrere a metodi euristici.
Due variabili possono stare nello stesso registro se, in nessun punto del programma, entrambi i valori sono necessari: si dice che gli *intervalli di vita* delle due variabili sono disgiunti. Per assegnare i registri, è necessario calcolare gli intervalli di vita delle variabili. Il programma, ormai in codice macchina, è rappresentato dal suo *grafo di controllo* (CFG[5]), un'astrazione in cui ci si limita a guardare quali variabili siano lette o calcolate da ogni istruzione. Gli insiemi delle variabili vive in ogni punto del CFG sono calcolati risolvendo con metodo iterativo certe *equazioni di flusso*, facilmente ricavabili, istruzione per istruzione. La teoria delle equazioni di flusso (presente nei riferimenti [1, 2, 3, 4]) si fonda sull'algebra dei semi-anelli commutativi e trova applicazioni, non solo nella compilazione, ma anche nell'analisi e nella verifica dei programmi.
Risolte le equazioni, il compilatore, impiegando un metodo euristico, assegna registri diversi alle variabili che hanno intervalli di vita sovrapposti, senza superare per altro il numero di registri disponibili. In caso d'impossibilità, esso genera a malincuore del codice (spillatura) per copiare in memoria e poi riprendere i valori di certe variabili.

7. Ottimizzazioni: un catalogo senza fine

Ora il codice macchina è eseguibile, ma le sue prestazioni sarebbero inaccettabili, se non venisse migliorato con tante astute trasformazioni. L'ottimizzazione è la parte più impegnativa del compilatore, la cui complessità cresce con quella dei processori.
Conviene distinguere le trasformazioni indipen-

[3] Complex instruction set computer.

[4] Reduced instruction set computer.

[5] Control flow graph.

denti dalla macchina dalle altre. Le prime sono efficaci su ogni architettura. Si pensi al calcolo d'una espressione X + Y · Z che compare due volte nel programma, senza che le variabili cambino valore. Il compilatore salva in un registro il risultato del primo calcolo, e sostituisce il secondo con la copiatura del registro.
Un catalogo ragionato delle numerosissime trasformazioni esistenti è in Muchnick [4]. Una questione delicata è l'ordine in cui eseguire le trasformazioni; infatti una modifica, apparentemente non migliorativa, può abilitare altri miglioramenti.
Per esemplificare le ottimizzazioni dipendenti dalla macchina, possiamo soffermarci su due importanti aspetti architetturali.

8. Gerarchia di memoria

Per ridurre le attese (*latenze*) dovute alle operazioni sulla memoria RAM, le macchine usano memorie *cache*, più veloci ma molto più piccole. Un programma che con un ciclo iterativo spaziasse su troppe celle di memoria vanificherebbe l'efficacia della cache, che dovrebbe essere continuamente caricata e scaricata dalla memoria.
Concentrando l'attenzione sulle parti del programma eseguite più intensamente (i cosiddetti *nuclei* o *punti caldi*), l'ottimizzatore riorganizza i cicli in modo da operare su uno spazio di memoria contenibile nella cache. Inoltre esso inserisce anticipatamente nel codice le istruzioni di precaricamento (*prefetch*), che caricano la cache con i dati, in anticipo rispetto al momento in cui saranno richiesti.

9. Parallelismo schedulazione e predicazione

Le macchine offrono tante forme di parallelismo: pipeline, disponibilità di più unità funzionali, istruzioni composte da più codici operativi (architettura VLIW[6]), fino alle architetture multiprocessore.
Per sfruttare il pipeline, un buon codice deve rarefare le istruzioni di salto. Una trasformazione migliorativa consiste nell'ingrandire le dimensioni dei *blocchi basici* (una serie di istruzioni non interrotte da salti né da etichette), magari srotolando il corpo di un ciclo iterativo due o più volte.
Le architetture VLIW bene illustrano il problema della *schedulazione delle istruzioni*, ossia del loro riordino rispetto all'ordine originale, allo scopo di utilizzare al meglio in ogni istante di clock tutte le unità funzionali della macchina.
La schedulazione è semplice per un blocco basico, ma diventa intrattabile per l'intero programma. Il compilatore impiega una gamma di schedulatori, appropriati per diverse parti del programma: l'algoritmo di list scheduling (simile al job shop scheduling della ricerca operativa) si addice alla parti acicliche; l'elegante algoritmo di software pipelining e modulo scheduling si applica ai cicli più caldi del programma, l'algoritmo di trace scheduling privilegia la *traccia* del programma eseguita più spesso, anche al costo di duplicare del codice nelle altre parti.
Un altro meccanismo, spesso combinato con il parallelismo, è l'esecuzione *speculativa e predicativa* del codice. L'idea è che nessuna unità funzionale deve essere lasciata inerte. Se per esempio, un moltiplicatore fosse disponibile, ma la o le istruzioni in fase di esecuzione non lo richiedessero, sarebbe un peccato. Il compilatore, analizzando il codice, sceglie una futura e distante istruzione di moltiplicazione da far eseguire anticipatamente, pur senza la certezza che il flusso di controllo la raggiungerà.
Per confermare l'effetto dell'istruzione, quando si avrà la certezza della sua utilità, un bit o predicato, riceve il valore da un'apposita istruzione.
L'esecuzione predicativa non è indicata per dispositivi alimentati a batteria, perché consuma energia per eseguire operazioni talvolta inutili.

10. Elaborazione di flussi multi mediali

La diffusione dei dispositivi multi-mediali ha creato una nuova classe di coprocessori di flusso (stream), realizzati con architetture molto più veloci dei microprocessori, per tali elaborazioni. L'elaborazione consiste in un ciclo di istruzioni ripetute indefinitivamente sul flusso d'ingresso, per produrre quello d'uscita. La presenza di numerose unità funzionali inter-

[6] Very long instruction word.

connesse da una rete rende difficile la programmazione manuale e motiva lo sviluppo di compilatori capaci di estrarre da un programma generico le parti destinate al coprocessore di flusso e di tradurle nelle istruzioni del coprocessore stesso. La traduzione presenta difficili problemi di assegnazione ottimale delle istruzioni alle unità funzionali.
L'esempio dei processori di flusso è emblematico della compilazione per architetture speciali, un campo in sviluppo grazie alla flessibilità offerta dalle moderne tecniche di progetto dello hardware.
Ogni trasformazione ottimizzante dà un piccolo contributo (si spera positivo ma non sempre è così!) all'efficienza del codice macchina. Aggiungendo con abilità e tenacia qualche punto percentuale qua e là, si ottengono le buone prestazioni che qualificano i compilatori.

11. Interpretazione e compilazione dinamica

La Rete ha fatto emergere nuove esigenze, perché gli applicativi possono essere trasferiti da una macchina remota fornitrice a quella dell'utente, proprio al momento dell'esecuzione. Non essendo praticamente possibile trasferire i codici binari, perché la macchina utente è in generale ignota, la soluzione, affermatasi con il linguaggio Java, è l'*interpretazione*. L'applicativo, tradotto dal fornitore in un linguaggio intermedio o virtuale (ByteCode) indipendente dalla macchina, è spedito alla macchina destinataria, dove è eseguito da una *macchina virtuale* VM. In questo modo l'onere di adattare il linguaggio Java all'architettura è spostato dal mittente al destinatario dell'applicativo.
Ma la velocità dell'applicativo interpretato è almeno un ordine di grandezza inferiore, rispetto al codice compilato. Come conciliare l'indipendenza dalla macchina con l'efficienza? La risposta sta nella *compilazione dinamica* (anche detta JIT *just in time*). Accanto all'interprete, vi è un traduttore che converte le istruzioni virtuali in quelle del processore. Poiché, quando gira il traduttore, l'applicativo è fermo, e l'utente si spazientisce, occorre ridurre il tempo di compilazione. Tipicamente, l'esecuzione inizia mediante la macchina virtuale, poi parte il compilatore dinamico che traduce le parti calde, limitando le ottimizzazioni a quelle meno faticose. Per esempio l'assegnazione dei registri fisici è fatta con algoritmi più sbrigativi di quelli utilizzati da un compilatore statico. Occorre preliminarmente identificare le parti calde, per mezzo di tecniche di analisi statica e di conteggio dinamico (*profiling*) delle istruzioni eseguite.
Il compilatore dinamico deve essere programmato con estrema cura, per portare frutto. Una rassegna della compilazione dinamica è in [5].
Non sempre però la compilazione dinamica è possibile. I piccoli dispostivi, come i telefonini, non hanno sufficiente memoria per il compilatore dinamico. Per essi, la macchina virtuale resta l'unica possibilità, che deve sfruttare ogni possibile risparmio di memoria, senza per altro sacrificare la velocità.

12. Altri usi della compilazione dinamica

La compilazione dinamica ha altre applicazioni: la *traduzione da binario a binario* trasforma, al momento dell'esecuzione, il codice macchina della macchina A in quello della macchina B, curando anche la conversione delle chiamate di sistema operativo. Questa tecnica è molto utilizzata per emulare un'architettura ISA su di un'altra, per esempio, l'architettura Intel x86 sull'architettura TransMeta.
La *ottimizzazione continua* si applica sulle macchine, per esempio, i serventi delle basi dati, in ciclo continuo, con programmi in cui il profilo del carico da eseguire cambia, mettiamo, tra un giorno e il successivo.
Il codice macchina potrà essere ottimizzato in funzione del profilo attuale del carico. Per esempio, se una certa procedura in un certo giorno è invocata con un parametro fissato su un valore costante, il compilatore può *specializzare* il corpo della procedura, propagando il valore costante e eliminando eventuali test sul valore del parametro.
In un campo diverso, con dispositivi a batteria, si usa la compilazione dinamica per ridurre la potenza elettrica assorbita. È noto che la potenza cresce con il quadrato della frequenza del clock e che i nuovi processori dispongono di comandi per variare la frequenza di lavoro. Se l'applicativo in esecuzione ha sufficienti margini rispetto alle scadenze di tempo reale

da rispettare, la frequenza può essere rallentata, riducendo il consumo. È un monitore dinamico che valuta le opportunità e produce le istruzioni necessarie.

13. Infrastrutture aperte per la compilazione

Oggi chi deve sviluppare un compilatore ha la possibilità di partire dall'esistente e di modificarlo, e solo raramente si impostano da zero dei nuovi traduttori. Molti compilatori di proprietà di aziende sono l'evoluzione di progetti, anche molto vecchi, che via via si adeguano per le nuove architetture. Inoltre varie comunità hanno progettato e mantengono aggiornate le cosiddette infrastrutture aperte (o piattaforme) per la compilazione, veri progetti collettivi di componenti software per la costruzione dei compilatori.
La maggior parte delle piattaforme opera sui linguaggi più classici e diffusi: C, C++, Java e C#. Per inciso, nonostante la fioritura di tante proposte linguistiche, C e C++ restano la scelta preferita di quanti devono scrivere i compilatori e il software di sistema.
Altre comunità lavorano sui linguaggi funzionali e sui linguaggi interpretati di alto livello, come Python.
Limitandoci alle infrastrutture più diffuse, la più veneranda è *SUIF* della Stanford University, "una piattaforma libera progettata per aiutare la ricerca collaborativa sui compilatori ottimizzanti e parallelizzanti". È facile scrivere in C++ le proprie passate di compilazione appoggiandosi alle pratiche rappresentazioni IR di SUIF. Sono disponibili i fronti per C, C++, Fortran, e Java, e i retro per vari microprocessori. Una funzionalità, forse sorprendente ma utilissima, ritraduce in C un programma, dal linguaggio intermedio IR di SUIF. Ciò permette di analizzare l'effetto delle ottimizzazioni prodotte e di confrontarle con quelle di compilatori rivali.
Il grande progetto *GCC* (GNU Compiler Collection) fa capo alla organizzazione GNU e mira a fornire una linea di compilatori liberi, della stessa qualità di quelli industriali. Vi sono fronti per C, C++, Fortran, Java e Ada e retro per x86 e altre macchine.
Una gemmazione del progetto, DotGNU, si è rivolta ai linguaggi programmativi e per i servizi di Rete della Microsoft. Si ricorda che l'insieme Dot Net di Microsoft è principalmente un linguaggio intermedio, ispirato dal ByteCode di Java, di cui allarga le potenzialità e l'idoneità per diversi linguaggi sorgente.
L'affermarsi del progetto GCC ha l'effetto positivo di consolidare certi standard di fatto per le rappresentazioni intermedie, e di sottrarre lo sviluppo dei linguaggi al controllo esclusivo di una o poche aziende.
Sempre per le piattaforme Dot Net, ha avuto buona diffusione l'infrastruttura MONO, libera ma sponsorizzata da Novell.
Per il linguaggio Java esistono varie disponibilità di macchine virtuali e di compilatori dinamici; Kaffe è una delle più note. Sono questi però dei progetti di minor respiro e di sopravvivenza talvolta incerta.

14. Il futuro

L'innovazione nello sviluppo dei linguaggi di programmazione segna il passo, e i migliori tra i nuovi linguaggi che si affacciano mi sembrano delle abili rivisitazioni dei concetti classici, per adattarli a aspettative e motivazioni alquanto mutate. Il progetto dei loro compilatori (ossia dei fronti) sarà un lavoro routinario per i professionisti della compilazione.
Le direzioni di principale innovazione per la compilazione sono quelle attinenti allo sviluppo di nuove architetture di calcolo.
Per i processori, la diffusione a breve e medio termine delle architetture (esempio, Intel multi-core) con più processori interconnessi in rete, impone una complessa sinergia con il compilatore, al fine di parallelizzare e ripartire il codice, prima sui processori e poi sulle loro unità funzionali.
In un'altra direzione, per portare i linguaggi sui più piccoli dispositivi, sarà sempre necessario un artigianato di grande abilità, con attenzione alle tecniche di programmazione più compatte e efficienti.
Più a lungo termine c'è chi prevede una drastica caduta di affidabilità dei processori, causata da guasti transienti, provocati sui piccolissimi transistori delle fluttuazioni ambientali. In tale prospettiva, il compilatore sarà chiamato a generare codici eseguibili ridondanti, capaci di scoprire, e mascherare gli errori di basso livello.

Per quanto riguarda le Reti, il supporto necessario per far funzionare in modo sicuro i servizi e i programmi mobili sarà necessariamente dinamico, e la compilazione dinamica dovrà perfezionare i propri metodi di progetto, di convalida e di manutenzione.

Bibliografia

[1] Crespi Reghizzi S.: *Linguaggi formali e compilazione*. Pitagora, Bologna 2006.

[2] Aho A., Sethi R., Ullman J., Lam M.: *Compilers: Principles*. Techniques, and Tools, Addison Wesley, 2006.

[3] Appel A.: *Modern Compiler Implementation in Java*. Cambridge University Press, 2002.

[4] Muchnick S.: *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1997.

[5] Duesterwald E.: *Dynamic Compilation*. In Srikant Y. N., Shankar Priti (a cura di), *The Compiler Design Handbook*. CRC Press, 2002.

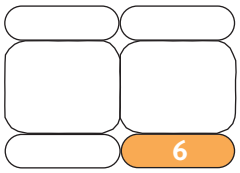
[6] Ujval Kapasi et al.: Programmable Stream Processors. *Computer*, Vol. 36, n. 8, p. 54-62., 2003.

Stefano Crespi Reghizzi lavora nel Dipartimento di Elettronica e Informazione del Politecnico di Milano. Il gruppo di ricerca da lui guidato studia la teoria dei linguaggi artificiali e progetta compilatori e macchine virtuali per i moderni processori.
Coordina il dottorato di ricerca in Ingegneria dell'Informazione: automatica, elettronica, informatica e telecomunicazioni. Insegna i corsi di Linguaggi formali e compilatori e di Analisi e ottimizzazione dei programmi, per la laurea in ingegneria informatica.
Ingegnere elettronico, ha conseguito il dottorato in computer science alla University della California di Los Angeles. Ha insegnato e collaborato scientificamente nelle università di Berkeley, Lugano, Pisa, Santiago de Chile, Stanford e Parigi. È uno dei fondatori del programma scientifico internazionale "Automata theory from mathematics to applications" della ESF, la fondazione europea per la scienza. Tra le sue opere sulla compilazione si ricorda il libro "Linguaggi formali nelle scienze della comunicazione", rivolto a lettori di matrice umanistica e il testo universitario "Linguaggi formali e compilazione" per gli ingegneri informatici.
E-mail: crespi@elet.polimi.it

LA BREVETTABILITÀ DEL SOFTWARE
IL NO DELL'EUROPA

Giovanna Sissa

Il software è tutelato dal diritto d'autore, ma talvolta si ripresenta l'ipotesi di cumulare a questa tutela anche quella brevettuale. La situazione europea è diversa da quella americana, dove sono brevettabili il software ed i "business method". Il brevetto software potrebbe avere conseguenze negative per chi opera nel settore ed in particolare per le aziende europee. La brevettabiltà delle "computer implemented invention" è stata rigettata dal Parlamento Europeo nel 2005, ma il tema non è certo fuori discussione.



1. INTRODUZIONE

L'innovazione consiste nel fare meglio quello che si faceva prima o nel riuscire a fare quello che prima era impossibile o impensabile. Quando parliamo di innovazione intendiamo, in effetti, l'innovazione tecnologica, ovvero l'invenzione, l'uso e lo sviluppo di una o più tecnologie. La convergenza di tali nuove tecnologie determina la nascita di prodotti nuovi e anche di nuovi comportamenti; è quello che gli economisti definiscono "innovazione combinatoria". Dall'invenzione del motore a scoppio sono derivate anche altre innovazioni, come le strade asfaltate e il pneumatico, ma soprattutto è cambiato il modo di spostarsi delle persone e quindi di percepire le distanze.
L'innovazione procede a salti, [1] si avvantaggia del pensiero laterale [2], del confronto dei punti di vista, della capacità di guardare a cose vecchie in una maniera nuova e di combinare cose note in modo originale; quindi consente di valorizzare la conoscenza implicita sulle cose e implementare saperi sociali diffusi. Più i soggetti produttori di conoscenza sono in comunicazione e scambio reciproco più l'innovazione procede.
Il brevetto ha avuto nella storia una sua logica di "scambio di conoscenza" nel sistema dell'innovazione: soggetti dotati raggiungono risultati concreti prima di altri, che a loro volta possono avvantaggiarsene per realizzare cose che altrimenti non sarebbero arrivati a realizzare e così via. I secondi però devono a pagare ai primi, per un periodo di tempo ragionevole, una cifra ragionevole. Se si abbassano le barriere d'ingresso l'economia del sistema consente l'innovazione.
Al di là del fatto che questa dinamica, funzionale per tanti anni nel mondo industriale, valga ancora in un'economia globalizzata, alcuni dei presupposti di tale filosofia vacillano nel momento in cui si va ad applicarla al software. Subentrano infatti molti cambiamenti: cambia il tempo di vita dei prodotti, il rapporto fra descrizione ed implementazione dell'invenzione, la molteplicità dei "sottosistemi" che un sistema software include e la possibilità di conoscere per ognuno di essi se già

esiste un brevetto. Cambia, almeno in Europa, la composizione dei soggetti coinvolti.
Il tema della brevettabilità del software è stato alla ribalta per parecchio tempo, fino alla decisione contraria del Parlamento Europeo del 2005. Se ne sta iniziando nuovamente a parlare e nuove iniziative si delineano su questo tema in ambito europeo.
Come si vede, si tratta di una situazione complessa e in movimento. È perciò opportuno cercare di fare il punto su un tema di non facile lettura anche per la comunità professionale degli informatici.

2. IL BREVETTO

L'etimologia della parola brevetto deriva dal latino *brevis*, di corta durata, che nel latino medioevale indicava un documento redatto da un notaio per conservare memoria e provare la conclusione di un negozio, dando origine al medioevale francese *bref* ed inglese *brief*, con il senso di breve scritto, e infine agli attuali diminutivi *brevet*, in francese, e *brevetto*, in italiano.
L'etimologia della traduzione inglese di brevetto, *patent*, ha invece privilegiato l'altro aspetto fondamentale di quest'istituto in quanto deriva dal latino *patens*, participio presente di *patere*, cioè essere aperto, reso pubblico.

2.1. Che cosa è

Il brevetto è un titolo giuridico in forza del quale viene conferito un monopolio temporaneo di sfruttamento dell'invenzione, in un territorio e per un periodo ben determinati, al fine di impedire ad altri di produrre, vendere o utilizzare l'invenzione senza autorizzazione. Per invenzione s'intende una soluzione nuova ed originale di un problema tecnico. Essa puó riguardare un prodotto o un processo (metodo, procedimento).
L'istituto del brevetto nasce agli inizi del secolo XIX ed è introdotto per uscire dalla strozzatura produttiva generata dal segreto sia corporativo sia del singolo artigiano, che era un freno allo sviluppo quantitativo e qualitativo della produzione. Da quest'originaria impostazione il brevetto si è evoluto, per opera delle grandi imprese, fino a diventare uno strumento di politica industriale.

WIPO - *World Intellectual Property Organization*
www.wipo.org

È un'agenzia specializzata dell'Onu, con sede a Ginevra, creata nel 1967 per promuovere la cooperazione internazionale sulla protezione della proprietà intellettuale; gestisce oltre 16 trattati multilaterali riguardanti gli aspetti giuridici e amministrativi della proprietà intellettuale. L'organizzazione elabora anche speciali modelli legislativi per le nazioni in via di sviluppo. Più di 160 Paesi sono membri della WIPO. Gli USA hanno aderito nel 1970; la Cina nel 1980.
Le attività della *WIPO* consistono nell'elaborazione di nuovi trattati internazionali in materia di proprietà intellettuale e nella realizzazione di un vasto programma di cooperazione che offre assistenza tecnica ai paesi in via di sviluppo. L'Organizzazione fornisce anche servizi destinati al settore privato, in base ad accordi internazionali che prevedono l'uso di mezzi semplificati ed economici per la tutela internazionale di brevetti, marchi di fabbrica e modelli industriali.
La WIPO provvede inoltre alla composizione delle controversie internazionali in materia di proprietà intellettuale nel settore privato.

Il **WIPO** (*World Intellectual Property Organisation*), definisce così il brevetto: "*A patent is an exclusive right granted for an invention, which is a product or a process that provides, in general, a new way of doing something, or offers a new technical solution to a problem. In order to be patentable, the invention must fulfill certain conditions*[1]".
I brevetti vengono richiesti agli **uffici brevetti**, che sono organizzazioni appositamente create dai governi per valutare e rilasciare i diritti di brevetto.

2.2. Che cosa si può brevettare e che cosa no

Per essere brevettabile [3] un'invenzione deve:
❑ presentare un carattere di **novità**, avere alcune nuove caratteristiche che vadano **oltre lo stato dell'arte del settore**;
❑ implicare un'**attività inventiva**, comportare uno "step inventivo" che non possa essere dedotto da una persona con una conoscenza media nel settore tecnico (essere **non banale**);
❑ essere suscettibile di **applicazione industriale** (avere un'utilità);
❑ apportare un **contributo tecnico**.
Fra parentesi sono indicati i requisiti di brevettabilità nello spazio giuridico USA.
In molti Paesi le teorie scientifiche, i metodi matematici, le varietà animali o vegetali, la scoperta di sostanze naturali, i metodi commerciali, o i metodi per il trattamento medico

[1] http://www.wipo.int/patentscope/en/patents_faq.html#patent

(in opposizione ai prodotti medici) sono non brevettabili. In particolare questo vale in Europa, secondo la **Convenzione Europea dei brevetti**, stipulata a Monaco nel 1973.
Il concetto di "contributo tecnico" è definito negli **accordi TRIPs**, che costituiscono la base di accordo internazionale in tema; vi si afferma che ..."i brevetti devono essere disponibili per invenzioni in ogni settore tecnologico". La delimitazione dell'ambito della brevettabilità viene così ricondotta alla definizione dei termini "settore tecnologico" e "tecnico".
Sulla definizione di "contributo tecnico" si è giocata buona parte del controverso iter della proposta Direttiva europea sul tema della brevettabilità del software. Michel Rochard, il parlamentare europeo che, come vedremo meglio oltre, è stato il relatore della seconda lettura della proposta, indica [3] come "il termine tecnico sia definito ovunque come l'insieme dei procedimenti ordinati e messi a punto scientificamente, che vengono impiegati per produrre un'opera o un determinato risultato o ancora per procedere all'investigazione e alla trasformazione della natura. Ciò che tutte queste definizioni hanno in comune è un riferimento implicito al mondo fisico, al tangibile, o ancora al reale, che si contrappone chiaramente al mondo delle idee o all'immateriale. Dopo molte ricerche, tale criterio ci è apparso come l'unico che consenta di distinguere chiaramente ciò che rientra nel settore della tecnologia e ciò che non vi rientra".

Convenzione Europea brevetti
(*European Patent Convention*), Munich 5 October 1973
http://www.european-patent-office.org/legal/epc/e/ma1.html

Article 52 - Patentable inventions
http://www.european-patent-office.org/legal/epc/e/ar52.html#A52

1. European patents shall be granted for any inventions which are susceptible of industrial application, which are new and which involve an inventive step.
2. The following in particular shall not be regarded as inventions within the meaning of paragraph 1:
a. discoveries, scientific theories and mathematical methods;
b. aesthetic creations;
c. schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;
d. presentations of information.
3. The provisions of paragraph 2 shall exclude patentability of the subject-matter or activities referred to in that provision only to the extent to which European patent application or European patent relates to such subject-matter or activities as such.

Gli accordi TRIPs in seno al WTO (*World Trade Organization*)

L'Organizzazione Mondiale del Commercio (meglio conosciuta come *World Trade Organization* - WTO) è un'organizzazione internazionale creata allo scopo di supervisionare numerosi accordi internazionali relativi al commercio tra i 149 stati membri. Nel 1994 elaborò l'accordo *Agreement on trade-related aspects of intellectual property rights*, (accordi TRIPs) in base al quale le nazioni aderenti (quasi tutte le nazioni mondiali sono membri del WTO) devono accettare pressoché tutte le condizioni della convenzione di Berna.
http://www.wto.org/english/tratop_e/trips_e/wtowip_e.htm
I TRIPs richiedono agli stati membri una stretta protezione dei diritti di proprietà intellettuale. Titolo d'esempio, sotto gli accordi TRIPs:

- i programmi di computer devono essere considerati come opere letterarie sotto la legge del copyright e godere dunque delle stesse protezioni;
- i brevetti devono essere possibili in tutti "i campi della tecnologia", con le eccezione di interesse pubblico(Art. 27.2 e 27.3 [1]).

Una recente controversia sui requisiti per la brevettabilità in ogni settore della tecnologia prevista dall'Articolo 27 riguarda proprio la necessità di consentire i brevetti del software e dei business method.
WIPO e WTO nel 1996 hanno concordato di cooperare a rendere operative gli accordi TRIPs.

2.3. Il brevetto ed il suo rapporto con l'innovazione

I diritti dei brevetti possono essere ceduti a terzi, nel caso, per esempio, che l'inventore non abbia le capacità industriali per poter sfruttare adeguatamente la sua invenzione. L'esistenza dei brevetti viene giustificata dal fatto che, grazie ai diritti di sfruttamento economico esclusivo, viene stimolata la produzione di nuove invenzioni, che diventeranno poi di pubblico dominio allo scadere del brevetto.
Al fine di stimolare lo sviluppo, un punto cardinale della normativa brevettuale in tutte le legislazioni nazionali sta nella "rivelazione dell'insegnamento inventivo[2]". Ci deve essere un rapporto equilibrato [4] fra *conoscenza pubblica ed esclusiva dell'autore*: per la concessione del brevetto è necessaria una descrizione chiara ed esaustiva dell'invenzione, che entra a far parte del patrimonio scientifico della collettività. In altre parole l'inventore deve spiegare "come si fa". Ovviamente, il "come si fa" non deve essere banale: l'invenzione deve essere costata tempo e fatica e deve costituire un reale avanzamento rispetto allo stato dell'arte. **Non deve essere neanche una legge di natura o qualcosa che è patrimonio comune di tutta l'umanità** [riquadro in alto a sinistra].

[2] ART 83 European patent Convention, "La domanda di brevetto europeo deve rivelare l'invenzione in modo sufficientemente chiaro e completo perché possa essere realizzata da una persona esperta del settore..."

Nel caso in cui il detentore del brevetto venga a conoscenza del fatto che una terza parte usa commercialmente un prodotto o servizio coperto dal suo brevetto, può agire contro la parte terza, in genere chiedendo il pagamento di royalties, che spettano al detentore del brevetto. Nel caso in cui la commercializzazione sia avvenuta in precedenza, il detentore del brevetto può richiedere i diritti retrospettivamente. In casi estremi il detentore del brevetto può richiedere che la parte terza interrompa la commercializzazione del prodotto o dei servizi che fanno uso del suo brevetto.
Ci sono differenze importanti fra i sistemi brevettuali di diverse aree geografiche. A differenza dell'Europa nel sistema USA sono brevettabili il software ed i "business method". I brevetti sono organizzati in categorie e, nello spazio legale nordamericano, i business method sono una categoria[3], così come il software.
Vedremo più avanti implicazioni e legami fra i brevetti sui business method ed il brevetto software.

3. LA PROTEZIONE INTELLETTUALE DEL SOFTWARE

Agli albori dell'informatica non c'era distinzione fra hardware e software; quest'ultimo era un complemento gratuito e senza valore economico e veniva fornito insieme all'hardware in cui invece risiedeva il valore commerciale. Il software era di fatto di dominio pubblico ed a codice sorgente aperto; non aveva infatti senso nascondere qualcosa privo di valore [5].
Negli anni sessanta il software aumenta il suo valore aggiunto al prodotto finale e così aumenta il suo costo di creazione. Tra la fine degli anni '60 e l'inizio degli anni '70, quando inizia ad emergere il mercato del software, c'è un generale consenso ad estendervi le tutele del diritto d'autore. In seguito a tale accordo in pochi anni le legislazioni dei Paesi sviluppati sono modificate in modo da proteggere il software con la normativa del diritto d'autore [6]. Simultaneamente la legislazione brevettuale è modificata in regioni, come l'Europa, per escludere esplicitamente i programmi software, che si aggiungono così a preesistenti esclusioni, come le teorie matematiche o gli algoritmi.
Il diritto d'autore come misura per proteggere il software è sancito da vari accordi internazionali, che vanno dalla **convenzione di Berna del 1979**, agli accordi TRIPs del 1994 [riquadro a p. 18] del WTO, alla vigente

La Convenzione di Berna (protezione del software: il diritto d'autore)

http://www.wipo.int/treaties/en/ip/berne/trtdocs_wo001.html

La Convenzione di Berna per la protezione delle opere letterarie e artistiche, adottata a Berna nel 1886, ha per la prima volta stabilito il riconoscimento reciproco del diritto d'autore tra le nazioni aderenti. Prima dell'adozione della convenzione, le nazioni spesso si rifiutavano di riconoscere sul materiale di nazioni straniere il diritto d'autore. La convenzione stabilisce che ogni contraente deve riconoscere come soggetto a diritto d'autore il lavoro creato da cittadini degli altri stati contraenti. La tutela è automatica, nessuna registrazione è richiesta e neppure è necessario apporre un avviso di *copyright*.
Inoltre alle nazioni firmatarie è proibito richiedere alcuna formalità come una registrazione agli autori stranieri che possa ostacolare il "godimento e l'esercizio" del diritto d'autore.
La convenzione stabilisce un termine minimo di tutela per tutta la vita dell'autore più 50 anni, ma le parti contraenti sono libere di estendere questo periodo, come ha fatto l'Unione Europea con la direttiva sull'armonizzazione del diritto d'autore nel 1993. Gli Stati Uniti hanno più volte esteso il termine di *copyright*, l'ultima volta con il *Sonny Bono Copyright Term Extension Act* nel 1998.
L'attuale durata generale del *copyright* è pari alla vita dell'autore (o dell'ultimo autore sopravvissuto se ne esiste più di uno) più 70 anni o nel caso di lavori creati da enti diversi da singoli individui, 95 anni dalla prima pubblicazione.
Gli Stati Uniti aderirono alla convenzione di Berna nel 1989 e in base a quanto in essa stabilito la nota di *copyright* non è più necessaria per ottenere la tutela del diritto d'autore (in Europa non lo è mai stata). La convenzione ebbe diverse revisioni e dal 1967 la convenzione è amministrata dalla *World Intellectual Property Organization*.

[3] Nell'8-th editione della International Patent Classification, in forza dal Gennaio 2006, la sottoclasse "business method" è la "G06Q" (che ne sostituisce una precedente).

Direttiva CEE del 14 maggio 1991 n. 250

http://europa.eu.int/eur-lex/lex/

relativa alla tutela giuridica dei programmi per elaboratore, *Gazzetta ufficiale n. L 122 del 17/05/1991 pag. 0042 - 0046*

Concerne i programmi per elaboratore espressi in qualsiasi forma, anche se incorporati nell'hardware, nonché i lavori preparatori di progettazione per realizzarli. Viene loro estesa la tutela riconosciuta dal diritto d'autore alle opere letterarie, determinando i diritti esclusivi dei quali i soggetti titolari possono avvalersi e la relativa durata.

I principi cardine di tale riforma sono i seguenti:

1. i programmi per elaboratori sono qualificati come opere letterarie e tutelati in base al diritto d'autore, purché originali;
2. la tutela non si estende ai principi, alla logica, agli algoritmi e al linguaggio di programmazione, nonché alle interfacce;
3. i diritti spettano a chi ha creato il programma: se questo è realizzato nel corso di lavoro subordinato o di un contratto d'opera spettano al datore di lavoro o al committente.

La Direttiva tratta anche il tema dell'interoperabilità all'Art. 6.

1. Per gli atti di riproduzione del codice e di traduzione della sua forma ai sensi dell'articolo 4, lettere a) e b), non è necessaria l'autorizzazione del titolare dei diritti qualora l'esecuzione di tali atti al fine di modificare la forma del codice sia indispensabile per ottenere le informazioni necessarie per conseguire l'interoperabilità con altri programmi di un programma per elaboratore creato autonomamente, purché sussistano le seguenti condizioni:

a. tali atti siano eseguiti dal licenziatario o da un'altra persona che abbia il diritto di utilizzare una copia del programma o, per loro conto, da una persona abilitata a tal fine;

b. le informazioni necessarie per ottenere l'interoperabilità non siano già facilmente e rapidamente accessibili alle persone indicate alla lettera a) e c) gli atti in questione siano limitati alle parti del programma originale necessarie per conseguire l'interoperabilità.

Direttiva CEE del 14 maggio 1991 n. 250 ed a quelle nazionali che la recepiscono.

Con la nascita del mercato dei programmi a codice chiuso, alla figura giuridica del diritto d'autore si aggiungono altre tre forme di protezione, quali il segreto industriale e commerciale, il marchio registrato e gli accordi di licenza d'uso.

Attualmente il codice proprietario (o chiuso) si regola mediante queste quattro figure. Il codice aperto si regola mediante il *copyright* [5a] (o meglio il *copyleft*) e le licenze associate (GPL, ...)[5b].

4. COPYRIGHT VS BREVETTO

Il software, dunque, è tradizionalmente protetto dal diritto d'autore, che garantisce tutti i diritti, anche quelli di commercializzazione, all'autore del programma e proibisce di copiare, ridistribuire o modificare senza autorizzazione.

Per avere i diritti su un programma è necessario e sufficiente esserne il creatore. In altre parole, provare di essere l'autore di un programma è sufficiente per garantire all'autore stesso tutti i diritti sul programma [6].

Il brevetto e il diritto d'autore sono due modi differenti [4] di garantire la tutela giuridica delle creazioni intellettuali. Implicano differenti modi di acquisizione dei diritti, differente durata del diritto e differenti strumenti di difesa da eventuali violazioni del diritto stesso. Il titolare di un brevetto e l'inventore possono non essere la stessa persona.

In particolare poi, il brevetto permette lo sfruttamento della creazione con riguardo al suo contenuto (infatti, la *moka* serve a preparare il caffè indipendentemente dal suo *design*); il diritto d'autore, invece, protegge la forma dell'espressione creativa, a prescindere dal contenuto in essa racchiuso (pertanto, non rileva che un quadro raffiguri un paesaggio, una persona o una natura morta).

I diritti sull'invenzione industriale sorgono nel momento del conseguimento del brevetto, mentre i diritti d'autore sulle opere d'ingegno sorgono nel momento stesso della creazione dell'opera.

In parole povere: la legislazione del diritto d'autore protegge il creatore di un programma, mentre quella del brevetto assegna i diritti a chi descrive le tecniche che il programma usa.

Il brevetto ha dunque una logica molto diversa.

5. IL BREVETTO SOFTWARE

Che cosa è esattamente un brevetto software? Il termine *software patent*, sebbene estremamente diffuso, è considerato da alcuni legislatori non rigoroso e talvolta vengono usati altri termini, come per esempio brevetti sulle *computer-implemented invention* (*o computer controlled invention, o computer based invention*...). Per *software patent* si intenderanno [6] qui tutti quei brevetti che possono avere un

impatto sulla commercializzazione di un programma. In altre parole, i brevetti software sono quei brevetti che sono usati per rivendicare diritti contro terze parti per la produzione, distribuzione o uso di programmi software.
In tale accezione un brevetto software può così riguardare:
1. Un servizio fornito da un programma. In questa categoria ricadono i business method erogati mediante un certo software.
2. Le funzionalità interne di un software. A seconda del concetto di software che applichiamo l'ambito brevettuale cambia. Infatti:
❑ se i programmi software sono intesi come *information process,* che restituiscono un dato in uscita dopo aver elaborato alcuni dati in ingresso, il brevetto può essere relativo a come il programma legge i dati in ingresso, a come li elabora, o a come produce i risultati in uscita;
❑ se i programmi software sono intesi come la descrizione formale di un algoritmo, scritta in forma eseguibile, allora ogni parte dell'algoritmo può essere brevettabile.
La brevettabilità del software solleva alcuni problemi che conducono a delle anomalie sul piano pratico ed a delle aberrazioni sul piano generale.
Creare un programma significa risolvere molti sottoproblemi. Nonostante questi task e sub-task siano talvolta inevitabilmente semplici e persone con media competenza del settore possano arrivare alla soluzione in qualche giorno o anche meno, essi possono formalmente essere oggetto di brevetto. Si parla in questo caso di *trivial patent*.
Spesso i brevetti software rilasciati coprono *realizzazioni che sono obsolete al momento stesso della richiesta*. Ovviamente, una volta concesso il brevetto, nessuno può muoversi in quella parte dello scibile umano senza pagare o essere portato in tribunale.
Inoltre la possibilità di violarli senza saperlo è altissima; l'autore di un programma può usare un elemento brevettato senza saperlo, semplicemente "reinventando il concetto da sé".
Un altro problema è come valutare il "passo inventivo" necessario per l'ottenimento di un brevetto.
Negli altri campi tecnologici il brevetto si riferisce al prodotto finito, cioè all'idea e all'arduo lavoro di realizzarla al fine di produrre un "insegnamento inventivo sull'uso delle forze naturali controllabili". Le forze della natura non entrano nel software, che rimane una creazione logica.
Problema centrale è quello dell'interoperabilità. Per garantire l'interoperabilità, gli art. 5 e 6 della direttiva CEE 250/591 [riquadro a p. 20] stabiliscono le modalità, nel caso in cui sia necessario il ricorso ad un processo brevettato per il solo scopo di assicurare l'interoperabilità fra due sistemi, affinché tale ricorso non sia considerato una violazione del brevetto. L'uso di standard brevettati finisce per forzare l'uso di altri formati alternativi. Per esempio, il formato PNG è stato introdotto per evitare problemi con il brevetto GIF e il formato OGG è stato introdotto per evitare i brevetti di MP3. Il tema dell'interoperabilità è stato anche al centro del dibattito in sede di istituzioni europee sulle *computer implemented invention* [3].
La ricerca e lo sviluppo costano molto e il brevetto dovrebbe garantire, nella filosofia originaria, un giusto ritorno di utili a chi li finanzia. Esso istituisce un monopolio dell'inventore sulla sua invenzione e per un certo periodo chiunque intenda servirsene deve pagare.
Tuttavia, il brevetto sul software non è assimilabile agli altri brevetti[6]. Infatti il software è considerato come un'opera intellettuale e protetto dai diritti d'autore come un romanzo o un quadro d'autore. È quindi il modo con cui è espressa l'idea che beneficia della protezione e non le parole che sono servite a scrivere il testo o il processo intellettuale che ha portato, sotto la mano dell'autore, alla stesura del romanzo.
Ecco alcuni eclatanti esempi [5, 6] di brevetti software esistenti e validi nello spazio giuridico americano, che sembrano più brevetti "sul problema" che "sulla soluzione del problema" stesso:
1. Sistema e metodo per *computer based test*. È un programma per verificare se un bambino ha compreso un materiale didattico, sulla base della risposta dell'individuo ad un determinato questionario (US5565316). Sostanzialmente descrive un sistema per la preparazione, somministrazione e valutazione di test attraverso una rete di computer.
2. Sistema per disporre un ordine d'acquisto attraverso una rete di comunicazioni. Questo brevetto è conosciuto come brevetto

Amazon *one-click*, e descrive l'uso di cookie per completare un acquisto di beni in un sito di *e-commerce* (US5960411). Descrive lo scambio di informazioni tra server e client per la vendita on line.

3. Compressione nelle comunicazioni *wireless*. Una connessione TCP standard tra due computer è utilizzata per multiplexare le comunicazioni tra applicazioni in entrambe le direzioni, e così ridurre i dati trasmessi (US5867661). Utilizza dei canali di comunicazione virtuali per il trasferimento di dati tra applicazioni poste su computer collegati tramite una rete TCP, multiplexati per il trasporto sul canale reale.

La commercializzazione di qualsiasi software o servizio basato su un software che viola uno qualsiasi dei singoli punti è soggetto ad autorizzazione dal detentore del brevetto, nello spazio legale americano dove hanno effettivo valore.

I lettori interessati possono consultare il testo completo di questi brevetti, incluso i loro *claim*, sul sito dell'Ufficio Statunitense dei brevetti [13], inserendo solo il numero indicato fra parentesi, senza il prefisso "US".

5.1. L'evoluzione del brevetto software in USA

Si è visto al paragrafo 2 come sia evoluta la situazione del software (che nasce a codice aperto e libero e che agli albori era visto come un "corredo all'hardware") il cui valore aggiunto è cresciuto nel tempo, diventando quindi oggetto da proteggere, sotto la tutela del diritto d'autore.

Negli Stati Uniti, come accennato, vi sono due importanti differenze rispetto all'Europa in tema di brevetti: il diverso peso della giurisprudenza nella pratica legale, che è maggiore negli USA, e l'assenza in pratica del requisito di tecnicità, (per cui è normale rilasciare brevetti a qualunque "novità" che possa "essere utile") che è invece presente in Europa [5, 6].

Se all'inizio c'era una certa ritrosia a rilasciare brevetti software, dovuta principalmente alla mancanza di esperti negli uffici preposti, con il tempo la pressione dei principali attori ha fatto cambiare questo atteggiamento; IBM è l'azienda che ha il maggior numero di brevetti concessi.

Durante gli anni '80 e '90 le corti degli USA hanno accettato molti casi di brevetti relativi al software. Vent'anni dopo, la pratica del brevetto software è ormai consolidata e considerata valida. La situazione si è resa perfettamente chiara verso la metà degli anni '90, con la registrazione di centinaia di migliaia di pratiche di brevetti software da parte dell"Ufficio americano brevetti (USPTO[4] - *United States Patent and Trademark Office*) che nel 1996 ha rilasciato le *Final Computer Related Examination Guideline*[5], le linee guida *ad hoc*.

Si deve attendere però l'era delle "dot COM", perché la pratica del brevetto del software giochi un ruolo importante nell'economia del settore. Affinché l'edificio del brevetto diventi effettivo è necessario infatti che i titolari possano effettivamente esigere i diritti derivanti dal loro monopolio e che siano quindi sanzionati coloro che violano i monopoli ventennali. Sono dunque dovuti trascorrere alcuni anni affinché le richieste siano state accordate, i brevetti siano validi e siano partite le richieste verso i terzi.

L'entità degli indennizzi richiesti per le violazioni (centinaia di migliaia di dollari), il numero di brevetti richiesti, ma soprattutto il numero di cause legali e di relative sanzioni economiche comminate, raggiungono cifre strabilianti [5, 6].

Si possono brevettare sia il software che i cosiddetti "business method"; in realtà la distinzione è inesistente, dato che nei brevetti cosiddetti "software" ciò che è brevettato non è il codice – che, fra l'altro, non è pubblicato nella descrizione dell'invenzione - ma la "funzione astratta" svolta dal particolare codice sviluppato ed è il carattere astratto della funzione brevettata che rende, in linea di principio, il brevetto opponibile ad un'altra implementazione della funzione stessa.

Nello spazio giuridico degli Stati Uniti d'America, ogni nuovo servizio in rete può essere (e spesso è) brevettato come software o "business method", il che determina immediatamente, per definizione, una situazione di monopolio [7] per il titolare del nuovo servizio, o meglio, per colui che ha richiesto per primo il rilascio del brevetto - monopolio che può tra-

[4] http://www.uspto.gov/

[5] http://www.bitlaw.com/source/soft_pats/final.html

dursi in una rendita di posizione o nell'interdizione di erogare il servizio per i potenziali concorrenti.
È solo di qualche anno fa la causa vinta dal portale *amazon.com* contro un suo concorrente per il fatto che quest'ultimo aveva copiato la funzionalità bevettata di fare acquisti con tre *click*, indipendentemente dal fatto che il suo concorrente avesse realizzato la funzionalità con un codice sorgente diverso.

5.2. La situazione in Europa

Il brevetto è regolato in Europa dalla Convenzione Europea dei brevetti, stipulata a Monaco nel 1973, recepita dalle varie legislazioni nazionali. L'articolo 52.2 stabilisce in modo esplicito che il software non è brevettabile e l'articolo 52.3 che i programmi per elaboratore sono esclusi "in quanto tali" [8]. L'intenzione del legislatore era esprimere che mentre il software non era brevettabile, lo avrebbero potuto essere le invenzioni che integravano software, sempre che tale software non fosse esso stesso oggetto di brevetto (software "in quanto tale").
Dall'interpretazione dell'espressione "in quanto tale" (*as such*) data dall'Ufficio Europeo dei Brevetti (*European Patent Office* - EPO) [9] derivano tutte le controversie degli ultimi anni su questo tema in Europa. Secondo l'EPO è brevettabile "un effetto tecnico addizionale che va molto oltre le normali interazioni" fra programma e computer, senza quantificare chiaramente però che cosa significa "addizionale", che cosa significa "tecnico", che cosa s'intende per "molto oltre" e per "normale".
Non si tratta di definizioni e basta. Da anni l'EPO rilascia brevetti software; nella guida delle valutazioni del 2001 viene introdotto il termine *computer-implemented invention* per poter brevettare qualunque idea che si implementa in modo informatico, sia o no tecnica (come i business method). Di fatto l'interpretazione fornita dall'EPO corrisponde, anche se non viene esplicitamente dichiarato, a quella dello spazio giuridico degli USA [3, 5, 6, 10].
L'interpretazione [3] data dall'EPO è in contrasto con la citata Convenzione di Monaco che descrive come "un programma di computer può assumere varie forme, come un algoritmo, un diagramma di flusso, una serie di istruzioni codificate, che possono essere registrate su nastro o altro supporto di lettura per una macchina e si può considerare come un caso particolare di un metodo matematico o una rappresentazione di informazioni".
Questa definizione di programma esclude assolutamente la brevettabilità del software (immateriale), mentre contempla la possibilità di brevettare invenzioni tradizionali (materiali) che contengano software al loro interno, sempre che non sia il software stesso oggetto del brevetto (*as such)*.
Il risultato di questa disarmonia fra l'EPO e le legislazioni degli stati membri sono i 30.000 brevetti concessi. Anche se sono, nei fatti, senza un riconoscimento effettivo: non sono infatti rivendicabili perché le legislazioni nazionali sono più vicine alle intenzioni originali della Convenzione di Monaco che l'attuale pratica dell'*European Patent Office* [6].
Alla battaglia contro la brevettabilità delle *computer-implemented invention* hanno partecipato molte associazioni e gruppi europei. Particolarmente attiva la *Foundation for a Free Information Infrastructure* (FFII) [10] che, per dare un'idea di quello che potrebbe aspettarci, ha realizzato, come esempio, il "sito di commercio elettronico europeo" [11]. In questo sito (Figura 1) viene simulato un servizio di *e-commerce* banale e vengono indicati tutti i brevetti software già rilasciati dall'Ufficio Europeo dei Brevetti, che sarebbero violati se le legislazioni, europee e nazionali, venissero modificate in tal senso.

FIGURA 1

Immagine tratta dalla pagina web http://webshop.ffii.de/

I numeri riportati nella figura 1 indicano i brevetti software o i business methods e le relative sigle, come rilasciati dall'Ufficio Europeo dei Brevetti[6]. Si va da software banali a businnes method. Esaminandoli, nell'ordine, troviamo:
1. *negozio on-line*: Vendere merce in rete, usando un server, un client e un gestore dei pagamenti, o usando un client ed un server - EP803105 e EP738446;
2. *ordinare tramite il telefono cellulare*: Vendere su una rete di telefonia mobile - EP1090494;
3. *carrello*: Carrello elettronico - EP807891 e EP784279;
4. *[CD] [Film] [Libri]*: Presentati tramite linguette (Tabbed palettes) - EP689133;
5. *collegamento all'immagine*: Finestra di anteprima - EP537100.
Si prosegue poi, per esempio, con:
11. *richiesta di prestito*: Domanda automatizzata di prestito - EP715740;
oppure con:
15. *database di supporto*: Supporto diretto via Rete con uso di database - EP673135;
fino al:
19. *codice di sconto*: Rende possibile ai clienti di inserire il codice per ottenere uno sconto - EP370847.
Sempre la FFII ha realizzato una "galleria degli orrori" dei brevetti europei [12], che include sia esempi di brevetti malfatti, sia esempi di impedimenti allo sviluppo software a causa di brevetti.

6. GLI EFFETTI DEL BREVETTO SOFTWARE

I brevetti astratti americani vengono spesso acquisiti da grosse società del settore informatico che li usano come merce di scambio con altre società, oppure da persone giuridiche create appositamente, le cosiddette "litigation companies", la cui unica attività è riscuotere licenze d'uso sui brevetti che detengono, senza svolgere alcuna attività produttiva né inventiva [14].
Nel meccanismo del brevetto solo una grande corporation può trarre vantaggio. Le grandi aziende possono acquistare il brevetto per cifre limitate (o attraverso transazione extragiudiziale) e utilizzarlo per ottenerne una rendita di posizione.
La costituzione da parte delle maggiori imprese del settore di portafogli di brevetti (software e business method) e lo scambio reciproco (sotto forma di baratto, che permette inoltre di sottrarsi all'imposizione della tassa sul valore aggiunto) di licenze su tali portafogli permette la costituzione di un vero e proprio cartello, che ha la capacità di impedire alle nuove imprese di entrare nel mercato. È praticamente impossibile concepire oggi un nuovo servizio in rete senza scontrarsi con un numero incalcolabile e sempre crescente di brevetti.
I titolari di brevetti software generano introiti dalla commercializzazione esclusiva di prodotti o servizi basati su questi brevetti, o dalla richiesta verso terzi di royalities per usare i brevetti stessi. Ogni studio sugli effetti economici deve dunque includere i costi che il sistema del brevetto software avrà sui creatori di software e sul settore ICT in generale [5, 6]. Come minimo, ogni produttore di software dovrà tenere in considerazione i costi richiesti per effettuare un'analisi dei brevetti relativi alla sua produzione, per identificare il numero di brevetti che possono riguardarlo e per preventivare le royalties da pagare per brevetti non correttamente identificati nel processo di analisi.
Chiunque lavori in un campo informatico produce in continuazione nuove idee o nuovi programmi per elaboratore e spesso la stessa procedura viene realizzata indipendentemente da vari autori [14]. Anche chi detiene un brevetto si dovrà, a sua volta, scontrare con innumerevoli altri brevetti non appena svolgerà un'attività produttiva. Quindi sarà tutelato solo in parte e soprattutto sarà efficacemente tutelato solo chi è inserito in strutture o organizzazioni che possono contare su uffici e sezioni che conducono in modo sistematico questo tipo di ricerca e forniscono questo tipo di assistenza (si veda, l'intervista col prof. Ceri a p. 25). Ottenere un brevetto non è una pratica semplice, per cui molte piccole imprese semplicemente non potranno usufruire di questa possibilità, ma dovranno lavorare in un campo minato dai brevetti realizzati dai loro concorrenti.

[6] http://ep.espacenet.com/ sul sito dell'Ufficio Europeo dei Brevetti si può consultare qualunque brevetto, effettuando la ricerca per esempio per numero.

INTERVISTA A STEFANO CERI DEL POLITECNICO DI MILANO

Prof. Ceri, lei ha una esperienza diretta di brevetto software. Può illustrare brevemente il percorso di brevettazione e le motivazioni che hanno guidato le scelte di brevettare il risultato di una ricerca universitaria?

L'idea da brevettare risale al 1998, durante la partecipazione ad un progetto europeo (W3I3, del Quinto Programma Quadro UE). Il progetto studia le tecniche e le metodologie per la pubblicazione di applicazioni per il Web; nasce così una "invenzione a due nomi" (il mio e quello del mio collega Piero Fraternali) di proprietà del Politecnico di Milano, relativa a Web Modeling Language (WebML), un linguaggio innovativo per la specifica visuale dei requisiti di un'applicazione Web con una notazione grafica di alto livello, in grado di descrivere sia la struttura dei dati, sia i contenuti delle pagine Web e i loro collegamenti ipertestuali, sia le regole estetiche di presentazione.

La brevettazione è un processo lungo e per certi versi imprevedibile. Abbiamo innanzitutto appreso cosa brevettare (un "metodo"), come (definendo l'"arte nota", sviluppandone una "descrizione" e infine individuando le nostre "rivendicazioni"), dove (in genere si fa prima una domanda italiana per definire l'anteriorità e guadagnare così un anno di tempo per presentare domanda in sede Europea ed Americana), in che lingua (la formulazione di un brevetto usa un linguaggio specialistico e quindi è indispensabile farsi assistere da uno studio di ingegneria competente). Dopo questo apprendistato, e grazie al supporto dell'Ufficio di *Trasferimento Tecnologico* (TTO) che il Politecnico di Milano ha inaugurato con noi (e che ora segue una cinquantina di brevetti all'anno), abbiamo depositato una domanda di brevetto (nel 1999) che è stata accolta dal Patent Office americano dopo ben tre anni, nel luglio del 2003, dopo accanite discussioni con revisori abbastanza incompetenti. Della domanda Europea si è persa traccia.

Se l'iter è lungo e nel nostro caso abbastanza laborioso, ci si può chiedere: val la pena di brevettare? La questione è controversa, specie per un ambito accademico che – secondo alcuni stereotipi – dovrebbe essere di vocazione contrario alla "proprietà delle idee". In realtà la mia risposta è certamente positiva. Nel nostro caso, lo sfruttamento del brevetto è stato inizialmente concesso, in modo non esclusivo, a una delle aziende partner del progetto W3I3 per la realizzazione di un prodotto di gestione di contenuti su vari canali: si tratta di un caso piuttosto raro di un brevetto del Politecnico ceduto commercialmente ad un'azienda privata, ed in tal caso il brevetto è servito a definire con correttezza le proprietà intellettuali e a valorizzare il nostro apporto.

Il brevetto è poi uno degli *asset* (assieme alle competenze, agli strumenti, alle persone) utili per intraprendere una qualsiasi azione di valorizzazione del lavoro fatto. Noi abbiamo creato nel 2001 Web Models s.r.l., società *spinoff* - partecipata dal Politecnico - costruita allo scopo di perseguire l'industrializzazione e la commercializzazione dell'idea; la società, dopo tre anni di sviluppo, ha messo sul mercato WebRatio™, uno strumento di produttività espressamente dedicato agli sviluppatori Web che si pone come obiettivi l'aumento della qualità del progetto e quindi anche la diminuzione dei suoi tempi e costi. Abbiamo constatato, andando varie volte a presentare il sistema – soprattutto nei contesti internazionali – che essere coperti da un brevetto è elemento indispensabile per affermare la "proprietà" e "originalità"dell'idea, anche se poi non è chiarissimo fino a che punto la protezione sia utilizzabile nei confronti di imitatori, specie se agguerriti.

Un mito da sfatare è che brevettazione e uso accademico di un'idea siano in contrasto. Un libro su WebML é pubblicato da Morgan-Kaufmann nel 2002, con edizione italiana McGraw-Hill del 2003; corsi su WebML sono offerti in circa 50 università distribuite in tutto il mondo, e abbiamo già contato circa 2000 *download* gratuiti nell'ambito del nostro "programma accademico". Il vero problema non è tanto se brevettare un'idea che abbia un possibile sviluppo commerciale (nel dubbio conviene farlo) quanto piuttosto se lanciarsi in prima persona in esperienze di *startup* a forte contenuto tecnologico, specie in un settore – come quello della Information Technology – dominato dalle tecnologie internazionali. La sfida è risultata finora avvincente; il suo esito dipende anche dal modo in cui le nostre istituzioni nazionali (e più in generale il mercato, in primis quello nazionale e poi quello internazionale) sapranno valorizzare esperienze di innovazione che nascono dalla ricerca italiana.

Un'altra caratteristica fondamentale del sistema brevettuale è la limitazione temporale [5, 6, 14] del monopolio garantito all'inventore. Tale limitazione è stabilita al fine di non bloccare lo sviluppo tecnologico del sistema produttivo, pur garantendo all'inventore un arco di tempo in cui godere in modo esclusivo dell'invenzione e recuperare gli investimenti di ricerca. In quest'arco di tempo l'insegnamento inventivo è comunque già stato pubblicato e arricchisce il patrimonio culturale complessivo. Mentre garantire un monopolio di venti anni può essere sensato nel campo delle realizzazioni meccaniche o idrauliche, tale arco di tempo non ha correlazione con il ciclo di vita di un pacchetto software, che solo di rado è così lungo e che, specialmente nell'era del Web 2.0, tende invece a misurarsi in due o tre anni al massimo. Una copertura brevettuale largamente superiore al ciclo di vita di un prodotto non può che bloccare la crescita culturale e limitare lo sviluppo complessivo di un settore produttivo, nuocendo quindi agli operatori del settore, con la sola esclusione dei pochi che si sono assicurati una copertura brevettuale sufficiente a non venire schiacciati da portafogli più nutriti.

7. IL BREVETTO SOFTWARE NELLE AZIENDE

7.1. Piccole e medie imprese informatiche

La maggior parte delle aziende informatiche in Europa sono SME, mentre, con pochissime eccezioni, le grandi multinazionali informatiche sono prevalentemente nordamericane.
Il prezzo di un brevetto software europeo, contando tutte le spese necessarie, si aggira fra i 30.000 ed i 50.000 €; il tempo per ottenerlo è tra i 3 e 4 anni; un giudizio per violazione supera il milione di euro per entrambe le parti [5].
Rispetto al brevetto software le aziende si possono dividere in alcune categorie principali: le imprese orientate all'innovazione, quelle orientate al cliente e quelle "non produttive".
Le imprese orientate all'innovazione se registrano brevetti ne trarranno vantaggio solo quando questi saranno loro accordati e potranno dunque sfruttarli; generalmente l'incertezza su queste pratiche è forte. Considerazioni analoghe valgono per quelle che si possono definire micro-imprese.
Un discorso a parte invece va fatto per gli *spin off* universitari o di ricerca, che costituiscono un tipo particolare di piccola impresa. Nate con questo scopo, queste piccole aziende, decisamente orientate specificamente proprio all'innovazione, possono trarre beneficio dal sistema brevettuale se la struttura di ricerca all'interno di cui sono state originate fornisce loro tutto il supporto necessario per non incorrere nei rischi che il sistema brevettuale pone e se garantisce inoltre supporto, servizi ed assistenza sia a valle della progettazione del software innovativo che a monte. La capacità di attrarre capitale umano, di trasferire i saperi all'industria e alla comunità (e viceversa), di garantire l'osmosi tra la produzione intellettuale e la sua applicazione commerciale e pubblica che i centri di produzione del sapere, soprattutto le Università, dovrebbero avere, comporta proprio questo.
La testimonianza del prof. Ceri del Politecnico di Milano, protagonista in quanto autore e titolare di brevetto software, evidenzia l'importanza, anche per quanto riguarda il percorso brevettuale, del supporto su cui ha potuto contare.
Vi sono poi le imprese orientate al cliente, che sono quelle imprese informatiche che possiamo chiamare normali: il loro modello di business ha avuto successo e offrono ai loro clienti servizi o sviluppo software (e hardware). Sono imprese realmente produttive. Per tali imprese i benefici che si possono ottenere da qualche brevetto rilasciato, va commisurato con il costo rappresentato dal pagare i molti brevetti che certamente saranno violati dal software sviluppato. D'altra parte il possibile beneficio ottenuto dalla supposta diffusione dell'innovazione (come si è detto, brevettare significa "rendere pubblico") è, in questo caso, discutibile: quale informatico è in grado di consultare il database dei 300.000 brevetti software statunitensi, prima di progettare un proprio software?
Infine troviamo le imprese "non produttive", costituite da avvocati, dedicate alla acquisizione di brevetti software [5]. Ovviamente l'innovazione e la creazione di software sono estranei al loro business. Questo tipo di impresa certamente si avvantaggia dal brevetto software, poiché, senza correre alcun rischio di violare brevetti al-

trui, può invece far valere i propri, esigendo licenze da tutti coloro che le violeranno nel mercato o vendendoli al miglior offerente. Inoltre è particolarmente agressiva nell'esigere pagamenti da parte dell'impresa informatica produttiva. DI questo devono tenere conto le piccole imprese cui può costare più caro pagare un avvocato che non le licenze richieste.

7.2. Grandi multinazionali del software

Le grandi multinazionali del software e della tecnologia hanno su questo terreno un forte vantaggio rispetto alla SME: la possibilità di ricorrere a frequenti macro-accordi d'interscambio di portafogli di brevetti. Tale possibilità vale ovviamente solo per un insieme ristretto di aziende.
In questo gruppo di imprese si possono distinguere due tipi. Quelle che detengono un monopolio di fatto in una nicchia più o meno ampia e le altre.
Per le aziende che detengono il monopolio o quote di mercato più alte nella propria nicchia (Microsoft, Oracle, SAP, Cisco, Adobe ecc.) i brevetti software possono rappresentare un modo di mantenere il proprio monopolio, impedendo che i concorrenti abbiano accesso a risorse software che in alcuni casi sono indispensabili.
A fronte di questo beneficio di perpetuare o aumentare ulteriormente la propria quota di mercato o di monopolio, il pagare talvolta varie centinaia di milioni di dollari è un rischio perfettamente assumibile: caso Eolas vs. Microsoft, e-Plus vs. SAP. Come si ricorderà Eolas, aveva chiesto a Microsoft, per le tecnologie comprese nel browser Internet Explorer, un enorme ammontare di danni. La battaglia legale è in corso e al momento in cui viene pubblicato questo articolo Microsoft ha annunciato che apporterà una modifica al proprio browser, relativa al controllo degli ActiveX [15] per non violare i brevetti Eolas.
Fra quelle che non detengono la quota di mercato più ampia in una certa nicchia il possibile guadagno derivante dai loro brevetti e le perdite per violazioni di brevetti potrebbero arrivare a compensarsi. Tali aziende devono sempre fare i conti con l'incertezza che il loro prodotto principale, il software, violi brevetti impossibili da eludere e che per questo debbano pagare decine o centinaia di milioni di dollari non previsti: si vedano i casi di Kodak vs. Sun, e-Plus vs. Ariba, NTP vs. RIM(BlackBerry) [5,6].
D'altronde per nessuna delle grandi aziende la tariffa per ottenere i brevetti o le cause legali per esigere le royalities dei brevetti rappresentano un costo significativo, almeno in paragone alle SME.

7.3. Soluzioni software di aziende non informatiche

Le società di questo tipo sono obbligate a pagare le licenze ed a non violare i brevetti software esistenti o in arrivo. Tali imprese si troveranno di fronte ad una seria problematica, poiché è molto comune il fatto che un loro sviluppo sia conosciuto da molti sviluppatori che non sono dell'azienda o che un giorno o l'altro smetteranno di esserlo (20 anni sono tanti) e che possono lavorare oggi in una società di consulenza e domani in un'altra. È dunque possibile che i detentori di brevetti vengano a conoscenza di tali sviluppi e richiedano indennizzi molto consistenti per i brevetti violati [5].
Può accadere che ex-dipendenti tentino delle forme di estorsione minacciando di svelare delle violazioni.
Questo rischio è ancora più elevato per aziende che erogano servizi massicciamente, come aziende di telefonia o ASP/ISP, poiché la cifra di indennizzo dipende dal numero di utenti finali.

7.5. Il mondo del software libero

I modelli di distribuzione gratuita del software potrebbero trovarsi in grosso svantaggio competitivo rispetto ad altri modelli di distribuzione. Per essi infatti è impossibile conoscere il numero di copie che un determinato pacchetto software ha nel mercato; è pertanto impossibile quantificare l'indennizzo o il numero di licenze per copia che dovrebbero pagare nel non raro caso di violare qualche brevetto [5, 6].
Per i detentori di brevetti software è molto più facile comprovare e controllare se il proprio brevetto è stato violato perché hanno a disposizione il codice sorgente.
In definitiva, chi è più danneggiato da un mercato in cui siano legalizzati i brevetti software sono gli utenti del software libero.
Sul tema del software libero e del suo rapporto con il brevetto software si veda l'Intervista a Stefano Maffulli a p. 28, presidente della Free Software Foundation Europe.

INTERVISTA A STEFANO MAFFULLI, PRESIDENTE DELLA FREE SOFTWARE FOUNDATION EUROPE

La brevettabilità del sw, secondo la FSF, in che modo avrebbe un impatto sullo sviluppo del sw libero?

L'impatto sarebbe lo stesso deleterio impatto dei brevetti software sul software proprietario. Non è corretto pensare che il software libero sia diverso in questo contesto: tanto RIM (l'azienda che distribuisce il cellulare Blackberry) o anche Microsoft quanto il kernel Linux sono minacciati allo stesso modo.
La differenza sta nelle scappatoie che i grossi produttori di software come Microsoft possono prendere. Per esempio basti pensare a come IBM reagì all'attacco di SCO per supposta violazione di diritto d'autore: con una contro-citazione per violazione di 4 brevetti software a caso.
Le grandi aziende distributrici di software hanno a disposizione armi di rappresaglia, mentre gli sviluppatori di software libero no. Ma la vulnerabilità è identica.
Inoltre le grandi aziende fanno cartello condividendo grandi quantità di brevetti, concedendosi licenze reciproche in cambio di patti di non belligeranza. Di fatto questi accordi servono a costruire barriere all'ingresso di nuovi concorrenti sul mercato. E queste aziende sono al 90% colossi statunitensi.
La nuova bozza della licenza GNU GPLv3 ha delle nuove clausole di rappresaglia contro i brevetti software volte ad allungare il mantello protettivo di questi accordi di scambio tra aziende in modo da proteggere anche gli utenti finali, ignorati da questo tipo di accordi.
Per esempio, con la bozza di GPLv3 si prevede che se un'azienda distributore di software libero ha un accordo di scambio di brevetti con un terzo, allora il distributore dovrà impegnarsi ad indennizzare e proteggere da cause di violazione di brevetti anche i suoi clienti.
In sintesi i brevetti software:

- instaurano dei monopoli su idee astratte;
- ostacolano l'innovazione potendo essere rilasciati senza aver prodotto alcuna implementazione (codice sorgente) e impediscono di fatto la ricerca;
- ostacolano la concorrenza di mercato, garantendo un potere di mercato spropositato ai soggetti dominanti;
- ostacolano la rivelazione delle idee, la motivazione originaria per l'introduzione del sistema brevettuale;
- riducono la competitività europea;
- ostacolano l'interoperabilità, aumentando la dipendenza da un singolo fornitore;
- diffondono i loro effetti negativi in molte altre aree dell'economia.

E sul mercato del software libero avrebbe ripercussioni, dirette o indirette?

Di nuovo, le ripercussioni le avrebbe il mercato, tout-court. Per parafrasare un comico, "i brevetti sono una livella": davanti ad essi siamo tutti uguali e tutti vulnerabili.
Per esempio, il progetto Samba sta sviluppando un sistema di server groupware innovativo, basato sullo standard CIFS e conduce lo sviluppo in maniera indipendente, studiando le specifiche dello standard aperto CIFS e analizzando il traffico di rete per raggiungere delle funzioni di interoperabilità con i server Microsoft. Tutto il codice scritto da Samba è scritto senza guardare codice o altro materiale che non sia già pubblico, legalmente accessibile ai programmatori del team.
Ebbene, una qualsiasi azienda, concorrente di Samba o semplicemente un *patent-troll* [aziende che non sviluppano e non spendono in R&D, ma comprano diritti di brevetti da università e altre aziende per intentare cause legali e farsi pagare, esempio Eolas], potrebbe reclamare prima o poi che il lavoro del Samba team infrange un loro brevetto e pertanto chiedere un pizzo o fermarne completamente la distribuzione.
Questo tipo di attacco lo sta subendo RIM per violazione di brevetto da parte del suo sistema Blackberry: un'azienda canadese innovativa, benché non abbia nulla a che fare con il software libero, è messa in ginocchio da un brevetto statunitense e forse sarà costretta a pagare il pizzo ad un *patent-troll* per poter continuare a vivere... fino alla prossima causa.
Il settore del software è un'opportunità per l'Europa; che i brevetti software stanno invece consegnando nelle mani di altri.

8. LA VICENDA DELLA SOFTWARE PATENT NELLE ISTITUZIONI EUROPEE

Tutti concordano su un punto: l'Europa necessita di maggiore innovazione. La strategia di Lisbona lo dice: la necessità di far fronte alla concorrenza americana e giapponese sul mercato globale è ormai un'evidenza. Negli Stati membri della UE, come già detto, il software in quanto tale (che non è destinato ad applicazioni industriali e non apporta un contributo tecnico) non è brevettabile [8]. Esso beneficia, allo stesso titolo di un'opera letteraria, della tutela dei diritti d'autore.
In pratica, ogni Stato ha il suo modo di vedere la questione ed il suo specifico sistema di brevetti. Non solo le modalità di registrazione dei brevetti, ma anche le definizioni di ciò che è brevettabile e di ciò che non lo è possono divergere. Oggi in Europa, un brevetto può essere concesso dagli uffici nazionali e dall'Ufficio Europeo dei Brevetti. Va osservato come l'Ufficio Europeo dei Brevetti non sia un organismo della Commissione Europea, ma un organo amministrativo, autonomo, esterno, indipendente e senza un diretto controllo pubblico [5, 6, 10, 14].
La Commissione Europea dal 2000 ha più volte dichiarato di voler armonizzare i sistemi nazionali sulla brevettabilità delle innovazioni [16] che, per essere realizzate, necessitano di un software specifico. Con tale intenzione la Direzione Generale del Mercato Interno della Commissione Europea ha presentato nel 2002 una proposta di direttiva europea sulla "brevettabilità delle invenzioni attuate per mezzo di elaboratori elettronici". Secondo la Commissione non si trattava di modificare alcunché, ma di basarsi sulle prassi esistenti. In altre parole di sancire e rendere legalmente validi i brevetti rilasciati dall'EPO. Secondo il parere di molti analisti, questo avrebbe esteso l'ambito della brevettabilità in Europa, finendo per includere i programmi software, in modo analogo a quanto avviene negli USA [6]. La proposta di direttiva era sta accolta con entusiasmo da alcuni gruppi di pressione, come l'Ufficio Europeo dei Brevetti, esperti di Proprietà Intellettuale e grandi aziende, mentre era stata duramente avversata dal mondo delle piccole aziende, organizzazioni professionali di informatici e dalla comunità del software libero. Contrarietà era stata espressa anche da diversi partiti politici e da autorità garanti (per esempio, quella della concorrenza e del mercato tedesca). Anche l'industria era divisa, ma quella parte di essa che si opponeva alla direttiva era, in termini di fatturato, la meno importante [8].
Sulla questione, oggetto di dibattiti per tre anni, i deputati del Parlamento Europeo erano sempre stati divisi. Raggiungere un compromesso sembrava un'impresa difficile, ma i deputati vi erano riusciti nel settembre 2003. Pur dicendosi favorevoli alla brevettabilità delle "invenzioni attuate mediante computer" avevano limitato l'ambito di applicazione della direttiva per evitare la brevettabilità del software in quanto tale, inserendo, nella proposta della Commissione e del Consiglio, molti emendamenti. Ma proprio i principali emendamenti presentati dal Parlamento sono stati poi respinti dal Consiglio. Per la prima volta i deputati hanno così ritenuto che fosse utile ricorrere ad una fase ulteriore di negoziazione (la cosiddetta "procedura di conciliazione" prevista dal trattato): le divergenze erano troppo marcate. Dopo un lungo e complesso braccio di ferro sul piano istituzionale, nel marzo del 2005 la conferenza dei presidenti di gruppo del Parlamento europeo adottò la risoluzione del comitato legislativo (JURI) del Parlamento di rimandare a capo tutto il processo [17].
Il Parlamento non vedendo alcuna possibilità di compromesso sulla base della posizione comune degli Stati Membri, ha respinto così il testo nella sua totalità a grandissima maggioranza: 648 voti favorevoli, 14 contrari e 18 astensioni. Si tratta della prima volta nella storia che il Parlamento respinge una proposta legislativa in seconda lettura della procedura di codecisione.
Al centro dell'aspro dibattito tra le istituzioni europee con da un lato la Commissione ed il Consiglio dell'UE e dall'altro buona parte del Parlamento europeo, ci sono state soprattutto le conseguenze possibili sul mercato europeo delle nuove tecnologie. La bozza di direttiva, così come elaborata dall'esecutivo UE e approvata dai ministri della competitività dell'UE nel marzo 2005, non riconosceva in linea di principio la brevettabilità dei software in quanto tali, ma introduceva la possibilità

di brevettare le invenzioni che li utilizzano nell'ambito di un sistema e che vi fanno ricorso per garantire il funzionamento di apparecchiature o strutture.
Il Parlamento europeo, in mancanza di un quadro giuridico chiaro, ha bocciato così, in seconda lettura, una proposta di direttiva che avrebbe rischiato di aprire la strada alle grandi multinazionali informatiche. Come il relatore di seconda lettura della proposta, Michel Rocard, ha annunciato poco prima della votazione, la reiezione della posizione comune del Consiglio è stata dovuta, non tanto alla convergenza di vedute dei deputati, quanto al fatto che i diversi schieramenti presenti in Parlamento preferiscono la reiezione all'adozione degli emendamenti presentati dai gruppi avversari [3].
Sul merito, il deputato ha sottolineato la necessità di approfondire il dibattito in quanto la normativa "non è matura". Egli afferma che "...la reiezione sarà un chiaro messaggio all'ufficio europeo dei brevetti: il Parlamento ha rifiutato di legalizzare le recenti derive volte ad ampliare il campo d'applicazione della brevettabilità a taluni software. Se queste derive dovessero continuare, è chiaro che una maggioranza parlamentare emergerebbe per opporvisi...."[3].
Un ruolo importate hanno avuto nel dibattito gli accordi del trattato TRIPS *Agreement on Trade-Related Aspects of Intellectual Property Rights*. Il riferimento è all'articolo 27, il quale prescrive che "brevetti dovranno essere concessi per qualsiasi invenzione, sia di prodotto che di processo, in tutti i campi della tecnologia, a condizione che essi siano nuovi, coinvolgano un salto inventivo, e siano capaci di applicazione industriale" [18].
Per Rochard [3] "...la formulazione che è apparsa come la più esaustiva e al contempo la più chiara per definire il campo della tecnologia è quella che definisce come tecnica una nuova istruzione sull'utilizzo delle forze controllabili della natura[7] sotto il controllo di un programma informatico e distinta dai mezzi tecnici necessari all'attuazione di tale programma (....) Aggiungiamo così che il trattamento, l'elaborazione, la rappresentazione e la presentazione dell'informazione mediante un programma informatico non sono tecnici, anche nel caso in cui a tal fine vengano impiegati dei mezzi tecnici".

9. CONCLUSIONI

I brevetti sono sempre stati applicabili all'implementazione fisica delle idee, secondo quanto descritto nei formulari degli uffici brevetti. In questo modo si divulga la conoscenza dell'innovazione con in cambio un monopolio temporale.
Quello che si brevetta non è la descrizione, ma la combinazione concreta delle componenti fisiche che costituiscono l'apparato, come figura nella descrizione e nella rivendicazione del brevetto. Tale combinazione di apparati, nel caso del software è sempre lo stesso: un computer o più computer [3].
Quando parliamo di software stiamo trattando con qualcosa di immateriale e intellettuale [3, 5, 6], la cui implementazione coincide con la descrizione stessa, se quest'ultima è sufficientemente precisa. Nel brevettare la descrizione, per tanto che sia imprecisa o di alto livello, quello che stiamo brevettando e monopolizzando è proprio la descrizione dell'idea, ovvero l'idea stessa e non l'implementazione della stessa, come è richiesto ad un brevetto.
A favore della brevettablità del software vengono di solito citate quattro necessità: la tutela dell'inventore, l'armonizzazione della legislazione europea, l'aumento di competitività e l'attrazione di capitali. Tali necessità sono evidenti. L'Europa è in ritardo rispetto ad altri mercati. Senza regole comuni e soprattutto semplici l'Europa non può guadagnare competitività ed attrarre capitali. E per attrarre capitali è anche necessario che la dimensione delle sue aziende cresca e quindi l'aggregazione deve essere la direzione in cui andare. E l'innovazione deve essere assolutamente favorita e tutelata.
La domanda da porsi dunque è se i brevetti software siano positivi rispetto alle quattro necessità citate.
L'impressione di chi scrive è che l'inventore, dopo aver investito in spese legali per avere il suo brevetto, dovrà sostenere ulteriori spese legali per difendersi dalle cause legali per

[7] Gravitazionale, elettromagnetica, interazione forte ed interazione debole

violazione di altri brevetti. In presenza di brevetti software, per sviluppare software in modo sicuro, senza rischiare di incorrere in infrazioni di brevetti, la *conditio sine qua non* consiste nel consultare il Database di tutti i brevetti software per assicurarsi di non violarne nessuno. Praticamente, ogni programma attualmente minimamente utile viola qualche brevetto software. A questo bisogna aggiungere che in certi casi non è possibile sviluppare una via alternativa.
I Lloyds di Londra hanno creato e propongono negli Stati Uniti una copertura assicurativa contro i rischi giuridici legati all'uso del software libero e aperto.
I brevetti europei oggi sono un'arma spuntata: non sono rivendicabili di fronte a nessuna giurisdizione nazionale, legata dalla legislazione attuale alla brevettabilità ai soli processi e prodotti che mettono in gioco trasformazioni di massa e di energia (le cosiddette "forze della natura").
Per fare un esempio, in Europa è oggi tecnicamente e giuridicamente infondato avanzare nei confronti di un'impresa di vendita di prodotti *on line* rivendicazioni (sostenute da brevetti) simili a quelle avanzate dalla stessa Amazon nei confronti di Barnes & Nobles (il famoso caso del brevetto relativo alla possibilità d'acquisto *one-click*) ed è impossibile ottenere (come invece ha ottenuto Amazon) un verdetto favorevole dalla giurisdizione europea.
La tranquillità e la certezza di poter lavorare al riparo dalle minacce giudiziarie (che possono manifestarsi in modo inaspettato e che possono costringere qualsiasi piccola impresa americana in qualsiasi momento all'arresto dell'attività prima dell'esito del processo) possono costituire, sempre nella opinione di chi scrive, per le SME europee una condizione abilitante all'innovazione.
La durata di 20 anni del brevetto viene a corrispondere a varie "generazioni" informatiche nel calendario accelerato dell'evoluzione del software.
In aree come l'Europa, il mercato della creazione di software è in mano quasi totalmente alle SME; si potrebbe creare un serio problema di competitività o mantenimento di tessuto industriale locale.
La condizione di libertà dai brevetti software dell'ecosistema informazionale dell'Unione Europea può invece forse costituire un vantaggio competitivo per l'Europa nella "guerra economica" globale.
Le imprese europee infatti possono sviluppare un'offerta di servizi software sul mercato mondiale, esente dal costo e dai rischi dei brevetti software, in competizione con l'offerta delle imprese nordamericane e delle altre zone economiche rilevanti; anche in India, che è allo stato attuale uno dei concorrenti potenziali più agguerriti su questo terreno, è passata una legislazione sui brevetti software simile a quella americana.
Per un'industria europea del software e dei servizi in rete, considerata un elemento essenziale dello sviluppo quantitativo e qualitativo dell'economia europea ed oggetto di tutte le attenzioni nel programma di Lisbona, è una prospettiva da considerare.
La questione é aperta. Si è conclusa a fine marzo 2006 la consultazione [19] sul "Brevetto Europeo", lanciata a metà gennaio 2006, dalla direzione *Internal Market and Services*, con lo scopo di semplificare la burocrazia associata al rilascio dei brevetti. L'attenzione potrebbe dunque spostarsi sul tema della brevettabilità in senso generale.

Bibliografia

[1] Bessen James, Maskin Eric: *Sequential Innovation, Patents, and Imitation*. Working paper Department of economic -MIT, 2000: http://www.researchoninnovation.org/patent.pdf

[2] DeBono E.: *New Think: The Use of Lateral Thinking in the Generation of New Ideas*. New York: Basic Books, 1967.

[3] Rochard M., European Parlament Committee on Legal Affaires: *Raccomandazioni per la seconda lettura* relativa alla posizione comune del Consiglio in vista dell'adozione della direttiva del Parlamento europeo e del Consiglio relativa alla brevettabilità delle invenzioni attuate per mezzo di elaboratori elettronici Commissione Giuridica, 21-06-2005, 2002/0047 (COD).

[4] Piva A., D'Agostini D: La tutela giuridica dei programmi per elaboratore. *Mondo Digitale*, n. 1, marzo 2003, p. 66-77.

[5] Barrionuevo Alberto: *Patentes de Software, situación tras el rechazo europeo*. II World Conference on Open Source. Malaga 2006, p. 5-13.

[5a] Meo A.R.: Software libero e Opensource. *Mondo Digitale*, n. 2, 2002, p. 3-16

[5b] Sissa G.: Open Source e pubblica amministrazione. *Mondo Digitale*, n. 7, settembre 2003, p. 54-62.

[6] Esteban Juan Antonio: *ATI, CEPIS Software patent working group*. Software Patentability on CEPIS national Membres, Aprile 2005, http://www.ati.es/DOCS/documentos/SW-patentabilitydiscussionpaperCEPISv200.pdf

[7] Mosaic: Proposal of Directive on Software Patents Rejected by The European Parliament. *UPGRADE*, Vol. VI, n. 3, June 2005, p. 59.

[8] Palmieri Nicola Walter: *I brevetti software sono contro la Costituzione europea*. Intelex–21.02.05, http://www.interlex.it/copyright/palmieri13.htm

[9] European Patent Office, http://www.european-patent-office.org

[10] Foundation for a Free Information Infrastructure, http://www.ffii.org/

[11] Sito di commercio elettronico Europeo brevettato, http://webshop.ffii.de/

[12] European Software Patent Horror Gallery, http://swpat.ffii.de/pikta/index.en.html

[13] http://patft.uspto.gov/netahtml/PTO/srchnum.htm

[14] Rubini A.: Il *problema dei brevetti sulle idee*. In: Aliprandi S., (a cura di) *Compendio di libertà informatica e cultura open*. Edizioni primaora, febbraio 2006, p. 103-112.

[15] *Punto informatico*. Eolas-Microsoft, punto-informatico.it

[16] Hart, Holmes: Reid -The Economic Impact of Patentability of Computer. Programs - Study Contract, http://europa.eu.int/comm/internal_market/indprop/comp/study_en.htm

[17] Rochard M., European ParlamentCommittee on Legal Affaires: *On the patentability of computer-controlled inventions*. Working Document (2002/0047 (COD)), 13-04-2005.

[18] Software patent under TRIPs agreements, Wikipedia, www.wikipedia.org

[19] European Commission Internal Market and Services DG. Questionnaire On the patent system in Europe, Brussels, 09/01/06, http://europa.eu.int/comm/internal_market/indprop/docs/patent/consult_en.pdf

[20] FFII Position Paper WGEPL/0601- The Consultation on Future Patent Policy in Europe, Marzo 2006, http://consultation.ffii.org/Downloads?action=AttachFile&do=get&target=wgeplo601-1.ob.pdf

GIOVANNA SISSA è esperta indipendente sui temi della Società dell'Informazione per la Commissione Europea, nell'ambito dei Programmi Quadro Ricerca, e del Ministero dell'Economia, nell'ambito del Quadro comunitario di Sostegno delle regioni Obiettivo 1. Ha partecipato ai lavori della "Commissione sul software a codice sorgente aperto nella PA" del Ministero Innovazione.
Consulente Ricerca e Sviluppo per industria e PA, fra gli anni '80 e ' 90 ha diretto lo sviluppo di sistemi esperti per Ansaldo Trasporti. Presso l'Università di Genova, dove è stata professore a contratto di Sistemi Basati su Conoscenza, dirige dal 2001 l'Osservatorio Tecnologico del MIUR, un servizio di trasferimento tecnologico dell'ICT alle scuole. Membro del gruppo italiano di definizione della certificazione EUCIP-IT Administrator. È autrice di numerosi libri ed articoli.
E-mail: giosissa@tin.it