OLIMPIADI DELL'INFORMATICA

Giovani talenti cercansi

Le Olimpiadi in Informatica, sponsorizzate dall'UNESCO e riservate agli studenti delle scuole superiori, rappresentano la più prestigiosa gara internazionale di questo genere. Il MIUR ha affidato all'AICA il compito di gestire l'organizzazione e la partecipazione italiana a tali gare a partire dal 2000, istituendo inoltre le Olimpiadi Italiane dal 2002. Questo articolo illustra il processo di selezione e i contenuti dei temi di gara, commentando i risultati delle gare e la preparazione in informatica richesta ai candidati.

Bruno Fadini
Roberto Grossi

1. COSA SONO LE IOI

Le Olimpiadi Internazionali di Informatica (**IOI**, *International Olympiad in Informatics*) hanno l'obiettivo di scoprire, incoraggiare, far incontrare, mettere in competizione e dare riconoscimento ai giovani che nel mondo mostrano maggior talento in informatica nonché di promuovere l'amicizia fra studenti e docenti di diversa cultura. La prima edizione delle IOI si è svolta nel 1989 in Bulgaria, su iniziativa e con il patrocinio dell'UNESCO (*United Nations Educational, Scientific and Cultural Organization*) e si sono svolte con cadenza annuale; l'ultima edizione ha avuto luogo in Polonia mentre quella del 2006 si svolgerà in Messico. Esse rappresentano la più prestigiosa gara internazionale di informatica riservata agli studenti delle scuole superiori: vedono la partecipazione di oltre 80 Paesi, ciascuno con una squadra di al più quattro studenti. Gli studenti gareggiano individualmente in due giorni distinti (con tre problemi di natura algoritmica e di *problem solving* da risolvere in ciascun giorno, entro cinque ore), trascorrendo ulteriori cinque giorni in attività socio-culturali. In coerenza con gli obiettivi della competizione, la premiazione avviene a fasce per cui la metà dei

Edizioni delle IOI

1. 1989 - Pravetz, Bulgaria, 16-19 maggio
2. 1990 - Minsk, Repubblica Bielorussa, Unione Sovietica, 15-21 luglio
3. 1991 - Atene, Grecia, 19-25 maggio
4. 1992 - Bonn, Germania, 11-21 luglio
5. 1993 - Mendoza, Argentina, 16-25 ottobre
6. 1994 - Haninge, Svezia, 3-10 luglio
7. 1995 - Eindhoven, Paesi Bassi, 26 giugno - 3 luglio
8. 1996 - Veszprém, Ungheria, 25 luglio - 2 agosto
9. 1997 - Cape Town, Repubblica del Sudafrica, 30 Novembre - 7 dicembre
10. 1998 - Setúbal, Portogallo, 5-12 settembre
11. 1999 - Antalya-Belek, Turchia, 9-16 ottobre
12. 2000 - Pechino, Cina, 23-30 settembre
13. 2001 - Tampere, Finlandia, 14-21 luglio
14. 2002 - Yong-In, Repubblica Coreana, 18-25 agosto
15. 2003 - Kenosha, Wisconsin, USA, 16-23 agosto
16. 2004 - Atene, Grecia, 11-18 settembre
17. 2005 - Nowy Sacz, Polonia, 18-25 agosto
18. 2006 - Mérida, Yucatán, Messico, 13-20 agosto

partecipanti ottiene una medaglia: precisamente, un dodicesimo dei partecipanti vince la medaglia d'oro, un sesto quella d'argento e un quarto la medaglia di bronzo. Il sito ufficiale delle IOI è www.ioinformatics.org.
L'Italia partecipa alle IOI a partire dal 2000 con il supporto scientifico ed economico da parte del MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca) e dell'AICA (Associazione Italiana per l'Informatica e il Calcolo Automatico) che ne fu la promotrice, riconoscendo l'alto valore educativo di tale iniziativa come strumento di sviluppo dell'informatica. L'obiettivo primario affianca, agli obiettivi generali delle IOI, quello di stimolare l'interesse dei nostri giovani verso gli aspetti scientifici dell'Informatica e di contribuire alla sua conoscenza e diffusione nelle scuole italiane.

2. LE OLIMPIADI DI INFORMATICA IN ITALIA

Nel 2000, nel contesto di un accordo di collaborazione MIUR-AICA che prevedeva anche lo sviluppo dell'ECDL e uno studio sui programmi di informatica nelle scuole, il MIUR ha affidato all'AICA il compito di gestire l'organizzazione e la partecipazione italiana alle gare internazionali delle IOI. La gestione avviene tramite il **Comitato Olimpico**, una struttura paritetica MIUR-AICA, che definisce l'organizzazione e la sovrintende e il Gruppo dei Selezionatori Nazionali (ovvero gli "allenatori"), che si occupa della formazione scientifica e, in collaborazione con il Comitato, del processo di selezione degli studenti per la gara internazionale. Dal 2002, poi, il Comitato Olimpico ha istituito le "OII, Olimpiadi italiane di informatica" che si svolgono in concomitanza con il processo di selezione.
La partecipazione in sole sei edizioni ha prodotto un discreto numero di medaglie: 1 oro, 4 argenti e 8 bronzi (si veda il riquadro a p. 11). Il cammino verso la selezione e la preparazione della squadra italiana si svolge durante tutto il corso dell'anno ed è organizzato in quattro fasi di selezione:
❑ **Scolastica**: vengono scelti un migliaio di studenti tra gli oltre 10.000 partecipanti.
❑ **Regionale**: vengono scelti 80 studenti tra il migliaio a disposizione.
❑ **Nazionale**: viene svolta la gara nazionale, denominata Olimpiadi Italiane di Informatica, individuando i vincitori nazionali e selezionando in totale un gruppo di 15 studenti chiamati "Probabili Olimpici".
❑ **Olimpica**: vengono scelti i 4 componenti della squadra italiana per partecipare alle IOI, più 2 riserve, tra i 15 Probabili Olimpici dopo un periodo di allenamento e formazione residenziale e telematica con i Selezionatori Nazionali.
Nel seguito descriviamo le tappe di un tale cammino per dare un'idea di come le selezioni avvengano in concreto. Immaginiamo che uno studente che abbia ricevuto in regalo un calcolatore...

Comitato Olimpico Italiano e Gruppo dei Selezionatori Nazionali

Presidente:
- Prof. Fadini - Università "Federico II" di Napoli

Esponenti AICA:
- Dott.ssa Genovié De Vita
- Prof. Casadei - Università di Bologna
- Prof. Italiani - Università di Milano
- Prof. Marchetti Spaccamela - Università "La Sapienza" di Roma

Esponenti MIUR:
- Dott.ssa Davoli - Dir. Gen. Ordinamenti Scolastici
- Preside Affronti - Liceo Classico V. Emanuele II - Palermo
- Preside Calabrò - ITIS Panella - Reggio Calabria
- Preside Di Rienzo - ITC Tosi - Busto Arsizio (VA)

Selezionatori Nazionali:
- Dott. Chierichetti - Università "La Sapienza" di Roma
- Prof. Grossi - Università di Pisa (coordinatore)
- Prof. Laura - Università "La Sapienza" di Roma
- Prof. Rizzi - Università di Udine
- Prof. Santini - Università di Milano

Leader e Deputy Leader alle IOI:
- Prof. Boldi - Università di Milano (IOI 2000–2003)
- Prof. Grossi - Università di Pisa (IOI 2004–2005)
- Prof. Rizzi - Università di Udine (IOI 2005)
- Prof. Vigna - Università di Milano (IOI 2000–2004)

3. PRIMA DELLE OLIMPIADI: DAL GIOCO ALLA PROGRAMMAZIONE

Il nostro amico studente inizia a utilizzare il calcolatore per attività ludiche e di svago. Successivamente, spinto dalla curiosità e dalla volontà di dominare la macchina (è lo stesso impulso che spingeva le vecchie generazioni a smontare i giocattoli di allora) ne cambia la configurazione, sviluppa esperimenti... Il passo per giungere alla decisione

di programmarlo è vicino; infatti, anche se non tutti gli appassionati di videogiochi vogliono programmare, molti desiderano crearne di nuovi e nasce così la prima spinta verso la programmazione... Lo studente si rivolge ad amici più esperti, naviga in rete o va in biblioteca alla ricerca di materiale adatto alle proprie conoscenze. L'impatto è duro: l'elettrodomestico-calcolatore si trasforma in una macchina infernale difficile da controllare e le conoscenze richieste sembrano immense. Molti studenti motivati riescono a superare questa fase e aspirano a confrontarsi con altre persone che condividono gli stessi interessi. Riuscire a mettere gli studenti di questa comunità invisibile in contatto tra loro è già un successo che ripaga gli organizzatori delle Olimpiadi Italiane.

4. IL PRIMO CONTATTO CON LE OLIMPIADI: LA FASE SCOLASTICA

C'è attività frenetica in molte scuole durante il mese di novembre per l'organizzazione della selezione scolastica. Il numero delle scuole iscritte alle Olimpiadi è in costante crescita e oggi supera le 500 con oltre 10.000 studenti degli ultimi 3 anni delle superiori. La partecipazione della scuola è spesso sollecitata dagli insegnati specifici di informatica, e in qualche caso anche dagli studenti. La gara viene gestita localmente dagli insegnanti che rivestono il ruolo di Referenti Scolastici. Il nostro studente deve affrontare una serie di semplici esercizi senza l'ausilio del calcolatore: quesiti logico-matematici e di programmazione (in Pascal e C/C++), a risposta libera o multipla. E qui occorre segnalare l'iniziale delusione del nostro studente autodidatta e sedicente esperto che si trova a competere su un campo che a lui non sembra quello dell'informatica da lui stesso inseguita (che c'entra la matematica? e i quiz di logica e poi...linguaggi obsoleti come il Pascal!). Ma il nostro studente è preparato e comunque supera brillantemente la prova, piazzandosi al primo posto nella scuola (adottando un criterio misto tra il rappresentativo e il meritocratico, sono ammessi alla successiva selezione regionale i primi 2 di ogni scuola e i rimanenti migliori in assoluto fino a coprire la capienza delle sedi di gara). Ha anche compreso (e lo comprenderà meglio durante lo sviluppo delle Olimpiadi) che la matematica e la logica sono componenti fondamentali dell'informatica, che Pascal e C++, oltre a essere i linguaggi ufficiali delle IOI, ben si prestano a esprimere problemi di natura algoritmica e che, soprattutto, l'informatica come disciplina scientifica non è lo smanettamento al calcolatore ma una disciplina ricca di teorie e metodologie per la risoluzione di problemi. Inoltre (e queste sono le motivazione del Comitato Olimpico), l'uso dei quesiti logico-matematici permette di individuare potenziali candidati che hanno buone possibilità di migliorare durante le fasi successive, anche se non già esperti in informatica (si veda il riquadro a p. 13).
Ovviamente, occorre sottolineare che, nonostante gli obiettivi di un'ampia partecipazione, la manifestazione è pur sempre una gara alla ricerca di talenti e che dunque lo studio è di fondamentale importanza, ma l'impegno a trovare motivazioni interiori per crescere scientificamente è altrettanto importante, ovviamente oltre al talento che diversi possiedono. È anche importante coinvolgere i Referenti Scolastici attraverso un forum di discussione che incoraggi a loro volta una crescita professionale. È al miglioramento di questo livello di interazione che il Comitato Olimpico punta e, per questo, ha supportato la creazione di un portale nel *www* per tutti coloro che sono interessati a questo tipo di attività (www.olimpiadi-informatica.it).

5. L'APPROFONDIMENTO: LA FASE REGIONALE

Gennaio è un mese critico per il nostro studente perché ha luogo la selezione regionale. Durante i mesi di novembre e dicembre ha dovuto impegnarsi in modo significativo per acquisire abilità programmative nettamente superiori a quelle necessarie per la precedente selezione. Adesso i problemi da affrontare hanno una natura più algoritmica e richiedono una certa dose di creatività per risolverli durante la gara, in modo da far emergere gli studenti che sono dotati di capacità algoritmiche elementari (vedi esempi a p. 14). È previsto l'uso del calcolatore (finalmente! pensa il nostro amico studente) per cui bisogna trovare il metodo di soluzione, programmarlo in

Pascal o C/C++, compilare, fare una prova di esecuzione... La competizione è diventata più impegnativa: il nostro studente probabilmente sente la necessità di un approfondimento culturale, scientifico e tecnologico. Vuole conoscere meglio l'architettura di un calcolatore, i sistemi operativi (ha sentito parlare di questo mitico Linux), i costrutti dei linguaggi di programmazione e i loro compilatori, gli algoritmi efficienti di risoluzione dei problemi computazionali. Fortunamente il World Wide Web permette l'accesso a diverse collezioni di esercizi e di altro materiale a disposizione in lingua inglese: è sufficiente immettere i termini *Olympiad Informatics* oppure *IOI tasks* in un motore di ricerca per ottenere vari risultati. Sul fronte interno, il citato portale fornisce alcuni forum di discussione, diversi riferimenti utili e ulteriore materiale didattico in italiano. Il Comitato Olimpico sta anche studiando i mezzi per sollecitare iniziative di formazione regionali, con l'ausilio delle direzioni scolastiche e delle università locali, già messe in atto da alcune regioni (Campania, Emilia-Romagna, Liguria, Lombardia, Trentino Alto-Adige).
Rispetto alla fase scolastica, anche la sede di gara è cambiata: quest'anno vi sono 23 sedi regionali dislocate sul territorio (per la numerosità delle scuole partecipanti, alcune regioni hanno due sedi di gara) e la gestione della gara viene affidata a insegnanti che rivestono il ruolo di Referenti Regionali. I testi dei problemi da risolvere arrivano presso il server locale delle sedi regionali attraverso un server centrale che li comunica in forma criptata, per cui neanche i Referenti ne conoscono il contenuto. Quando inizia la gara, viene resa pubblica la chiave crittografica e, al momento della consegna elettronica, il sistema di gara prevede che gli elaborati dei gareggianti siano nuovamente criptati con un'altra chiave segreta. In questo modo, la distribuzione sul territorio dei problemi di gara e la successiva raccolta degli elaborati possono avvenire in modo asincrono tra le varie sedi senza avvantaggiare chi ha potenti mezzi di comunicazione, nel rispetto delle ragionevoli regole di correttezza sportiva e di equità. Al contrario delle scolastiche, la correzione degli elaborati è centralizzata e automatica, con un sistema simile a quello descritto in seguito per la selezione nazionale.
Circa un migliaio di studenti partecipa a questa fase e soltanto 80 di essi passano alla fase successiva: in analogia alle scolastiche, i primi 2 di ogni sede e i rimanenti migliori in assoluto fino a coprire la capienza della sede di gara nazionale.

Prerequisiti scientifici per la partecipazione alle selezioni nazionali

- Ottima conoscenza di un linguaggio di programmazione tra Pascal, C e C++
- Tecniche di programmazione:
  - memoria dinamica (allocazione/deallocazione) e puntatori
  - array e aritmetica dei puntatori
  - gestione dei file di input/output (testo e binari)
  - strutture e tipi di dati
  - ricorsione (funzioni che invocano se stesse)
- Strutture di dati:
  - pile e code
  - liste concatenate
  - alberi binari (di ricerca o meno)
  - alberi generali
  - grafi e loro rappresentazione
- Algoritmi di base:
  - ordinamento (quicksort o mergesort)
  - ricerca binaria o dicotomica
  - visite di alberi (anticipata, simmetrica, posticipata)
  - visite di grafi (BFS, DFS).
- Cenni a tecniche algoritmiche:
  - divide et impera
  - greedy
  - programmazione dinamica

6. IL SALTO QUALITATIVO: LA FASE NAZIONALE

La fase nazionale del processo di selezione è anche, dal 2002, la gara per le "Olimpiadi italiane di Informatica". Useremo una metafora atletica per dare un'idea dell'impegno richiesto per lo studio e la risoluzione di problemi di difficoltà crescente in vista della preparazione per la fase nazionale ai primi di marzo, data in cui si svolgono le gare per le Olimpiadi Italiane di Informatica. Volendo correre i 100 metri, siamo tutti in grado di farlo in tempi più o meno ragionevoli. Pochi sanno farlo sotto i 15″ (gli studenti selezionati alle regionali), ma alle gate nazionali occorre scendere sotto i 12″ e alle internazionali anche sotto i 10″. Alle gare nazionali di informatica siamo di fronte a una parete a picco e molti non sanno scalarla, neanche impiegando un mese: alcuni dei problemi sono irrisolvibili se non abbiamo l'intuizione giusta oppure un profondo allenamento non disgiunto da quel pizzico di intuizione e talento. Le strategie standard di risoluzione dei problemi aiutano, ma dobbiamo scovarle per il caso specifico e dobbiamo associare a esse l'inventiva e la velocità per realizzare in codice quanto escogitato. Il nostro studente deve impiegare la massima disciplina e concentrazione per superare la prova e non perdere tempo.

La gara si svolge sulla falsariga delle gare internazionali delle IOI ma con problemi relativamente più semplici (ma comunque difficili!); vedi esempi a p. 14. Partecipano circa 80 studenti in un'unica sede di gara (quest'anno, al Politecnico di Milano), impiegando una rete di calcolatori con caratteristiche identiche ma con la possibilità di usare uno qualunque dei due sistemi operativi più diffusi (Linux e Windows). La novità dell'evento risiede nel fatto che gli studenti entrano a contatto diretto per la prima volta con i loro Selezionatori Nazionali, i quali gestiscono completamente il sistema di gara ed effettuano la valutazione degli elaborati ai fini della classifica finale. Il sistema di correzione utilizza Linux e i compilatori g++ e gcc (*GNU C/C++ Compiler*) e fpc (*Free Pascal Compiler*). Oltre al tempo limite di cinque ore per risolvere i tre problemi (come in una giornata di gara delle IOI), esiste un ulteriore limite sul tempo massimo di esecuzione, in genere di pochi secondi. Infatti, non solo le soluzioni proposte devono poter essere compilate senza errori nel sistema di correzione, ma devono terminare e restituire il risultato corretto *entro* il tempo limite. I concorrenti hanno la possibilità di abituarsi a questo sistema il pomeriggio precedente al giorno di gara.
I vincitori delle Olimpiadi Italiane ricevono le medaglie a fasce con una regola simile a quella per le IOI: 5 medaglie d'oro, 10 d'argento e 20 di bronzo. Inoltre, i primi 15 classificati (oro e argento) sono ammessi alla fase olimpica come Probabili Olimpici. La premiazione avviene in forma ufficiale alla presenza delle autorità, della stampa e della radio-televisione.

7. LA SQUADRA ITALIANA: LA FASE OLIMPICA

Lo *status* di Probabile Olimpico per il nostro studente comporta un impegno significativo, considerato che va integrato con i normali impegni scolastici. La fase olimpica dura circa tre mesi, fino a maggio, e prevede due corsi residenziali tenuti da docenti universitari e tutor (solitamente ex-olimpici) del Gruppo di Selezionatori Nazionali (ultimamente gli studenti e gli allenatori sono ospitati dall'Università di Pisa e dalla Scuola Normale Superiore). È un'esperienza di alto valore educativo, non solo dal lato scientifico, ma anche umano e sociale: in tale contesto, spesso si formano delle amicizie che proseguono ben oltre la normale attività. Il clima è molto collaborativo e i Selezionatori hanno la possibilità di conoscere meglio le attitudini dei Probabili Olimpici durante il periodo di formazione. Vengono discussi argomenti e tecniche di risoluzione più evolute, e sono affrontati insieme svariati problemi delle IOI così da entrare nel clima di gara. L'allenamento prosegue anche al di fuori dei due corsi residenziali, che solitamente durano circa una settimana l'uno e sono tenuti a ridosso delle vacanze di Pasqua e del "ponte" tra il 25 Aprile e il Primo Maggio. I Probabili Olimpici ricevono dei libri di testo specializzati su algoritmi e programmazione, e vengono incoraggiati a risolvere per proprio conto i problemi delle IOI precedenti in ambiente Linux usando i linguaggi C/C++ con il compilatore gcc/g++ (gli strumenti ufficiali per la valutazione degli elaborati alle IOI). Alla fine del secondo corso è prevista una gara con le stesse modalità delle nazionali ma con problemi più

Studenti che hanno vinto la medaglia d'oro alle Olimpiadi nazionali (OII)

OII 2003:
1. Stefano Maggiolo – ITI Severi di Padova
2. Stefano Soffia – Liceo Scientifico Aldo Moro di Reggio Emilia
3. Andrea Bergia – ITI G. Vallauri di Fossano (CN)
4. Samuel Kerschbaumer – ITI Max Valier di Bolzano
5. Andrea Matiacic – Liceo Scientifico France Preseren di Trieste
6. Nicola Pasta – ITI Majorana di Seriate (BG)
7. Dieter Steiner– ITI Max Valier di Bolzano
8. Matteo Brigo – ITI Carlo Zuccante di Venezia

OII 2004:
1. Alessandro Piva - Liceo Scientifico Majorana di Latina
2. Alessio Caiazza - ITI Antonio Meucci di Firenze
3. Luca Barbieri - Liceo Scientifico Donatelli Pascal di Milano
4. Alessio Guerrieri - Liceo Scientifico Leonardo da Vinci di Trento
5. Stefano De Rossi - Liceo Scientifico Galileo Galilei di Dolo (VE)
6. Paolo Giarrusso - Liceo Scientifico Galileo Galilei di Catania
7. Dario Zappalà - Liceo Scientifico Galileo Galilei di Catania
8. Marco Signorini - ITI Galileo Galilei di Livorno
9. Matthias Silbernagl - ITI Max Valzer di Bolzano

OII 2005:
1. Luca Barbieri - Liceo Scientifico Carducci di Milano
2. Giorgio Audrito - Liceo Scientifico M. Curie di Pinerolo (TO)
3. Andrea Canciani - Liceo Scientifico G. D. Cassini di Genova
4. Francesco Bonazzi - Liceo Scientifico Leonardo da Vinci di Milano
5. Marco Bonizzato - Liceo Scientifico Galileo Galilei di Verona

difficili: è sorprendente vedere come il livello scientifico di molti Probabili Olimpici evolva in pochi mesi!
La selezione della squadra dei quattro studenti, e delle due riserve, si basa quindi sui risultati della gara nazionale, della gara suddetta e di un colloquio che i candidati hanno con i membri del Comitato Olimpico e del Gruppo di Selezionatori Nazionali. I quattro della squadra ricevono in regalo un portatile per continuare ad allenarsi telematicamente con alcuni dei Selezionatori fino al momento della partenza per le IOI. I loro nomi vengono ufficialmente comunicati dal Comitato agli organizzatori delle IOI per l'inserimento nell'albo dei partecipanti.

8. LA PARTENZA: LA FASE INTERNAZIONALE

Per i componenti della squadra, l'attività di allenamento non si ferma a maggio ma prosegue telematicamente fino alla settimana di gara delle IOI, utilizzando il server italiano di gara del portale ufficiale delle Olimpiade Italiane e quello russo del *St. Petersburg Institute of Fine Mechanics and Optics* e partecipando a varie gare in linea che si tengono in altri Paesi (nel passato, si è anche partecipato ufficialmente alle "Olimpiadi europee" e si sono anche organizzati raduni di preparazione prima della partenza). Infine, l'appuntamento è all'aeroporto di Milano o di Roma per partire insieme alla volta della nazione che ospita la gara delle IOI, ove si ritrovano oltre 500 persone tra studenti, accompagnatori e guide locali. Gli accompagnatori sono coinvolti in una serie di incontri tecnici che portano alla definizione dei problemi di gara e in una serie di assemblee per deliberare su questioni gestionali del comitato internazionale. Gli studenti, oltre alle due giornate di gara, partecipano a gite, attività culturali e sociali (talvolta insieme agli accompagnatori). Il primo giorno c'è la cerimonia di apertura e poi una sessione simulata per fare pratica con il sistema di gara. Il secondo e il quarto giorno sono dedicati alla gara, il terzo e il quinto allo svago e alle gite, il sesto e ultimo alla cerimonia solenne di premiazione. Indipendentemente dal fatto che si riesca a ottenere una medaglia o meno, è appagante aver superato una serie di selezioni per essere annoverati tra i migliori giovani talenti informatici a livello mondiale.
A p. 15 alcuni esempi di esercizi dati alle IOI.

9. CONCLUSIONI: OLIMPIADI, GARE DI INFORMATICA E SCUOLE

Certamente, chi ama ascoltare musica, leggere libri o guardare film non è detto che voglia diventare musicista, scrittore o regista. Nel caso dell'informatica, la conclusione non è però così immediata: siamo sempre più sommersi dall'informazione e imparare a gestirla in modo appropriato è un dovere formativo degno quanto imparare a leggere e scrivere. Gli analfabeti del futuro saranno coloro che non sapranno organizzare e gestire selettivamente la marea di informazioni a cui siamo sottoposti. La conoscenza dell'informatica (e non solo) nelle scuole superiori è uno dei prerequisiti perché ciò avvenga, ricordandosi di non confondere lo strumento (il calcolatore e il *software*) con la scienza (l'informatica) come diceva Edsger Dijkstra, *premio Turing* 1972 (il premio *Nobel* per l'informatica): "*Computer science is no more about computers than astronomy is about telescopes*".
Qual è il rapporto fra questa premessa e le Olimpiadi di Informatica? Queste ultime, insieme alle altre Olimpiadi per le scuole superiori (di Astronomia, Biologia, Chimica, Fisica e Matematica), non costituiscono semplicemente un motivo di orgoglio della comunità scientifica internazionale ma rappresentano un'importante cartina di tornasole per valutare la diffusione della cultura scientifica nelle scuole superiori e quindi delle generazioni future. Lo stato dell'insegnamento dell'informatica nelle scuole superiori italiane non è certamente fra i migliori: a parte gli Istituti tecnici specialistici e alcuni licei sperimentali, l'insegnamento è assente né la recente riforma ha realizzato la svolta che si sperava: gli Istituti tecnici di Informatica e gli istituti sperimentali confluiranno nelle otto tipologie di liceo (dal classico al tecnologico)... ove l'informatica non c'è!

Ma se Sparta piange...anche negli Stati Uniti ci si lamenta della insufficiente preparazione nelle scuole. Un'intensa discussione ha avuto luogo infatti in diversi forum e riviste (*cfr. slashdot.org, CNET News.com e Communication of the ACM*), a partire dalla primavera del 2005, in merito all'andamento della squadra statunitense nelle gare ACM-ICPC (*ACM International Collegiate Programming Contest*) organizzate dall'*ACM*. Poiché queste gare, rivolte agli studenti universitari, sono per il resto analoghe a quelle delle IOI, molte delle osservazioni che stiamo per riportare si applicano anche a quest'ultime e ai problemi della formazione scolastica.

Le pessime prestazioni della squadra statunitense nelle ultime gare ACM-ICPC (le peggiori in 29 anni di gare) vengono commentate non per semplici motivi campanilistici, ma in quanto campanello d'allarme di una sempre minore competitività statunitense nel panorama mondiale dell'ICT. Tale percezione non è propria solo degli USA ma anche di molti altri paesi industrializzati come l'Italia che temono l'invasione scientifica e tecnologica dei paesi dell'Est europeo e asiatico. Il segnale d'allarme viene lanciato da James Foley, professore al *Georgia Institute of Technology* e da Craig Barret, presidente dell'*Intel Corporation*: oltre al collasso delle aziende basate sulla *Internet economy* e all'affidamento dello sviluppo di software a esperti fuori dal paese (*outsourcing*), disincentivando di fatto molti giovani a investire il loro futuro nell'ICT, ha pesantemente contribuito l'educazione scolastica, poco sensibile a formare capillarmente una cultura informatica a livello di scuola superiore. Un anonimo dirigente di un'importante azienda americana aggiunge che la preparazione scientifica degli studenti statunitensi è agli ultimi posti tra i Paesi industrializzati, come riportato dall'*Organization for Economic Cooperation and Development* (per gli addetti ai lavori, il famoso PISA - *Programme for International Student Assessment*). David Patterson, professore della *University of California* a Berkeley, fornisce ulteriori considerazioni in merito. Innanzi tutto, c'è una maggiore diffusione dell'ICT in alcuni Paesi in via di sviluppo in quanto non richiede enormi investimenti; inoltre, primeggiare in tali gare è una forma di orgoglio nazionale per cui si investe molto su pochi a scapito di molti (si pensi alle nuotatrici olimpiche della Germania dell'Est). A livello nazionale, diversi corsi di studio in informatica sono obsoleti (Patterson cita il corso di compilatori a Berkeley) e la ricerca scientifica non è finanziata sufficientemente: i continui tagli al finanziamento pubblico hanno tarpato le ali a molti laboratori di ricerca, fornendo implicitamente un messaggio negativo ai giovani. Patterson conclude quindi che un ridimensionamento è accettabile in quanto sono scomparsi diversi speculatori apparsi alla soglia dell'informatica durante il boom di Internet (chiaramente poco interessati alla disciplina in sè stessa): tuttavia, il timore di *outsourcing* ha eccessivamente spaventato le future generazioni di programmatori nazionali.

Nonostante qualche voce fuori del coro (per esempio, Norman Matloff, professore della *University of California* a Davis), ce n'è a sufficienza per riflettere sul panorama della preparazione degli studenti nelle scuole superiori: sul piano politico, partiamo in ritardo rispetto agli Stati Uniti e quindi potremmo non fare gli stessi errori...ma il non far niente non può che acuire il distacco se non verso gli Stati Uniti, almeno verso i Paesi dell'Est! Sul piano delle gare IOI, dobbiamo dichiararci molto soddisfatti per i risultati finora conseguiti, anche in assenza di una specifica preparazione scolastica, ma tutta basata sull'impegno e sul talento dei nostri studenti e sull'abnegazione della squadra degli allenatori.

Ringraziamenti

Si ringrazia la dott.ssa Margherita Ortolani per il supporto organizzativo fornito durante la scrittura di questo articolo, il prof. Romeo Rizzi per aver collaborato alla traduzione dei testi riportati nella scheda di p. 15 e i vari colleghi che hanno contribuito alla stesura dei testi riportati nelle schede di p. 13 e 14. Inoltre, siamo riconoscenti al prof. Paolo Boldi, all'Ing. Rosario Cristaldi, al dott. Luca Foschini, al prof. Sebastiano Vigna e a tutti gli ex-olimpici,

per i loro significativi contributi nella creazione del sito Web per gli allenamenti.

Bibliografia

[1] Cooper C.: *Can the U.S. still compete*? CNET News.com, Apr. 2005 (http://news.com.com/Can+the+U.S.+still+compete/2010-1071_3-5672106.html)

[2] Frauenheim E.: *U.S. slips lower in coding contest*. CNET News.com, Apr. 2005 (http://news.com.com/U.S.+slips+lower+in+coding+contest/2100-1022_3-5659116.html)

[3] Frauenheim E.: *Can Johnny still program*? CNET News.com, Apr. 2005 (http://news.com.com/Can+Johnny+still+program/2008-1036_3-5675770.html)

[4] Matloff N.: Globalization and the American IT worker. *Commun. ACM*, Vol. 47, n. 11 (Nov. 2004), p. 27-29 (http://doi.acm.org/10.1145/1029496.1029516).

[5] Matloff N.: *Johnny can so program*. CNET News.com, May 2005 (http://news.com.com/Johnny+can+so+program/2010-1007_3-5700858.html)

[6] Patterson D.A.: Reflections on a programming Olympiad. *Commun. ACM*, Vol. 48, n. 7, (Jul. 2005), p. 15-16. (http://doi.acm.org/10.1145/1070838.1070852)

[7] Patterson D.A.: Restoring the popularity of computer science. *Commun. ACM*, Vol. 48,n. 9 (Sep. 2005), p. 25-28 (http://doi.acm.org/10.1145/1081992.1082011).

[8] Skiena S.S., Revilla M.: *Programming Challenges*. Springer, NY 2003 (ISBN: 0387001638).

Bruno Fadini è professore ordinario di Calcolatori elettronici presso l'Università Federico II di Napoli dal 1973 e presidente del Comitato Olimpico italiano per le Olimpiadi di Informatica. Già presidente dell'AICA, ha diretto o partecipato a molte attività di ricerca nazionali ed internazionali. I suoi interessi scientifici e didattici sono oggi rivolti all'architettura dei sistemi di elaborazione ed ai processi di e-learning. È autore di oltre 100 pubblicazioni scientifiche e di 10 libri.
E-mail: fadini@unina.it

Roberto Grossi è professore associato di Informatica presso l'Università di Pisa dal 1998 e allenatore della nazionale italiana per le Olimpiadi Internazionali di Informatica. I suoi interessi scientifici e didattici sono rivolti al progetto, all'analisi e alla sperimentazione degli algoritmi e delle strutture di dati. Ha visitato diversi centri di ricerca internazionali (dagli Stati Uniti al Giappone), pubblicando oltre 60 lavori scientifici e un brevetto.
E-mail: grossi@di.unipi.it

Olimpici che hanno vinto una medaglia IOI

Gilberto Abram, *del Liceo Scientifico di Cles – TN (argento, IOI 2003). Ho deciso di compiere i miei studi universitari all'estero e frequento i corsi di Electrical Engineering and Computer Science presso il Massachusetts Institute of Technology, Cambridge, MA, USA. Inoltre lavoro saltuariamente nel gruppo di ricerca di ottica ed elettronica quantistica del prof. Franz Kaertner, a partire dall'estate 2005. Ho iniziato a programmare quando frequentavo le scuole elementari, usando il linguaggio Basic per il Commodore 64. Il primo programma impegnativo che ho scritto è stato per realizzare un motore grafico tri-dimensionale nel sistema operativo DOS che non ho portato a termine, cosa che ho fatto per un interprete del linguaggio Scheme nell'ambito del corso di programmazione. Visto che i miei voti alle superiori non erano bellissimi, a parte quelli in alcune materie, volevo dimostrare di essere in gamba almeno in qualcosa come programmare e le olimpiadi erano l'opportunità che cercavo. Poi a partire dalla fase regionale ho iniziato ad apprezzare gli algoritmi.*

Alessandro Arzilli*, dell'ITI Sarocchi di Siena (bronzo, IOI 2000). Sono appassionato di informatica e, per continuare a occuparmene, fingo di frequentare il Corso di Laurea in Informatica all'Università di Pisa. Ho iniziato a programmare a circa 14 anni usando il Pascal come linguaggio e, francamente, non ricordo il mio primo programma impegnativo. Traggo soddisfazione dalla mia passione per l'informatica: decisi di partecipare alle olimpiadi perché la mia insegnante di informatica, notando tale passione, insistette che mi iscrivessi.*

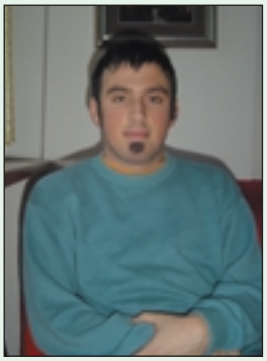
Giorgio Audrito*, del Liceo Scientifico Curie di Pinerolo – TO (bronzo, IOI 2004 e 2005). Studio Matematica all'Università di Torino e, per il momento, ho potuto vivere di rendita negli esami di informatica per la preparazione ricevuta durante gli allenamenti di Pisa. Oltre all'università mi dedico alla musica, in particolare al pianoforte e alla composizione: l'anno passato ho scritto la colonna sonora per due film muti del museo del cinema di Torino. Quest'anno sto seguendo anche corsi di organo e di violino e, durante la scorsa dura estate, ho dovuto anche dare l'esame dell'8° anno di pianoforte subito dopo le olimpiadi! Ho iniziato a programmare quando frequentavo le scuole elementari, scrivendo in Pascal qualche semplice giochino con interfaccia testuale, ma ho smesso per poi iniziare nuovamente durante gli anni del liceo. Per le olimpiadi ho imparato a usare il linguaggio C, scrivendo codice di qualche "utilità" per me; in Pascal ho sviluppato giochini e programmi di grafica (per quanto si possa fare con 16 colori). Forse il primo (e unico?) programma impegnativo è una unit in Pascal contenente una serie di oggetti astratti per strutturare un programma a eventi. Ho partecipato alle olimpiadi perché mi sono sempre piaciuti i problemi logici in informatica e matematica: quando frequentavo il secondo anno del liceo e non potevo ancora gareggiare, mi sono fatto dare i testi della fase scolastica per provare a svolgerli!*

Luca Barbieri, *del Liceo Scientifico Carducci di Milano (oro, IOI 2004, e argento, IOI 2005). Frequento il primo anno del corso di Matematica presso la Scuola Normale Superiore di Pisa. Ho iniziato a programmare quando frequentavo la scuola elementare, usando il linguaggio Basic (ehm). Non ricordo esattamente quale fosse, ma uno dei primi programmi che avessero una qualche utilità svolgeva automaticamente le espressioni date per il compito di matematica, includendo tutti i passaggi intermedi (ma non dirò mai i nomi dei miei compagni di scuola che ne beneficiarono). Sono venuto a conoscenza delle olimpiadi internazionali scientifiche attraverso le Olimpiadi della Matematica, a cui la mia scuola aderiva. Incoraggiato dai successi in tale ambito e dalla mia passione preesistente per l'informatica, ho deciso di provare a sostenere anche le Olimpiadi di Informatica [ndr: Luca ha vinto delle medaglie anche in altre olimpiadi].*

Matteo Bruni*, dell'ITI Montani di Fermo - AP (bronzo, IOI 2003). Frequento l'Università di Bologna, corso di Ingegneria Informatica, e sono al terzo anno. Ho iniziato a programmare verso i 10 anni, poi un po' più seriamente a 14 anni, adoperando il linguaggio GWBASIC e passando successivamente al Visual Basic e quindi al linguaggio C. Probabilmente il primo programma utile che ho scritto era quello per trovare il modo di riempire ottimamente il contenuto di un CD: avendo un insieme di file la cui dimensione complessiva superava la capacità del CD, dovevo prenderne un sotto-insieme per massimizzare lo spazio occupato nel CD. Quando mi sono messo a farlo mi sembrava una sciocchezza, però mi sono smentito subito. All'epoca non sapevo della complessità del problema, una variante del problema dello zaino (problema notoriamente NP-completo)!*

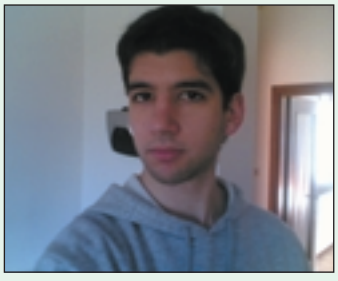
Dario Cazzaro*, dell'ITI Zuccante di Venezia (bronzo, IOI 2004). Sono iscritto al secondo anno di Ingegneria Informatica all'Università di Pisa e sono Allievo Ordinario presso la Scuola Superiore S. Anna di Pisa. Ho iniziato a programmare con il Commodore 64 verso i 10-11 anni con il Basic (mentre aspettavo il lunghissimo tempo di caricamento dei giochi) e poi ho continuato quest'attività iscrivendomi a un istituto tecnico. Grazie anche alla scuola, mi sono appassionato all'informatica scrivendo un po' di programmi (giochini, chat e così via) per*

trascorrere le lunghe ore nei laboratori con i miei compagni. Il primo programma importante è stato, probabilmente, quello per la mia tesina di esame di stato: un gioco (a più giocatori e via rete) uguale a Risiko che permetteva di far giocare autonomamente anche i calcolatori. Partecipare alle olimpiadi era quasi una tradizione nella mia scuola e, chiaramente, partecipare alle nazionali una meta ambita. Che dire? Il gusto per la sfida era molto alto e, abbinato a quel tipo di informatica per nulla "meccanica" ma che ti constringe a ragionare, mi ha invogliato a impegnarmi seriamente per raggiungere dei buoni risultati a livello internazionale.



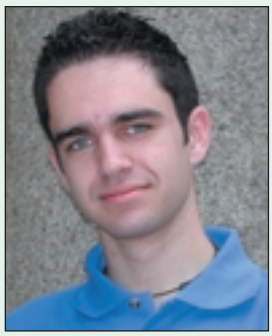
***Alessio Guerrieri**, del Liceo Scientifico da Vinci di Trento (argento, IOI 2005). Studio Informatica presso l'Università di Trento. I miei primi linguaggi di programmazione sono stati ToolBook (se vale, è un sistema di e-learning), il Pascal (usato per una settimana) e Visual Basic (adoperato tanto): il primo programma impegnativo è stato il videogioco di Dragonball, scritto appunto in Visual Basic. Ho deciso di partecipare alle olimpiadi inizialmente per saltare scuola (ehm), poi per interesse (giuro!).*

***Alessandro Maconi**, dell'ITC Greppi di Monticello Brianza – LC (bronzo, IOI 2001). Studio Ingegneria Elettronica presso il Politecnico di Milano e attualmente frequento il primo anno della laurea specialistica di tale corso. Ho iniziato a programmare a dodici anni, nel tentativo di imitare mio cugino più grande che programmava in Basic, che è quindi diventato il mio primo linguaggio. Ho sviluppato il primo programma impegnativo a quattordici anni: un tentativo di realizzare una specie di derive molto rozzo per disegnare una funzione. Potevo scegliere fra tre tipi di funzione: retta, parabola, iperbole equilatera. Effettuata tale scelta, dovevo indicare il valore dei parametri, l'intervallo di valutazione della funzione e il colore con cui volevo disegnarla: il programma disegnava gli assi cartesiani e tracciava la funzione scelta. Ho partecipato alle olimpiadi perché avevo visto un manifesto, chiedendo così al mio professore di informatica di far partecipare la scuola. Il mio sogno era di riuscire ad arrivare alla fase mondiale in Finlandia (nel 2001 ero in quarta) o in Corea (nel 2002 sarei stato in quinta).*



***Stefano Maggiolo**, dell'ITI Severi di Padova (bronzo, IOI 2002). Studio Matematica all'Università di Ferrara e ho iniziato a programmare a 9-10 anni, credo, usando il linguaggio qbasic. Il primo programma impegnativo (o almeno, il primo di cui non vergognarsi) è stato una specie di assembler. Perché ho partecipato alle olimpiadi... se permettete, che domanda è? E perché no? Per accrescere il mio ego – anzi, perché ci obbligò il professore di informatica (fino alla fase regionale non sapevo che esistessero anche le gare internazionali): va bene, lo confesso, voglia di agonismo (bla bla bla), possibilità di andare in giro per l'Italia e nel mondo (bla bla bla). Insomma, fatelo anche voi!*

***Giuseppe Ottaviano**, del Liceo Scientifico Fermi di Ragusa (argento, IOI 2003). Frequento il terzo anno del corso di laurea in Matematica dell'Università di Pisa e collaboro con un'azienda straniera specializzata nei motori di ricerca. Ho iniziato a programmare a 13-14 anni utilizzando il linguaggio Visual Basic (ehm), ma presto mi sono spostato sui linguaggi C e C++. Non ho mai scritto da zero un programma veramente impegnativo (a parte le soluzioni per gli esercizi delle olimpiadi), al massimo ho scritto qualche patch per programmi di altri. Il programma che mi ha richiesto più tempo è stato, recentemente, un programma giocattolo per simulare un sistema di molle attraverso la risoluzione di equazioni differenziali con delle tecniche viste a lezione. Ho deciso di partecipare alle olimpiadi perché mi ha sempre affascinato l'informatica teorica e la realizzazione in codice degli algoritmi. Ho cominciato per gioco e non mi aspettavo di arrivare neanche alle nazionali: volevo soltanto misurarmi con gli altri ragazzi che condividevano i miei stessi interessi per vedere se tutto il tempo che avevo perso a programmare era servito a qualcosa!*



***Alessandro Piva**, del Liceo Scientifico Majorana di Latina (bronzo, IOI 2004). Studio Informatica all'Università "La Sapienza" di Roma e, una volta presa la laurea triennale, spero di poter andare a studiare a Pisa. Ho iniziato a programmare quando frequentavo le scuole elementari, utilizzando un Commodore 64 e il Basic: era molto interessante, anche se ci metteva 10 minuti a caricare un programma da una cassetta! Poi è venuta l'era del linguaggio Pascal, e così via: il mio primo programmma impegnativo è stato probabilmente un gioco che ho realizzato (o almeno, ho iniziato) quando frequentavo le scuole medie. Raffigurava una pallina con occhi e bocca (molto facile da disegnare) che si muoveva in un piccolo bosco. La visuale era isometrica ed era scritto completamente in Pascal. Molto più ambizioso, sempre del medesimo periodo, un programma per visualizzare forme in tre dimensioni... sempre in Pascal! Le immagini che produceva erano a dir poco grottesche, con cubi che sembravano piramidi, ma erano pur sempre tridimensionali! Ho deciso di partecipare alle olimpiadi per le incredibili opportunità che mi offrivano: conoscere ragazzi appassionati come me, fare dei corsi intensivi con gli allenatori e, sopratutto, fare tanti bei viaggi! La cosa più importante che le olimpiadi mi hanno lasciato, comunque, è ben altra: è cambiato completamente il mio atteggiamento verso i progetti e gli impegni che mi assumo. Ho visto che posso ottenere buoni risultati se mi impegno a fondo, quindi adesso affronto le cose con più autostima e con meno timore. Senza contare, poi, che sono venuto a contatto con persone estremamente capaci e simpatiche, con cui continuo a tenermi in contatto!*

Esempi di esercizi dati alle selezioni scolastiche

Logico-matematico: Il direttore di un ristorante con capienza massima di 150 posti non ricorda quante erano le persone da lui servite in occasione dello scorso cenone di fine anno. Ricorda però che volendo sistemare tutte le persone servite in tavoli da 3 ne restava fuori esattamente una; inoltre, la stessa cosa succedeva sistemando tutte le persone in tavoli da 5 o tutte in tavoli da 7. Quante erano le persone servite in occasione dello scorso cenone di fine anno?

Schema di risoluzione: Erano $n = 106$ persone poiché $n \leq 150$ e $n - 1$ deve essere divisibile per 3, 5 e 7; essendo quest'ultimi dei numeri primi, l'unico multiplo che soddisfa le condizioni è $n - 1 = 3 \cdot 5 \cdot 7 = 105$.

Programmativo (*Pascal*): Data la seguente funzione ricorsiva

```
function mistero(m : integer; n : integer) : integer;
begin
if ( m = 0 ) then
   mistero := n
else if ( n=0 ) then
   mistero := mistero( m-1, 1 )
else
   mistero :=  mistero( mistero( m-1, n-1 ), n-1 )
end;
```

calcolare quale tra le seguenti risposte corrisponde ai valori restituiti invocando

```
writeln(mistero(0,3), ' ', mistero(1,3), ' ', mistero(2,3), ' ', mistero(3,3));
```

Programmativo (C/C++): Data la seguente funzione ricorsiva

```
int mistero(int m, int n)
 {if ( m == 0 )
   return n;
  else if ( n==0 )
   return mistero( m-1, 1 );
  else
   return mistero( mistero( m-1, n-1 ), n-1 );}
```

calcolare quale tra le seguenti risposte corrisponde ai valori restituiti invocando

```
printf("%d %d %d %d\n", mistero(0,3), mistero(1,3), mistero(2,3), mistero(3,3)):
```

a. 3 1 2 0
b. 3 2 1 0
c. 0 1 2 3
d. Nessuna delle risposte precedenti

Schema di risoluzione: Mantenendo traccia delle chiamate ricorsive attraverso un albero delle chiamate ricorsive e dei corrispondenti parametri attuali, risulta che la risposta corretta è la "b".

Esempi di esercizi dati alle selezioni regionali

Descrizione (livello di difficoltà D=2): Il nobile chimico Alfredo produce nel suo laboratorio due sostanze liquide potenzialmente inquinanti: l'*Aminozalina* e il *Brinofulo*. A fine giornata le deve smaltire in appositi contenitori, dislocati lungo il tragitto che parte dal laboratorio e arriva alla sua abitazione. Per limitare le possibilità d'inquinamento, Alfredo deve distribuire l'Aminozalina nel *maggior* numero possibile di contenitori mentre deve dividere il Brinofulo nel *minor* numero possibile di contenitori. Tuttavia Aminozalina e Brinofulo non possono essere assolutamente mescolati nel medesimo contenitore, altrimenti la loro miscela esplode. Ogni volta che raggiunge un contenitore per lo smaltimento dei liquidi, Alfredo deve eseguire una sola delle tre seguenti azioni: (i) versare Aminozalina fino al riempimento del contenitore; (ii) versare Brinofulo fino al riempimento del contenitore; (iii) non versare nulla nel contenitore. Data la quantità *A* di litri di Aminozalina e la quantità *B* di litri di Brinofulo da smaltire, e conoscendo l'elenco degli *N* contenitori (con rispettiva capacità) nell'ordine secondo cui sono incontrati lungo il tragitto dal laboratorio alla sua abitazione, aiutiamo Alfredo a decidere se e quale sostanza versare in ciascun contenitore.

Dati di input: Il file input.txt contiene nella prima riga gli interi *A* e *B* (rispettivamente i litri di Aminozalina e di Brinofulo da smaltire) e il numero *N* di contenitori disponibili. Tali valori sono separati da uno spazio. Nelle successive N righe (usando una riga per ogni contenitore) è contenuto un numero per riga: tali numeri rappresentano le capacità dei singoli contenitori elencati nell'ordine in cui vengono incontrati da Alfredo.

Dati di output: Il file output.txt deve contenere *N* righe, una per ogni contenitore. Ogni riga contiene due numeri separati da uno spazio, rispettivamente il numero di litri di Aminozalina e di Brinofulo smaltiti nel corrispondente contenitore. Notiamo che ogni riga deve contenere uno zero nei casi (i) e (ii) descritti sopra, e due zeri nel caso (iii).

Schema di risoluzione: Ordiniamo prima tutti i contenitori in base alla loro capacità, in ordine crescente (prima i contenitori meno capaci e poi quelli più capaci). Esaminando la sequenza in tal modo ordinata di contenitori, decidiamo di riempire i primi (quelli meno capaci) con l'Aminozalina fino a smaltirne *A* litri, così da usare il maggior numero possibile di contenitori. Per il Brinofulo, decidiamo di riempire partendo dagli ultimi contenitori fino a smaltirne *B* litri, così da usare il minor numero possibile di contenitori. Tale approccio goloso (*greedy*) funziona per le assunzioni del problema non riportate per brevità.

Esempi di esercizi dati alle selezioni nazionali

Descrizione (livello di difficoltà D=3 e tempo limite di 10 secondi): A Camelot, nel periodo di maggior splendore, ogni amicizia è corrisposta. Ogni cavaliere è amico di oltre la metà dei suoi compagni d'arme, ossia, se *N* è il numero dei cavalieri, ciascuno di essi può contare su almeno $N/2$ validi amici. La tavola rotonda è disposta in un enorme salone, circondata da *N* sedie, con una sedia per cavaliere. Com'è facilmente immaginabile, ogni cavaliere pretende di avere amici ai suoi due lati, altrimenti rifiuta di sedersi al tavolo. Tutti contano su Mago Merlino per disporre i cavalieri intorno al tavolo nella soddisfazione generale di tutti. Ma il poverino ha perso la formula magica per ottenere ciò in men che non si dica! Aiutiamo Merlino, anche con soluzioni parziali, a disporre i cavalieri intorno al tavolo in modo che ogni cavaliere abbia degli amici ai suoi lati.

Dati di input: Il file input.txt contiene nella prima riga l'intero *N*, il numero dei cavalieri della tavola rotonda. I cavalieri sono numerati da 1 a *N*. Ognuna delle successive *N* righe contiene una sequenza di *N* valori o oppure 1, separati da uno spazio. La sequenza contenuta nell'*i*-esima di tali righe rappresenta le relazioni di amicizia del cavaliere numero *i*. In particolare, il *j*-esimo valore in tale riga indica se i cavalieri *i* e *j* sono amici (valore = 1) o meno (valore = 0). Poiché l'amicizia è sempre corrisposta, se il cavaliere *i* è amico del cavaliere *j* allora il cavaliere *j* è amico del cavaliere *i*.

Dati di output: Il file output.txt deve contenere una sola riga contenente una sequenza di *N* numeri separati da uno spazio, per rappresentare la disposizione dei cavalieri attorno alla tavola rotonda. Il primo numero è sempre 1, a rappresentare il cavaliere numero 1. Il resto della sequenza è una permutazione dei numeri 2, 3, ..., *N* (cioè una disposizione in un qualche ordine, senza ripetizioni). Per ogni coppia *i* e *j* di numeri adiacenti nell'intera sequenza, i cavalieri *i* e *j* devono essere amici. Inoltre, il cavaliere alla fine della sequenza deve essere amico del cavaliere numero 1, in quanto vicini di posto.

Schema di risoluzione: Rappresentiamo il problema come un grafo *G* i cui nodi sono i cavalieri e i cui archi collegano due nodi se i corrispettivi cavalieri sono amici. Il problema di disporre i cavalieri attorno alla tavola rotonda equivale a trovare un *ciclo hamiltoniano* nel grafo *G*, ovvero un cammino che attraversa tutti i nodi una e una sola volta e che si ricongiunge al nodo di partenza. In generale, non conosciamo un algoritmo efficiente per risolvere tale problema (tecnicamente, la sua versione decisionale è *NP-completa*): fortunatamente possiamo ottenerlo nel nostro caso sfruttando l'importante ipotesi del problema, quella per cui ciascun nodo in *G* ha almeno $N/2$ nodi adiacenti (tecnicamente, il *grado* è almeno $N/2$). A tale scopo, troviamo un qualunque cammino massimale *P* che non passa due volte attraverso uno stesso nodo e indichiamo i suoi due nodi estremi con *x* e *y* (tale cammino attraversa un numero di archi che è pari almeno al grado dei suoi nodi, ovvero almeno $N/2$). Per l'ipotesi sul grado dei nodi, possiamo sempre trovare due archi che collegano *x* e *y* a due nodi adiacenti (siano essi *w* e *z*) all'interno del cammino *P*: tali archi (x, z) e (w, y), incrociandosi, creano un ciclo denominato *C* ottenuto eliminando l'arco (w, z) da *P* e aggiungendo (x, z) e (w, y). Notiamo che deve esistere un nodo *u* in *C* che è collegato a un nodo non appartenente a *C* (quest'ultimo è infatti più lungo di $N/2$ e tutti i nodi hanno grado almeno $N/2$). Rompiamo il ciclo *C* in corrispondenza di *u* per ottenere un cammino che sicuramente estendiamo a partire da *u*: ripetiamo più volte i passi precedenti di estensione del cammino/ciclo corrente fino a toccare tutti i vertici una e una sola volta (cioè il ciclo diventa hamiltoniano). Da notare che tale soluzione termina velocemente perché aggiunge almeno un nodo al cammino corrente per ogni iterazione, ma non si applica al caso generale perché non è detto che valga l'ipotesi sul grado dei nodi.

Esempi di esercizi dati alle gare internazionali

Descrizione del primo problema (*tempo limite di 0.5 secondi*): Byteman ha il più bel giardino in Bytetown. Nel giardino ha piantato n rose. È arrivata l'estate e i fiori sono cresciuti grandi e belli. Byteman si è reso conto di non poter curare tutte le rose da solo e ha deciso di assumere due giardinieri per aiutarlo. Byteman vuole selezionare due aree rettangolari in modo che ciascun giardiniere possa curare le rose all'interno di ciascun rettangolo. I rettangoli devono essere disgiunti e contenere esattamente k rose ciascuno. Byteman intende recintare il perimetro dei due rettangoli ma, essendo a corto di denaro, vuole impiegare meno recinto possibile. Aiutate Byteman a individuare i due rettangoli: il giardino di Byteman, un rettangolo lungo l metri e largo w metri, è diviso in $l \cdot w$ quadratini di lato un metro. Fissate un sistema di coordinate con gli assi paralleli ai lati del giardino. Ogni quadratino è individuato da una coppia di interi (x, y) con $1 \le x \le l$ e $1 \le y \le w$. I quadratini possono contenere anche più di una rosa. Selezionate i due rettangoli con lati paralleli a quelli del giardino e con vertici a coordinate intere, notando che un rettangolo con angoli nei quadratini $(l_1; w_1)$, $(l_1; w_2)$, $(l_2; w_1)$ e $(l_2; w_2)$ contiene tutti i quadratini (x, y) con $l_1 \le x \le l_2$ e $w_1 \le y \le w_2$ e ha perimetro di lunghezza $2 \cdot (l_2 - l_1 + 1) + 2 \cdot (w_2 - w_1 + 1)$. I due rettangoli debbono essere disgiunti, ossia non possono avere quadratini in comune. Qualora abbiano parti di perimetro in comune, essi dovranno avere comunque recinzioni separate. Scrivete un programma che legge da standard input le dimensioni del giardino, il numero delle rose nel giardino, il numero delle rose da riporre in ciascuno dei due rettangoli, e le posizioni delle rose; trova gli angoli di due rettangoli la cui somma dei perimetri è minima, sotto le condizioni di cui sopra; scrive su standard output il minimo valore per la somma dei perimetri di due rettangoli disgiunti contenenti ciascuno precisamente k rose, (oppure la singola parola NO, ove una tale coppia di rettangoli non esista).

Dati di input: La prima riga del file input.txt contiene due interi separati da un singolo spazio, ossia la lunghezza l e la larghezza w del giardino ($1 \le l, w \le 250$). La seconda riga contiene due interi separati da un singolo spazio: il numero n di rose nel giardino e il numero k di rose che dovrebbe essere racchiuso in ciascun rettangolo ($2 \le n \le 5000$, $1 \le k \le n/2$). Le n righe seguenti contengono le coordinate delle rose, una rosa per ogni riga: la riga $(i + 2)$ riporta due interi l_i e w_i ($1 \le l_i \le l$, $1 \le w_i \le w$), separati da un singolo spazio, ovvero le coordinate del quadratino contenente la rosa i-esima. Due o più rose possono riproporsi nello stesso quadratino.

Dati di output: Il file output.txt deve contenere un'unica riga, contenente solamente un intero che rappresenta il valore minimo per la somma dei perimetri di due rettangoli disgiunti contenenti ciascuno precisamente k rose, oppure una singola parola NO, nel caso una tale coppia di rettangoli non esista.

Schema di risoluzione: Ipotizziamo che sia $w = l$ per semplicità, ottenendo così un giardino quadrato di lato w. Indichiamo con il termine k-rettangolo un rettangolo interno al giardino che contiene k rose. Essendoci all'incirca w^4 possibili rettangoli da considerare nel giardino, la soluzione che seleziona i k-rettangoli tra questi ultimi e poi li esamina a coppie per trovarne una con perimetro minimo, riesce a risolvere solo un piccolo numero di casi nel tempo limite assegnato al problema (in quanto il numero di passi computazionali è molto elevato, proporzionale a circa w^8 operazioni di calcolo). Bisogna quindi procedere in altro modo, evitando di calcolare esplicitamente tali valori. Forniamo due idee che aiutano a risolvere diversi casi (laddove occorre raffinarle, per risolverli tutti, con ulteriori tecniche algorithmiche, pervenendo così a una soluzione che richiede all'incirca w^3 operazioni di calcolo). La prima idea permette di calcolare il numero di rose di un qualunque rettangolo in un numero costante di operazioni, indipendentemente dalla dimensione del rettangolo, dopo un'opportuna pre-elaborazione che richiede circa w^2 operazioni. Quest'ultima considera in modo incrementale soltanto i rettangoli "prefissi" $P(x, y)$ tali che hanno un angolo nel quadratino $(1, 1)$ e quello opposto in (x, y): dato un qualunque rettangolo con angoli nei quadratini $(l_1; w_1)$, $(l_1; w_2)$, $(l_2; w_1)$ e $(l_2; w_2)$, il numero di rose in esso contenuto può essere sempre espresso come la somma del numero di rose in $P(l_1 - 1, w_1 - 1)$ e $P(l_2, w_2)$ a cui va sottratto il numero di rose in $P(l_1 - 1, w_2)$ e $P(l_2, w_1 - 1)$. La seconda idea è che non è necessario considerare tutte le coppie di k-rettangoli, ma solo quelle che non ne contengono ulteriori al loro interno: presa una coppia di tali k-rettangoli disgiunti, quest'ultimi devono essere separabili da una linea orizzontale o verticale (altrimenti non sarebbero disgiunti). Possiamo quindi considerare w tagli orizzontali (e altrettanti tagli verticali) nel giardino, concentrandoci sul sotto-problema di trovare il k-rettangolo di perimetro minimo in ciascuna parte del taglio corrente. Possiamo affrontare quest'ultimo sotto-problema utilizzando uno schema basato su una "finestra scorrevole" di altezza prefissata che permette di identificare i k-rettangoli di tale altezza molto velocemente. Maggiori dettagli sono riportati nel sito www.ioi2005.pl/downloads/tasks.php.

Descrizione del secondo problema (*tempo limite di 5 secondi*): Consideriamo una sequenza non-decrescente di interi $s_1,\ldots,s_{n+1}$ (dove $s_i \le s_{i+1}$ per $1 \le i \le n$). La sequenza $m_1,\ldots,m_n$ definita da $m_i = \frac{1}{2}(s_i + s_{i+1})$ per $1 \le i \le n$, è chiamata la *sequenza delle medie* di $s_1,\ldots,s_{n+1}$. Per esempio, la sequenza di medie per 1, 2, 2, 4 è pari a 1.5, 2, 3. Notiamo che gli elementi di una sequenza di medie possono essere frazionari. Tuttavia, il problema proposto tratta esclusivamente sequenze di medie i cui elementi sono tutti interi. Data una sequenza non-decrescente di n interi $m_1,\ldots,m_n$ il problema richiede di calcolare il numero di sequenze non-decrescenti costituite da $n + 1$ interi $s_1,\ldots,s_{n+1}$ aventi $m_1,\ldots,m_n$ come sequenza delle medie. Scrivete un programma che legge da standard input una sequenza non-decrescente di interi e calcola il numero di sequenze non-decrescenti la cui sequenza delle medie corrisponda alla sequenza data.

Dati di input: La prima riga del file input.txt contiene un intero n ($2 \le n \le 5000000$). Le seguenti n righe contengono la sequenza $m_1,\ldots,m_n$. Il valore di m_i ($0 \le m_i \le 1000000000$) è fornito nella riga numero $i + 1$.

Dati di output: Il file output.txt deve contenere esattamente un intero che rappresenta il numero di sequenze non-decrescenti di interi che hanno la sequenza di input come sequenza di medie.

Descrizione del terzo problema (*tempo limite di 3 secondi*): Il Mountain Amusement Park ha aperto un nuovissimo simulatore di montagne russe. Il percorso consiste di n spezzoni collegati agli estremi, a partire dal primo spezzone fissato a quota 0. L'operatore Byteman può riconfigurare il percorso a piacere aggiustando le inclinazioni di spezzoni consecutivi. L'inclinazione dei rimanenti spezzoni non viene alterata. Ogni volta che egli agisce su uno spezzone, lo spezzone successivo viene sollevato o abbassato al fine di mantenere la continuità della rotaia, fermo restando che il primo spezzone resta fisso a quota 0. Ciascuna corsa inizia con il lancio del carrello a cui è impressa una certa energia, sufficiente al raggiungimento di una data quota h. Il carrello continua la sua corsa fino a quando non supera quota h oppure raggiunge la fine del percorso. Data la sequenza di tutte le corse e di tutte le operazioni di riconfigurazione avvenute nella giornata, calcolate, per ciascuna corsa, il numero di spezzoni attraversati dal carrello nella sua corsa. Internamente, il simulatore rappresenta le inclinazioni lungo il percorso come una sequenza di n dislivelli, uno per ogni spezzone. L'i-esimo numero di rappresenta, in centimetri, il dislivello in corrispondenza dello spezzone i-esimo. Ipotizzando che il carrello sia a quota h dopo aver attraversato i primi $i - 1$ spezzoni, esso raggiungerà quota $h + d_i$ dopo aver attraversato anche lo spezzone i-esimo. Inizialmente gli spezzoni sono tutti orrizzontali, ovvero d_i è pari a 0 per tutti gli i. Le corse e le operazioni di riconfigurazione si alternano arbitrariamente durante la giornata. Ciascuna riconfigurazione è specificata da tre numeri: a, b e D. Gli spezzoni coinvolti sono quelli del segmento da a a b (inclusi gli estremi). Il dislivello per ciascun spezzone del segmento viene posto uguale a D. In altre parole, $d_i = D$, $a \le i \le b$. Ciascuna corsa è specificata da un numero h che rappresenta la massima quota raggiungibile dal carrello. Scrivete un programma che legge da standard input una sequenza di operazioni di riconfigurazione e di corse e, per ciascuna corsa, calcola il numero di spezzoni attraversati dal carrello.

Dati di input: La prima riga del file input.txt contiene un intero positivo n che rappresenta il numero di spezzoni, $1 \le n \le 1\,000\,000\,000$. Le righe che seguono contengono le operazioni di riconfigurazione e le corse. Segue un terminatore. Ciascuna riga successiva alla prima è una delle seguenti:

- Riconfigurazione — una sola lettera 'I', e gli interi a, b e D, separati da spazi singoli (dove $1 \le a \le b \le n$ e $-1000000000 \le D \le 1000000000$).
- Corsa — una sola lettera 'Q', e un intero h ($0 \le h \le 1000000000$) separati da uno spazio;
- Una sola lettera 'E' — il terminatore, a indicare la fine dell'input.

Possiamo ipotizzare che, in un qualsiasi momento, la quota raggiunta da un qualsiasi punto del percorso sia nell'intervallo [0, 1000000000] in centimetri. L'input non contiene più di 100000 righe.

Dati di output: Il file output.txt deve contenere, nella riga i-esima, un solo intero che rappresenta il numero di spezzoni attraversati dal carrello durante l'i-esima corsa della giornata.

I NUOVI PERCORSI DELL'INTERAZIONE UOMO-MACCHINA

Stefano Levialdi

Le tecnologie ICT mettono sul mercato prodotti sempre più complessi per le applicazioni più varie: calcolo, comunicazione, servizi, creazione di contenuti, immagini, suoni. L'uso di questi prodotti implica una forte interattività con l'utente e pone quindi il problema di creare interfacce quanto più possibile amichevoli. Questo articolo presenta lo stato delle ricerche su alcuni tra i paradigmi più innovativi e sofisticati.

3.5

1. INTRODUZIONE

È noto ai lettori di questa rivista, attraverso articoli precedenti [16], che uno dei colli di bottiglia più significativi per l'impiego di nuove applicazioni nel mondo ICT è costituito dall'interfaccia. In generale, possiamo considerare il settore dell'interazione uomo-macchina come campo di ricerca applicata dove vengono studiate le caratteristiche che devono avere le interfacce per poter essere usabili ed accessibili dal maggior numero di persone. Usabili nel senso che siano facili da apprendere, agevoli da impiegare, efficienti dal punto di vista dello sforzo intellettuale necessario per capirle e adoperarle, ed effettive, cioè, che producano esattamente i risultati richiesti, niente di meno e soprattutto niente di diverso. Devono essere accessibili a tutti e, in particolare, come sottolineato dalla Comunità Europea, a portatori di handicap (visivi, uditivi o motori) e a utenti molto diversi per età, cultura, esperienza tecnica, origine culturale e linguistica.

Gli studi sull'interazione uomo-macchina – che negli Stati Uniti vanno sotto il nome di CHI, *Computer-Human Interaction*, mentre in Europa si preferisce il termine HCI, privilegiando il fattore umano sulla macchina – comprendono diverse tematiche e discipline come le scienze cognitive (i meccanismi di apprendimento e memorizzazione), lo studio dell'attenzione e della percezione negli esseri umani, la modellazione dell'utente e dei diversi livelli di esperienza tecnica e, infine, gli esperimenti appositamente definiti per valutare – in termini quantitativi – la validità delle nuove proposte di interfaccia.

Per affrontare e risolvere le questioni sopra poste, si deve ricorrere a specialisti e progettisti informatici, che dovranno implementare le nuove interfacce lavorando in stretto contatto con gli utenti, cioè con un approccio di tipo *user centered*. L'utente è sicuramente il vero esperto del dominio applicativo per il quale l'interfaccia è progettata; è quindi colui che conosce le esigenze tipiche di quel dominio, vale a dire: le priorità, le connotazioni, ciò che deve essere visualizzato e stampato, la disponibilità dei risultati parziali e totali ecc.. Nell'ambito del

gruppo di progettazione sarà presente anche un esperto di usabilità (a volte chiamato *human factors engineer*) onde realizzare una interfaccia usabile da un certo gruppo di utenti, che svolge un particolare lavoro, in un determinato ambiente, che risulti efficiente ed effettiva.
Questo articolo intende presentare i nuovi percorsi che sono attualmente esplorati per consentire, ad una quantità sempre maggiore di utenti, di sfruttare le funzionalità dei nuovi dispositivi che la tecnologia ICT propone al mercato. Questi nuovi percorsi si chiamano: calcolo emotivo, calcolo ubiquo, calcolo indossabile ed, infine, calcolo trasparente.

2. CALCOLO EMOTIVO

La "qualità affettiva" di un sistema può essere definita in base alle reazioni emotive suscitate.
"L'emotività include il modo, le sensazioni ed i sentimenti che sono le caratteristiche fondamentali dell'essere umano e che influenzano notevolmente i suoi riflessi, la sua percezione, la sua conoscenza ed infine il suo comportamento" [14]. Il processo emotivo è di natura giudicante, assegna in modo veloce ed efficiente valenze positive o negative all'ambiente, mentre quello cognitivo interpreta tali valenze e fornisce un significato al mondo circostante.
È appena il caso di sottolineare che gli effetti delle emozioni sui processi razionali dell'uomo sono estremamente significativi, tanto da velocizzarli o rallentarli, a seconda del segno positivo o negativo delle emozioni percepite.
In effetti, la qualità affettiva è l'abilità di un oggetto o stimolo a modificare l'attaccamento di una persona a quell'oggetto. I dati sperimentali ottenuti nelle ricerche sulla interazione uomo-macchina, suggeriscono che la qualità affettiva di una interfaccia ha un impatto sulla usabilità percepita dall'utente di tale interfaccia.
Lo stato emotivo, psicologico, sentimentale di una persona è di natura fisiologica, accessibile in modo conscio e semplice, non riflessivo; una miscela di piacere/dispiacere, di valori buoni/cattivi che generano interesse/repulsione in misura proporzionale all'impegno che la persona ritiene di volere assumere. Questo stato è di natura primitiva, universale ed è alla base di ogni tipo di evento portatore di emozioni.
La qualità affettiva di un sistema si può anche definire come la capacità di suscitare una modifica del nucleo emotivo di una persona. Questa proprietà, che appartiene agli stimoli, è presente anche nell'interfaccia del sistema. In altre parole, questi oggetti determinano la qualità affettiva in quanto vengono interpretati dall'utente sulla base di quanto siano essi piacevoli/spiacevoli, eccitanti/noiosi, scatenanti/tranquillizzanti e influenzano quindi le reazioni dell'utente.
Introduciamo ora dei parametri che servono per studiare le motivazioni delle scelte degli utenti rispetto alle caratteristiche emotive della interfaccia. Iniziamo dalla PAQ (*Perceived Affective Quality*), che possiamo definire come la qualità emotiva percepita: è la percezione individuale della capacità che ha la tecnologia informatica di modificare la componente emotiva, e viene misurata attraverso la sua valenza e l'attivazione conseguente. Un secondo parametro interessante è la PU (*Perceived Usability*) o usabilità percepita; essa ha un forte impatto sulla intenzione del comportamento dell'utente, denominata BI (*Behavioural Intention*), così come anche lo ha la PEOU o facilità d'uso percepita (*Perceived Emotional Observed Usability*).
L'ansia, una tipica forma di emozione, generata dall'uso (almeno all'inizio) di una nuova interfaccia, produce un effetto sulla PU mediato dalla PEOU; d'altra parte il piacere di usare l'interfaccia (detto *joy-of-use*) ha sia le componenti della valenza che dell'attivazione sul nucleo personale dell`emotività (che quindi fornisce un impatto positivo), sia sulla usabilità percepita, che sulla facilità d'uso percepita. La scelta dell'interfaccia da parte dell'utente mostra una forte correlazione con i valori della PU e della PEOU, poiché quella scelta lascia intravedere una maggiore usabilità.
Da una analisi compiuta sul tema [14], è stato ricavato un modello empirico delle relazioni fra i parametri prima introdotti (Figura 1); i valori numerici indicati riflettono i legami osservati negli esperimenti sugli ef-

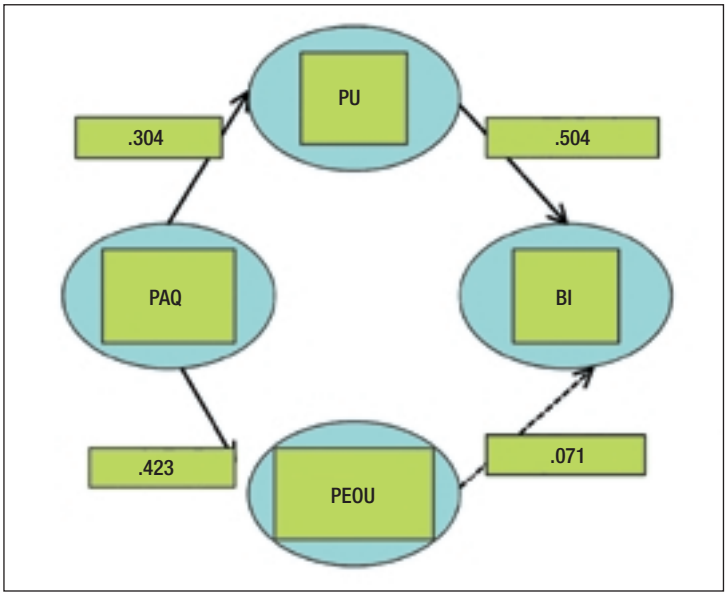


FIGURA 1
Modello empirico per l'analisi dei parametri emotivi [14]

fetti emotivi percepiti dagli utenti. Questi esperimenti sono stati fatti con studenti che dovevano valutare il sito Internet della loro università; la PAQ è stata misurata sulla base di quattro valori legati alle qualità del sito: divertente, noioso, piacevole, spiacevole; hanno partecipato all'esperimento 194 giovani. Da una analisi statistica accurata, è risultato che la PAQ aveva un effetto positivo notevole sia sulla PU che sulla PEOU, ma non in modo diretto sulla BI (comportamento successivo), che è invece mediato da PU e da PEOU. In sintesi, la valutazione cognitiva da parte dell'utente dello strumento informatico (il sito Internet in questo caso) discende dal primo impatto emotivo, detto anche *touch-and-feel*, così come avviene nell'incontro con una persona per la prima volta, dove le sensazioni emotive iniziali sono determinanti.

I concetti di affetto ed emozione sono di natura complessa e difficile da esprimere quantitativamente e, ancor più, da inserire in programmi o in macchine che dovranno interagire con esseri umani; la psicologia, però, può aiutarci a capire quale è il ruolo delle emozioni nei nostri comportamenti e quindi perché ne dobbiamo tenere conto durante la progettazione di sistemi informatici come le interfacce e, più in generale, di programmi applicativi. Dai parametri considerati si vede che vi sono delle valutazioni sulle aspettative degli utenti; in altre parole, quali emozioni susciteranno i programmi che adopereremo per la prima volta? Oltre a ciò, quali emozioni umane possiamo dire che vengano riconosciute dalla macchina (da opportuni programmi) in modo tale da adeguare le successive interazioni ai nostri "umori emotivi"? Come si vede, ci troviamo davanti ad una situazione simmetrica, tra essere umano e sistema, dove le emozioni giocano un ruolo importante per la buona riuscita delle successive interazioni.

Possiamo anche considerare la qualità estetica delle presentazioni di informazioni [2]. Anche esse suscitano emozioni nell'utente, in particolare per quanto riguarda la pluralità (la molteplicità di informazioni) e la prospettiva (la possibilità di osservazione da diversi punti di vista), analoghe alle "viste" dei dati nelle basi di dati. Riprendendo una frase da [2], "è preferibile vedere un tramonto con mille occhi diversi, che mille tramonti". Vi è un forte collegamento fra l'aspetto estetico e le emozioni, in quanto il primo suscita le seconde, facilitando l'estrazione di dati significativi da grosse moli di informazioni. Lo stile, il genere ed il tema corrispondono ad una struttura che può facilitare la comprensione ed il ritrovamento dei dati rilevanti per l'utilizzo dell'applicazione. Come non vi è un'unica interpretazione della realtà, così dicasi per la rappresentazione della realtà; non esiste la visualizzazione ottimale, bensì quella più adatta per un certo insieme di utenti che devono svolgere un compito all'interno di uno specifico dominio applicativo.

Tra i maggiori esperti del calcolo emotivo, nonché antesignana del settore, è Rosalind Picard, che lavora al prestigioso MIT di Boston [7]. Lei ed i suoi collaboratori si muovono in un ambito che parte dalle scienze cognitive, prende i risultati della psicologia, delle neuroscienze, della medicina, della psicofisiologia, della sociologia e dell'etica, per sviluppare degli strumenti ingegneristici in grado di misurare, modellare, ragionare e reagire alle emozioni. A questo scopo, stanno sviluppando sensori, algoritmi, sistemi e teorie che dovrebbero consentire di sintetizzare nuove forme di intelligenza e di comprensione da parte degli umani.

Come esempio di progetto emotivo che viene

svolto al MIT, si può citare il "Sarto per Interfaccia", un agente (cioè un programma con autonomia e con scopi prefissati) che cerca di adeguare un sistema in risposta ad una reazione emotiva. La frustrazione, rivelata attraverso esperimenti, viene impiegata come elemento di selezione fra le reazioni dell'utente a diverse interfacce. Per esempio, questo approccio è utilizzato per cercare di rendere più adattivo l'"Assistente" dei programmi Microsoft.
Si possono citare diverse fonti che illustrano quanto sia significativo considerare le emozioni nei momenti decisionali, di ragionamento e di comportamento. In particolare, il comportamento dell'utente dipende da ciò che viene percepito [4]; per esempio, una interruzione durante un ragionamento può portare a concentrarsi su un maggior livello di dettagli, oppure può essere causa di frustrazione (per esempio, squilla il telefono durante l'esecuzione di una parte critica di un lavoro). La frustrazione, invece, genera sempre un atteggiamento negativo ed aumenta la probabilità di errore.
Ci sono ormai nel mondo diversi gruppi qualificati che si occupano dell'approccio emotivo per migliorare l'interazione uomo-macchina (utente-programma); un breve elenco è riportato in [1]. Un centro di ricerca è presso l'Università di Stanford, dove l'importanza delle emozioni è ben presente ("No aspect of our mental life is more important to the quality and meaning of our existence than emotions" [10]). Come già accennato, anche al MIT c'è, da tempo, un gruppo di ricercatori che lavora sul tema del calcolo emotivo [8].
Donald Norman, guru dell'informatica cognitiva, che si occupa di un progetto per le macchine emotive, spiega così le sue idee sul ruolo delle emozioni nei calcolatori del futuro: "progetto emotivo: bellezza e cervello, piacere ed usabilità vanno mano nella mano per ottenere una buona progettazione" [5].

3. CALCOLO UBIQUO

Passiamo ora ad un'altra forma di calcolo che è alla ribalta: si tratta del calcolo ubiquo. Il principale propugnatore di questa forma di calcolo è Mark Weiser dello Xerox PARC (*Palo Alto Research Center*) in California, fucina di molte delle idee rivoluzionarie nel campo dell'informatica degli ultimi cinquanta anni.
Questo tipo di calcolo si ha quando la computazione viene integrata con l'ambiente; ci troviamo allora davanti ad un sistema di calcolo ubiquo (*ubiquitous computing*). Va detto che il calcolo ubiquo può coesistere col calcolo emotivo visto prima.
Vi è anche un altro modo di illustrare lo stesso concetto, dicendo che il calcolo ubiquo coincide col calcolo "pervasivo", secondo cui gli utenti non dovrebbero più portare con sé o avere in ufficio o in casa dei calcolatori, in quanto potrebbero sfruttare le potenze di calcolo (e di comunicazione) presenti nei sistemi integrati in edifici, pareti, mobili ecc.. In questo modo viene facilitata anche l'interazione col sistema attraverso gesti, parole, sguardi, movimenti del corpo. In effetti, i sistemi integrati nell'ambiente dovranno essere sensibili ad ogni cambiamento prodotto in loco, riconoscere automaticamente tale cambiamento, ed agire poi sulla base delle necessità dell'utente e delle sue preferenze.
Come semplici esempi di applicazioni di questo tipo, si può citare il segnale proveniente da un sistema GPS (sistema di posizionamento globale) che arriva ad un guidatore in automobile, fornendogli le indicazioni necessarie per arrivare a destinazione; oppure un lettore di codice a barre dei prodotti di un supermercato, che fornisce all'uscita il costo della merce inserita nel carrello.
Il già citato Weiser (ora all'IBM) in un suo articolo [13] parla di "spazi mediatori" e presenta il suo punto di vista sul calcolo ubiquo. Secondo Weiser, il calcolo ubiquo rappresenta la terza ondata della computazione: la prima è quella dei mainframe (nati negli anni '50), la seconda è quella dei personal computer (anni '70-'80), e la terza è quella del calcolo ubiquo (il "terzo paradigma" del calcolo, anche secondo Alan Kay). Le tre ere sono rappresentate nel grafico di figura 2. Come si vede, l'autore è assai ottimista sulla crescita del calcolo ubiquo.
Il calcolo ubiquo può essere contrapposto alla realtà virtuale nel senso che questa cerca di inserire le persone, gli utenti, in una realtà artificiale creata dai programmi (sempre più

realistici), mentre il calcolo ubiquo spinge le macchine dovunque ci siano delle persone, ma non necessariamente nelle loro case od uffici. La realtà virtuale richiede grandi potenze di calcolo per elaborare o riprodurre le sottigliezze e i dettagli della realtà fisica; il calcolo ubiquo, invece, si basa su una buona integrazione di fattore umano, informatica, ingegneria e scienze sociali.

I primi oggetti destinati a sperimentare il concetto di ubiquità sono stati costruiti negli anni 1988-1994 presso i laboratori di ricerca applicata dello Xerox PARC, ed hanno nomi come *tab, pad*, e *board*. Sono costituiti da piccoli calcolatori che possono eseguire compiti semplici, sono mobili e possono comunicare fra di loro senza fili mediante trasduttori a raggi infrarossi; nel loro insieme costituiscono un sistema chiamato PARCTAB.

La premessa da cui è partito Weiser è che le persone vivono e lavorano in accordo a prassi e conoscenze tacite che, normalmente, nella realtà non sono individuabili (sono quindi invisibili).

Il primo tentativo di riprodurre questa nuova realtà è stato quello di distribuire alle persone e negli uffici centinaia di dispositivi di calcolo senza fili, di tutte le dimensioni, da piccoli schermi di un pollice a quelli giganteschi da muro. Questo tentativo ha richiesto la riprogettazione dei sistemi operativi, delle interfacce utente, delle reti, dei dispositivi di comunicazione, dei pannelli di visualizzazione e di molti altri dispositivi tecnologici. Questa attività è stata denominata "calcolo ubiquo" ed è diversa dalla costruzione di PDA (assistenti personali), dynabook (quaderni con puntatori ipermediali) o anche dei minidispositivi che forniscono informazioni sulla punta delle dita dell'utente.

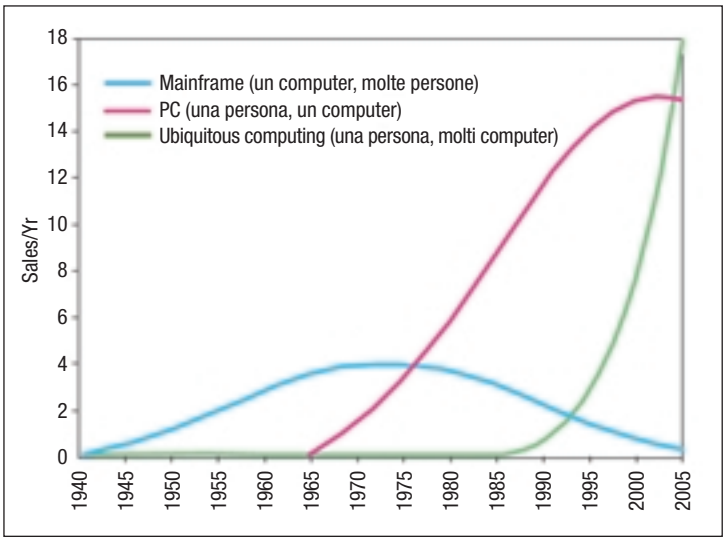


FIGURA 2

Le ere dell'informatica secondo Weiser [13]

Si tratta quindi di realizzare un sistema che non si vede, che funziona ovunque, che non risiede all'interno di un dispositivo personale, ma si trova in ogni posto; un sistema, cioè, ubiquo.

Molti gruppi di ricerca applicata hanno cercato di rendere il calcolatore un oggetto del desiderio: esteticamente interessante, funzionale, semplice da usare, di valore simbolico, che "parlasse delle qualità di chi lo possiede". Un esempio di oggetto che riassume queste caratteristiche è iPod della Apple, una memoria portatile che consente l'ascolto di brani musicali, di cui sono stati prodotti più di cinquanta milioni di esemplari.

In direzione opposta si colloca il calcolo ubiquo che mira ad oggetti così ben inseriti nel contesto da essere invisibili e poter essere usati senza neanche conoscere dove si trova la piattaforma di calcolo.

Presso il *Georgia Institute of Technology* è in funzione il *Graphics Visualization ed Usability Lab* [3], che ha un settore dedicato al calcolo ubiquo, dove sono in corso diversi progetti interessanti, ispirati a questo paradigma di calcolo.

Un esempio è l'archivio di fotografie familiari, dove sono registrate sequenze di immagini corredate di dati che consentono di ritrovare la sequenza voluta. Questo progetto include anche la possibilità di annotare automaticamente una sequenza di immagini durante la ripresa (*Context Camera*).

Un altro progetto, denominato *Abaris*, usa una tecnologia di cattura e di recupero di scenari per facilitare l'analisi, in modo discreto, dei comportamenti di bambini autistici. *PAL* invece è un loop audio che aiuta a riprendere il filo di una conversazione quando questo è stato interrotto (anche pochi minuti prima). Infine, un altro esempio di progetto applicativo è quello che utilizza telecamere, macchine fotografiche e proiettori per impedire riprese (foto, video, audio) non autorizzate, come ad esempio nei musei e gallerie d'arte.

4. CALCOLO INDOSSABILE

Secondo una definizione (Steve Mann [4]), il calcolo indossabile si ha quando il calcolatore (o la piattaforma di calcolo) è integrato nello spazio personale dell'utente ed è operativo ed interattivo in ogni momento, essendo sempre acceso e disponibile. In generale, si tratta di uno o più dispositivi che, essendo sempre assieme all'utente, sono in grado di ricevere ed eseguire i comandi che ricevono in qualunque situazione, sia che la persona sia ferma oppure in movimento. Naturalmente, a differenza di altri dispositivi indossabili come l'orologio o le cuffie, il calcolatore indossabile (*wearable computer*) ha tutte le funzionalità di un sistema programmabile; è cioè utilizzabile per applicazioni generali ed è riconfigurabile, restando però strettamente connesso a chi lo indossa.

Per descrivere con maggiore dettaglio il calcolo indossabile, dobbiamo riferirci alle sue tre caratteristiche operative e cioè:

1. *constancy*;
2. *augmentation*;
3. *mediation*.

La prima caratteristica sta ad indicare il fatto che il funzionamento è continuativo e la disponibilità nel tempo è totale, a differenza di altri dispositivi come, ad esempio, il personal assistent (PDA) che deve essere acceso ed avviato; il flusso informativo fra calcolatore indossabile ed utente è sempre presente in entrambe le direzioni.

La seconda caratteristica si riferisce al fatto che il calcolatore è operativo anche mentre l'utente sta svolgendo una attività; in questo modo vengono aumentate le capacità sensoriali e intellettive dell'utente.

Infine, la terza caratteristica, consente di proteggere l'utente in misura maggiore rispetto a ciò che avviene con calcolatori usuali, anche portatili. Possiamo ulteriormente precisare questa caratteristica considerando due aspetti, cioè isolamento e *privacy*. Il primo indica il funzionamento come filtro, in modo da non ricevere informazioni indesiderate. Il secondo riguarda, invece, la possibilità di nascondere alcune caratteristiche dell'utente o di presentarne altre a seconda degli interlocutori, degli ambienti, delle situazioni. Quando dobbiamo (o vogliamo) interagire con sistemi di elaborazione ignoti o potenzialmente pericolosi, potremmo proteggerci mediante il calcolo indossabile, presentandoci in modo "schermato". Diventa più difficile osservare un utente (via telecamera o microfono) se egli si presenta con un programma che lo "avviluppa".

Poiché il calcolatore indossabile è a contatto di pelle (attraverso tessuti con microcalcolatori incorporati) di chi lo indossa, è anche possibile rilevare parametri biomedici di interesse per monitorare lo stato di salute del portatore. In sintesi, fra utente e calcolatore è possibile inserire uno strato di protezione opzionale che si può togliere, anche parzialmente, se si desidera una modalità operativa che sfrutti una combinazione di *augmentation e mediation*.

I flussi informativi corrispondenti ai tre elementi prima indicati sono rappresentati nella figura 3, dove il ruolo dell'umano rimane costante nel loop con la macchina, oppure si modifica all'interno di un loop o, infine, si trova semplicemente all'interno del percorso di elaborazione.

Vediamo ora le sei caratteristiche specifiche dei segnali informativi all'interno del calcolo indossabile, che sono presenti nel rapporto sinergico fra l'essere umano e il calcolatore indossato.

1. Caratteristica non monopolizzante: implica la possibilità per l'utente di svolgere altre attività oltre a quelle possibili con il calcolatore indossato. Diversamente dalla realtà virtuale, la persona non è immersa in un mondo diverso da quello reale. Il calcolo risulta una attività secondaria anzichè l'aspetto primario dell'attenzione. In effetti, in teoria, il sistema indossato può fornire ulteriori capacità sensoriali anche se potrebbe mediarle, ciòè, modificarle, o addirittura diminuirle.
2. Caratteristica non restrittiva: non diminuisce la mobilità dell'utente, non inibisce il suo spostamento fisico.
3. Caratteristica di osservabilità: in ogni momento l'attenzione dell'utente può essere rivolta ai risultati del calcolo, agli stati successivi del sistema indossato.
4. Caratteristica di controllabilità: il sistema reagirà sempre e soltanto su richiesta dell'utente; quindi il controllo sarà sempre in

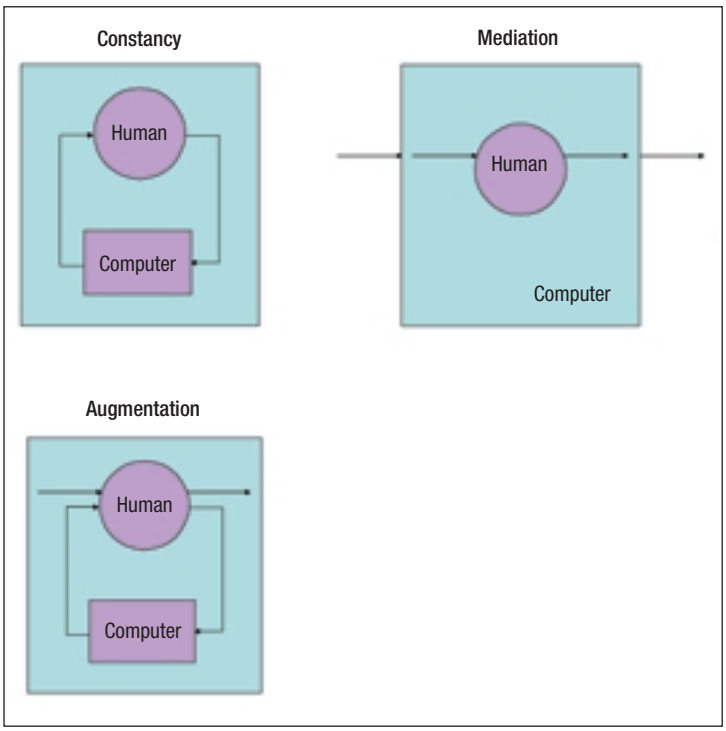


FIGURA 3
I flussi informativi nelle tre diverse modalità operative del calcolo indossabile [4]

mano di chi lo indossa. Anche in presenza di un programma automatico, l'utente potrà sempre interromperne l'esecuzione. La continuità del rapporto fra utente ed interfaccia produce la totale controllabilità del sistema indossato.

5. Caratteristica di attenzione: è riferita all'ambiente, ossia sensibilità rispetto al posto ed al contesto dove il sistema si trova.

6. Caratteristica comunicativa: rispetto ad altri sistemi, aumentando le capacità di espressione dell'utente oppure migliorando la produzione di informazioni di vario tipo, ad esempio artistico.

Possiamo ora vedere quali sono veramente le applicazioni del calcolo indossabile. Mann osserva che ogni nuova disciplina informatica inizia con una teoria di base, quasi un sogno del proponente; a volte le teorie hanno delle applicazioni semplici ed immediate.

Nel settore del riconoscimento di forme (*pattern recognition*) l'analisi dei volti per confrontarli con quelli di persone pericolose aumenta la sicurezza in situazioni difficili; questo è sicuramente un campo nel quale il calcolo indossabile può fornire una risposta sicura ed affidabile se gli algoritmi di riconoscimento vengono eseguiti mediante confronti con una base dati di volti.

Un'altra applicazione, può essere la riparazione di uno strumento: non sempre è possibile avere un manuale a disposizione, mentre il dispositivo indossabile può fornire la descrizione, le regole di funzionamento e le prove necessarie per trovare il guasto di una fotocopiatrice, di una fotocamera digitale ecc..

Nel caso di persone con problemi di vista, l'applicazione *BlindVision* fornisce un sistema basato su radar che, incorporato in una maglia da indossare, è in grado di rivelare oggetti vicini al portatore, restituendo segnali e.m. riflessi dagli oggetti; gli oggetti vicini forniscono segnali diversi da quelli lontani. Questo sistema si può descrivere come la visione basata sul senso del tatto. Il sistema potrebbe anche essere impiegato da ciclisti o motociclisti che debbano muoversi in ambienti con ostacoli.

Un'altra applicazione, inversa della precedente, è quella chiamata *Mediwear*, che elabora segnali elettrofisiologici del corpo umano (anziché quelli esterni prodotti dalle onde radar riflesse), svolgendo funzioni di monitoraggio, valutando parametri utili a stabilire la differenza fra stato normale e stato critico; in quest'ultimo caso, *Mediwear* manda una comunicazione ad un presidio medico.

ENGwear è, invece, un sistema per la raccolta di informazioni da spedire ad un gruppo, una comunità con la quale l'utente condivide degli interessi. Mentre le applicazioni viste sopra sono di natura individuale, questa differisce nel senso che consente a più persone di scambiarsi informazioni (testuali, grafiche, sonore ecc.) su argomenti di interesse comune. In altre parole, *ENGwear* è un sistema indossabile per la cooperazione assistita da calcolatore (*computer-supported collaborative work*, CSCW).

Le applicazioni menzionate sono solo alcune delle possibilità offerte dal calcolo indossabile; sicuramente ne nasceranno delle altre man mano che la tecnologia consentirà di incorporare negli abiti microdispositivi sempre più sofisticati.

Passiamo ora a riassumere le caratteristiche fondamentali del calcolo indossabile. Certamente l'aspetto essenziale è quello del potenziamento individuale grazie al fatto di

equipaggiare la persona con uno spazio informativo privato, personalizzabile e totalmente controllabile. Anche il calcolatore che funziona a casa o in ufficio è controllabile dall'utente, ma non è così personale e in grado di potenziare le capacità dell'utente come un calcolatore indossabile, totalmente integrato nello spazio personale.
Agli inizi, negli anni '70, il calcolo indossabile si presentava come uno schermo miniaturizzato collocato su uno o due occhi, con dispositivi di ingresso come tastiere o joystick da tenersi in mano ed un microfono per potere inserire il parlato. Naturalmente, e spinti da applicazioni militari, questi dispositivi, a parità di prestazioni, hanno ridotto grandemente le loro dimensioni nell'arco degli ultimi dieci anni.
Ma quali sono gli aspetti essenziali che consentono il potenziamento dell'utente attraverso il calcolo indossabile? Uno è costituito dall'ampliamento delle funzioni di memoria, ossia il recupero di tutte le informazioni raccolte (con telecamera, con radar, con microfono ecc.) e la possibilità di condividerle con più individui, che possono sfruttare informazioni reciproche, scambiarle, utilizzare informazioni provenienti da altri utenti ecc.. Una forma quindi di intelligenza collettiva, nel senso che più individui possono collaborare (attraverso sistemi CSCW) anche quando qualcuno di loro sta eseguendo un compito diverso. Altre applicazioni riguardano la sicurezza, che differisce dai sistemi di sorveglianza centralizzata in quanto il sistema si trova sugli abiti dell'utente e quindi è in grado di proteggerlo a livello individuale.
Si può operare senza connessioni fisiche, nel senso che non si richiede un cavo o una spina per poter funzionare, un funzionamento quindi totalmente autonomo. Infine, si può dire che si realizza un sistema sinergico, dove la persona svolge i compiti creativi o comunque più pertinenti alla sua natura, mentre il calcolatore svolge i lavori ripetitivi. Dopo un certo periodo, il calcolo indossabile diventa una estensione della mente e del corpo del portatore, ed allora non è più una entità separata. Possiamo, cioè, parlare di una integrazione delle due capacità, invece che di una semplice somma; a volte questa sinergia viene chiamata Intelligenza Umanistica. Mann [4] parla anche di un aumento della qualità della vita, che appare però discutibile, tenendo conto di certe conseguenze dell'automazione in genere e delle applicazioni informatiche in particolare. È più prudente esprimere un giudizio cauto; il calcolatore da indossare può risultare per alcuni un piacevole accessorio che facilita i suoi compiti, per altri, invece, un fastidioso aggeggio dal quale è meglio prescindere.
Presso il MIT di Boston è in funzione, all'interno del Media Lab, il Laboratorio di Dinamica Umana dove, sotto la direzione di Sandy Pentlan, lavora un gruppo di una ventina di ricercatori. L'approccio di questo gruppo si basa sull'idea che l'impiego del calcolatore deve essere completamente modificato rispetto all'uso tradizionale, deve cioè essere indossato come gli occhiali o gli abiti e l'interazione con esso deve dipendere dalla situazione. Per potere sfruttare questa tecnologia bisognerà fornire il portatore con visualizzatori collocati sulla testa, dispositivi di ingresso che non ostacolino le attività correnti, reti locali senza fili (sia per l'alimentazione che per la trasmissione/ricezione dell'informazione), e inoltre un certo numero di sensori, in modo che l'utente possa disporre di un agente pro-memoria, una realtà aumentata e possa interagire con una collettività di individui, anch'essi portatori di sistemi di calcolo indossabili.
Vediamo ora alcune applicazioni studiate presso il MIT. Il sistema *LiveNet* incorpora delle tecnologie per il controllo della salute (analogamente a *Mediwear*); è questo un tema che acquisterà sempre maggiore importanza in futuro, anche a causa dell'aumento dell'età media della popolazione. Il progetto ha, in particolare, l'obiettivo di fornire rapporti aggiornati sullo stato di salute di una persona combinando la disponibilità di un personal assistant (PDA), basato sul sistema operativo Linux, con quella di software open-source per la gestione dei sensori specializzati per la trasduzione di segnali bio-elettrici. *LiveNet* può essere impiegato per il monitoraggio prolungato nel tempo, per osservare variazioni lente di pa-

rametri biofisici e per comunicare i risultati ad unità terapeutiche a distanza.
GroupMedia è un insieme di strumenti ed applicazioni che consentono di rimanere aggiornati utilizzando le conoscenze dei membri di un gruppo mediante telefoni cellulari e strumenti tipo PDA. Mediante studi sui modelli di comportamento e di interazione sociale, si pensa di realizzare del software specifico per la prossima generazione di calcolo indossabile.
Un altro esempio rilevante è *Reality Mining*, considerato il più grande progetto basato su telefonia mobile negli Stati Uniti, che, sfruttando la grande diffusione dei cellulari e utilizzando dati sulla dinamica del comportamento individuale e di gruppo, cerca di costruire dei modelli con cui prevedere il comportamento futuro dell'utente e delle organizzazioni sociali.
Si può citare anche il progetto *Learning Humans* che mira a sviluppare delle tecniche per capire i meccanismi del comportamento umano nell'ufficio (o in altri ambienti) e valutare le interazioni delle persone nei processi operativi, onde consentire una integrazione ottimale fra attività informatizzate e manuali nell'attività quotidiana.
Un altro centro dedicato al calcolo indossabile è il *Wearable Computing Laborator* dell'University of Oregon [6]. È stato creato nel 1995 con l'obiettivo di progettare, sviluppare e valutare la tecnologia del calcolo indossabile e mobile, per facilitare e migliorare la collaborazione fra le persone.
Tra i progetti in corso c'è quello denominato "comunità di calcolo indossabile", che intende sfruttare le tecnologie di cui parliamo per aiutare le persone durante i loro incontri nel mondo quotidiano, nell'ufficio, nell'ascensore, nei negozi... Il quesito cui il progetto cerca di dare una risposta è come possa il calcolo mobile ed indossabile facilitare o aumentare le interazioni sociali e quale effetto possa avere sulle modalità di interazione fra le persone e le comunità cui appartengono.
Un altro progetto, denominato iSIM (*Computing Environment Simulator*) serve a collaudare applicazioni indossabili e dipendenti dal contesto prima della effettiva implementazione. I fattori considerati nelle simulazioni sono vari: la luce, i suoni, il sistema di posizionamento (GPS), la temperatura, le reti wireless ecc.. Il sistema è anche in grado di gestire le interazioni fra agenti, reti di interconnessione fra diverse piattaforme di calcolo, mappe topografiche ecc.. Ancora presso l'Università dell'Oregon, degno di menzione è il progetto *Auranet*. Questa applicazione è mirata a persone appartenenti a comunità diverse che si incontrano in uno spazio sociale chiamato "Aura". Il collante fra queste persone non è lo scambio di danaro o beni materiali, ma la disponibilità di tempo per aiutarsi a vicenda, condividere certi valori, creare e gestire rapporti di mutuo beneficio. Vi è una somiglianza fra queste comunità che indossano dispositivi di calcolo e quelle ormai diffuse su Internet; in entrambi i casi, infatti, l'obiettivo è di migliorare la qualità e lo spirito della cooperazione. Si può dire che ciò che i siti comunitari fanno nel Cyberspazio, il calcolo indossabile lo fà nella vita reale.
Non è possibile valutare ora quali saranno le conseguenze della diffusione del calcolo indossabile, quali i benefici e quali i limiti, ma sicuramente queste tecnologie influiranno sulle interazioni presenti nel mondo del lavoro, negli scambi sociali e nelle diverse attività, anche ludiche, svolte dagli esseri umani.

5. CALCOLO TRASPARENTE

Ed ora, per finire, un cenno ad uno dei temi più recenti della ricerca sulle interfacce e l'interazione uomo-macchina, cioè il calcolo trasparente. All'interno dell'informatica, la trasparenza è sempre stata vista come quella proprietà che nasconde uno strato di programmazione (interfaccia trasparente è quella che non si vede), facilitando quindi il lavoro dell'utente in modo che non debba preoccuparsi dei dettagli implementativi di funzioni (come per esempio la gestione delle finestre sullo schermo) e quindi possa concentrarsi sul lavoro da svolgere ossia l'utilizzo delle applicazioni.
Recentemente, su suggerimento di Steve Tanimoto ([11] e comunicazione personale), è nata l'idea di una computazione di natura trasparente, dove cioè l'esecuzione detta-

gliata dei programmi interni al sistema è visibile dall'utente. Normalmente, la visibilità dei dettagli del sistema, delle ipotesi di partenza, dei processi attivi e di altre caratteristiche del comportamento del sistema, sono in contrasto con i metodi cosiddetti della scatola nera, con l'incapsulamento degli oggetti e, più in generale, coi principi dell'ingegneria del software.
L'approccio del calcolo trasparente offre invece la possibilità di visualizzare – a diversi livelli – l'evoluzione della computazione, si tratti di calcolo numerico o di trasformazioni di immagini o del percorso su un grafo. Questa possibilità, che in futuro potrebbe diventare controllabile dall'utente, ha subito due conseguenze rilevanti:

1. una migliore comprensione del significato dei programmi e della loro esecuzione

2. una maggiore efficacia nella presentazione di un sistema o di una piattaforma a scopi didattici e/o di addestramento.

La trasparenza può, a differenza dell'ingrandimento del livello di granularità nell'astrazione, fornire viste diverse di uno stesso sistema, più adeguate per un utente che deve apprendere, riparare, utilizzare, spiegare tale sistema. Si possono anche fornire delle trasparenze specifiche come quelle sintattiche (dettagliando le regole), semantiche (fornendo significati precisi), temporali (visualizzando i diversi stati del sistema nell'arco del tempo).
Per fare un esempio di trasparenza sintattica nei confronti di un algoritmo e della sua successiva codifica in un programma, la figura 4 mostra la versione grafica di un linguaggio visivo basato sul flusso dei dati presentato in un convegno sui linguaggi visuali di diversi anni fa. Si tratta di un semplice convertitore da gradi Fahrenheit a Celsius e lo schema mostra i passi della computazione dall'ingresso del dato, alle elaborazioni successive fino al risultato finale nella casella designata.
Nelle interfacce grafiche sono comuni le icone che rappresentano dei comandi; una di queste è un interruttore con più di due stati (*toggle switch*), che però non mostra in quale stato si trova. Per esempio, il comando che serve per passare dalla proiezione di una diapositiva su uno schermo murale allo schermo del computer, o su entrambi gli schermi o su nessuno di essi, richiede evidentemente quattro stati che, però, non sono visibili all'utente. In questo caso si può parlare di mancanza di trasparenza semantica.

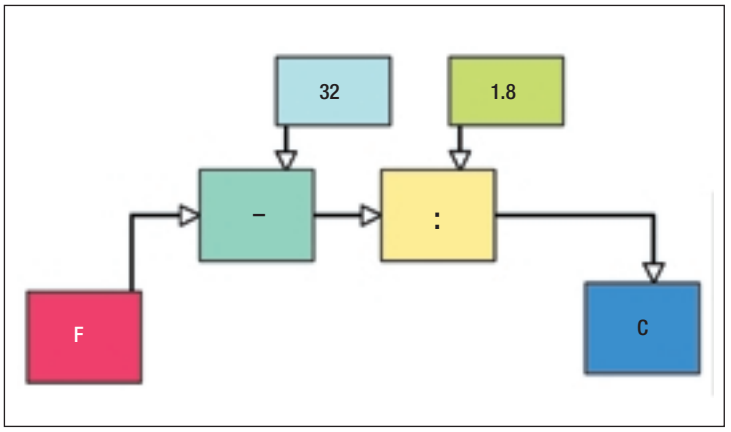


FIGURA 4
Convertitore da gradi Fahrenheit a Celsius

Sempre nell'ottica di aumentare la trasparenza all'utente e facilitargli il compito, sarebbe utile poter visualizzare le conseguenze di un comando dato selezionando e attivando una particolare icona.
In definitiva, ogni utente, o classe di utenti, potrebbe disporre di un modello di sistema, a seconda di ciò che gli serve conoscere del suo sistema. In questo modo, si realizzerebbe una trasparenza personalizzata.

6. CONCLUSIONI

Abbiamo visto tre diversi approcci (più, molto sommariamente, un quarto finale) diretti a superare gli attuali limiti delle interfacce e migliorare l'interazione uomo-macchina, rendendola fluida, naturale, efficiente ed effettiva.
I vari approcci non sono in contraddizione fra loro, anzi possono condividere alcune proprietà, cioè un sistema può essere contemporaneamente emotivo ed indossabile, emotivo ed ubiquo ecc.. Si tratterà quindi di sperimentare, in situazioni reali e non più in laboratorio, quali sono gli effetti, positivi o negativi, che questi approcci sono in grado di fornire. Non sempre la tecnologia ha migliorato la vita, ma è bene provare e riprovare per saperne di più.

Bibliografia

[1] Artificial Intelligence Topics, http://www.aaai.org/AITopics/html/emotion.html#good

[2] Fishwick P.: Enhancing experimental and subjective qualities of discrete structure representations with aesthetic computing. *Journal of Visual Languages and Computing*, Vol 16/5, 2005, p. 406-427.

[3] Georgia Tech, Ubiquitous Computing Group, http://home.cc.gatech.edu/ubicomp/52

[4] Mann, Steve, Univ. of Toronto, 1998, http://about.eyetap.org/fundamentals/

[5] Don Norman, http://www.acm.org/ubiquity/-views/v4i45_norman.html

[6] Oregon University, http://www.cs.uoregon.edu/research/wearables/

[7] Picard Rosalind, http://web.media.mit.edu/~picard/

[8] Picard2 Rosalind, http://affect.media.mit.edu/

[9] Sekuler R., Blake R.: *Perception*. McGraw Hill, New York, 1994,

[10] Stanford University, http://plato.stanford.edu/entries/emotion/

[11] Tanimoto Steve: *Transparent Interfaces: Models and Methods*. Workshop on Invisible and Transparent Interfaces, Aaron Quigley, Steven Tanimoto, organizers, Gallipoli, (Italy), 2004.

[12] Wearables, http://www.media.mit.edu/wearables/)

[13] Weiser Mark: *Mediated Spaces*, http://www.research.ibm.com/journal/sj/384/mark.html

[14] Zhang Png, Li Na: The Importance of Affective Quality. *Com of ACM*, Vol. 48, n. 9, sep. 2005, p. 105-108.

[15] Samudra Sengupta, Takayuki Dan Kimura, Ajay Apte: *An Artistst's Studio: A metaphor for modularity and Abstraction in a Graphical Diagramming Environment*. Proc. IEEE Symp. on Visual Languages, St Louis, Misouri, 1994, p. 128-136.

[16] Paternò F.: Interazione uomo-computer: un'introduzione. *Mondo Digitale*, n. 12, 2004.

STEFANO LEVIALDI è professore ordinario di informatica alla Università di Roma "La Sapienza" e Decano del Dipartimento di Informatica. Ha pubblicato oltre 230 lavori (con più di 120 co-autori) in convegni internazionali e su riviste scientifiche, sulla elaborazione di immagini, sui linguaggi visuali, e sulla interazione uomo-calcolatore, argomento che svolge in due corsi universitari. Levialdi è fondatore (1990) e co-editore del *Journal of Visual Languages and Computing* (Elsevier Press) ed editore associato di varie riviste scientifiche, organizzatore di oltre 20 convegni internazionali come Advanced Visual Interfaces (ogni due anni dal 1992), Interact (2005 a Roma). Infine è IEEE Fellow (1988) e dal 1996 IEEE Life Fellow.
E-mail: levialdi@di.uniroma1.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 caso della Rete dei Concessionari Alfa Romeo

Roberto Bellini

1. Introduzione

Viene presentato di seguito il risultato di un progetto pilota sviluppato tre anni fa per la rete distributiva di Alfa Romeo; il progetto ha coinvolto alcune decine di Concessionari, imprenditori commerciali con imprese che vanno dai 5-10 addetti a qualche decina.
Il progetto nasceva dalla consapevolezza del committente, il direttore della divisione, sui problemi di debolezza della rete distributiva, dovuti a varie vicissitudini nella storia aziendale.
Le richieste alla società di consulenza incaricata di sviluppare il progetto erano fortemente orientate:

- sviluppare strumenti e metodologie di marketing on line a supporto della attività commerciale dei concessionari scelti per il pilota;
- supportare l'introduzione di tali strumenti e metodologie con un forte affiancamento operativo locale e continuo sul concessionario.

Gli obiettivi posti dal committente erano molto stringenti e finalizzati, ancorché si trattasse di un progetto pilota, ad ottenere risultati concreti a brevissimo termine; in particolare l'ipotesi di lavoro era quella mettere il cliente al centro della attività del concessionario identificando target di clienti mirati a cui offrire la partecipazione ad eventi tramite campagne di Direct Marketing, per generare contatti e relazioni con prospect interessati, accogliere e profilare i prospects sul punto vendita, capire i loro bisogni e generare trattative da portare avanti differenziando argomentazioni di vendita, prezzi e condizioni di vendita, fino alla chiusura del contratto di acquisto di un auto.
L'esigenza da soddisfare era quella di migliorare la quota e la penetrazione in un mercato ipercompetitivo, aumentando la capacità di vendita di ciascun concessionario e migliorando l'efficacia di tutta la rete distributiva, a partire dalla messa in comune, fra concessionaria e costruttore, dei nominativi dei prospect e delle informazioni sul loro rispettivo profilo.
La soluzione adottata dalla società di consulenza è stata quella di sviluppare un portafoglio di servizi per il trade molto sofisticato e metterlo a disposizione dei concessionari in termini condivisi, anche dal punto di vista dei costi.

2. Storia dei concessionari e della rete Alfa Romeo

I concessionari nascono come ex venditori diretti della casa automobilistica produttrice; la necessità di ridurre, e possibilmente variabilizzare, i costi di vendita spinge la casa auto-

mobilistica produttrice ad imboccare decisamente la strada di costruire una rete di distribuzione (i concessionari) trasformando i suoi migliori venditori in imprenditori che si specializzano nella vendita delle auto nuove e usate e nella assistenza tecnica del parco auto in circolazione.
Questa soluzione organizzativa per la vendita (la costruzione di una rete di vendita indiretta da parte della casa costruttrice) viene adottata praticamente da tutte le case costruttrici a partire dagli anni 70; con la introduzione di questo tipo di intermediario, la casa costruttrice si alleggerisce da una parte dei costi di vendita ma dall'altra perde completamente il contatto diretto con il cliente finale e, in generale, con i mercati locali; il concessionario diventa infatti, inevitabilmente, il presidio di un mercato locale costituito dai clienti finali e dai concessionari di case costruttrici concorrenti che operano sul territorio/zona assegnato.
Per proteggere il proprio investimento commerciale, le case costruttrici tendono a definire rapporti contrattuali con i migliori concessionari sempre più orientati alla "esclusività"; in realtà i concessionari più capaci dal punto di vista imprenditoriale tendono invece a costruire una presenza sul mercato locale sempre più forte, attraverso accordi con più case costruttrici: l'obiettivo di questi imprenditori è quello di avere un portafoglio di offerta sempre più completo su un territorio che gradualmente imparano a conoscere molto bene e su cui identificano target differenziati che possono essere meglio serviti con auto di marche e modelli diversi.
Specificamente, la rete distributiva di Alfa Romeo, dopo l'acquisizione del marchio da parte di Fiat, viene incorporata nella rete distributiva di Fiat, in cui convivono tutti e tre i marchi della casa costruttrice torinese: Fiat, Lancia e Alfa.
Questa situazione genera nei concessionari, negli ultimi 10 anni, una situazione in cui il venditore diventa lo stesso per tutti e 3 i marchi, con grave riduzione della efficacia di vendita; il marchio Alfa, che ha la maggiore "personalità" e che nel tempo ha creato un vero e proprio mito presso una significativa parte della clientela italiana ed estera, viene annegato negli altri marchi facendo perdere la "personalità" di cui sopra.
Nel frattempo, a luglio del 2002 la Commissione Europea adotta una profonda riforma che definisce una nuova organizzazione della distribuzione automobilistica e dei servizi post-vendita in Europa nei prossimi anni, con l'obiettivo di stimolare la concorrenza e trovare un equilibrio tra i vantaggi dell'industria e i benefici del consumatore; la direzione della Divisione Alfa Romeo decide di sfruttare la introduzione della direttiva europea per cercare di perseguire due obiettivi:

- recuperare una modalità distintiva nella gestione delle vendite delle auto sotto il proprio marchio;
- recuperare un più diretto controllo sui mercati locali, mettendo a disposizione della rete dei concessionari una strumentazione di gestione del portafoglio clienti che sia condiviso fra i concessionari della rete e che permetta alla direzione commerciale di avere dati sulle vendite e sulle caratteristiche della clientela.

3. Organizzazione e processo di vendita di un concessionario d'auto

I principali processi nella Concessionaria comprendono *vendita* e *post-vendita*; sulla vendita il concessionario guadagna le sue commissioni come previsto dal contratto con la casa costruttrice, ma anche il *post-vendita* rappresenta una fonte di introiti molto elevata; i servizi post-vendita infatti comprendono l'assistenza attraverso l'officina meccanica (manutenzioni ordinarie, revisioni obbligatorie per legge, riparazioni), la vendita dei ricambi, la vendita di accessori (portapacchi, car stereo, prodotti derivati ecc.) e la gestione del Circuito di Garanzia.
L'aumento delle vendite delle **auto nuove** impatta in maniera decisamente elevata sia sul fatturato dei prodotti che su quello legato ai servizi; infatti alla crescita del **nuovo** corrisponde l'aumento del parco circolante (ovvero il numero di veicoli immessi su strada da ogni concessionario), da cui deriva l'aumento del fatturato di officina (con la crescita del numero degli interventi di manutenzioni) e l'aumento del fatturato di magazzino, con una crescita delle occasioni di contatto con il cliente.
Il processo che viene messo sotto analisi nel progetto pilota è il processo di vendita nelle

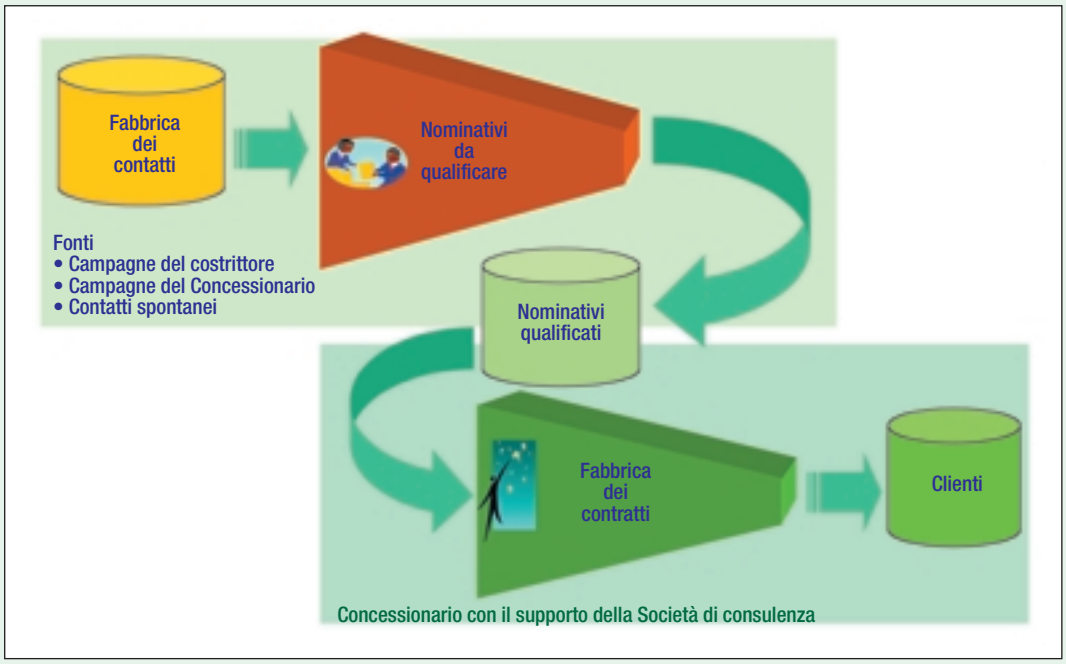


FIGURA 1
La Fabbrica dei Contatti e dei Contratti

sue due fasi principali che vengono chiamate rispettivamente (Figura 1).

❑ *Fabbrica dei Contatti*: il flusso prevede che per generare contatti sia necessario impostare e lanciare un certo numero di campagne all'anno a livello nazionale e locale, per la qualificazione sistematica di nominativi interessati all'offerta della casa costruttrice e resi disponibili nel DB Nominativi Qualificati;

❑ *Fabbrica dei Contratti*: il flusso prevede che i nominativi qualificati vengano estratti dal prezioso Data Base NQ uno ad uno dai venditori per diventare oggetto di trattativa e quindi, ancora con una attività molto mirata, di farli diventare clienti, in percentuale significativa.

Nel Processo di Vendita le principali figure professionali coinvolte sono:

❑ il Titolare, cioè l'imprenditore, che nasce comunque come venditore e continua a svolgere questa funzione sopratutto sui grandi clienti, assumendo spesso anche il ruolo di Responsabile Commerciale della Concessionaria, fino a che le dimensioni lo permettono;

❑ il Venditore, che è poi quello che ha il contatto diretto prima con potenziale cliente (prospect) e, una volta conclusa positivamente la trattativa, con il cliente;

❑ il Collaudatore, che assume un ruolo significativo nella vendita quando la trattativa con il cliente tocca gli aspetti tecnici della vettura o il cliente accetta di fare il Test Drive.

Lo strumento fondamentale progettato e messo a disposizione dei venditori è costituito dalla Scheda Cliente; la Scheda deve essere compilata in parte (per la sezione anagrafica in particolare) dal potenziale cliente, che deve anche dare la liberatoria all'utilizzo dei suoi dati a norma della legge sulla privacy, e per la parte più significativa da parte del venditore. Sostanzialmente la scheda è divisa nelle seguanti sezioni:

❑ Conoscere il cliente: questa sezione comprende
- i dati anagrafici e le modalità di contatto;
- il profilo sociodemografico (professione, titolo di studio, composizione della famiglia);
- parco auto della famiglia e motivazioni della sostituzione;
- la liberatoria.

❑ Capire il cliente: la sezione comprende:
- il nome del venditore;
- le attenzioni e le esitazioni: in particolare si cerca di catturare indicazioni su quali sono le auto che il potenziale cliente ha in mente e quale sarebbe interessato ad acquistare (non solo della marca rappresentata dal concessionario) e quindi la migrazione attesa dalla auto posseduta a quella nuova;
- i valori associati all'auto (come mezzo, come status symbol ecc.) e le principali funzioni d'uso indicate dal potenziale cliente;
- I principali fattori che determinano la scelta da parte del prospect in termini tecnici, economici ed emozionali.

❑ Tracciamento del contatto e della trattativa: questa sezione comprende:

- le argomentazioni di vendita usate dal venditore e le obiezioni espresse dal *prospect*;
- i fattori che spingono/frenano l'orientamento del *prospect* verso il modello proposto dal venditore/di interesse del *prospect*;
- lo sviluppo della relazione (visite, telefonate, scambio di documentazione) e la valutazione del venditore sulle probabilità di vendita al *prospect*;
- la conclusione della trattativa, in positivo (modello venduto, sconto) o in negativo (il cliente non si fa più vivo).

La Scheda Cliente costituisce il supporto alla gestione del prospect; tutti i dati rilevati vengono imputati dal venditore stesso o da un Assistente alla Relazione con il Cliente nel Customer Data Base reso disponibile al singolo venditore; il venditore ha immediatamente un beneficio da questi dati; gli viene stampata una scheda che riporta indicazioni su clienti che vengono individuati dal sistema di Customer Intelligence come simili a quello che il venditore sta seguendo, accompagnato da indicazioni su quali sono state le argomentazioni di vendita che hanno avuto successo e su quali vanno assolutamente evitate; in altre parole il sistema informativo mette a disposizione di ogni venditore tutta l'esperienza che giorno per giorno viene catturata da tutti i venditori della rete di vendita dei concessionari (Figura 2).

Nel dettaglio, il venditore cerca di generare l'opportunità di vendita attraverso argomentazioni di vendita specifiche per ciascun potenziale cliente; l'obiettivo intermedio, fondamentale in questa fase del processo di vendita, è quello di arrivare alla "prova su strada" dell'auto (Test Drive): l'esperienza insegna infatti che se il prospect accetta di fare il Test Drive, la conclusione della trattativa in termini positivi diventa molto più probabile; al Test Drive segue una ulteriore fase di trattativa sulle condizioni di vendita e sul prezzo, molto spesso legato alla valorizzazione dell'usato che il venditore deve ritirare. La trattativa si chiude positivamente solo quando il prospect firma l'ordine di acquisto e comunque, anche se la trattativa si interrompe, segue una ultima fase per la valutazione dei risultati del processo di vendita.

La messa a fattore comune dei profili dei prospect e dei clienti, dei contenuti di ciascuna fase del processo di vendita e dei risultati (in termini di successo e insuccesso della rete di vendita) permette di costruire un vero e proprio Knowledge Data Base finalizzato alla vendita messo a disposizione dei venditori in modo da poterli guidare in ogni singola azione commerciale verso un tasso di successo crescente.

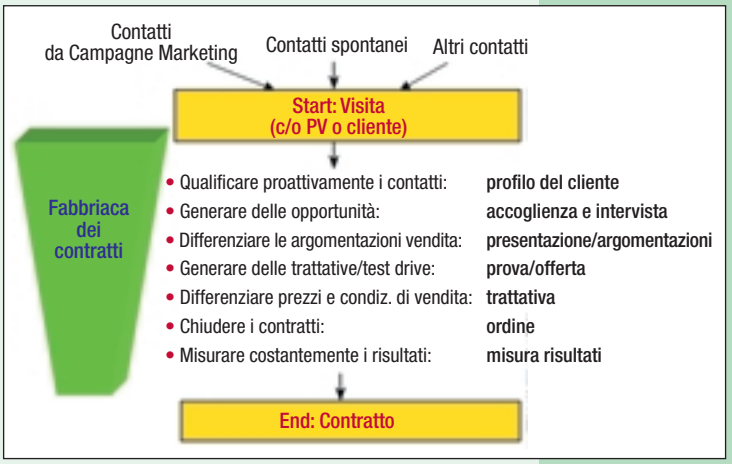


FIGURA 2

Fabbrica dei Contratti: le fasi supportate da profilo e modello interpretativo

4. Struttura fondamentale del sistema informativo di supporto

Il Sistema Informativo di supporto alla Rete di Vendita di Alfa Romeo era costituito da una Fabbrica di Servizi (CONNATE, sviluppata dalla società di consulenza che ha avuto l'incarico di sviluppare il progetto) e basata su 4 funzioni fondamentali:

❑ *Business Intelligence* (BI): funzione di analisi e di reporting sulle caratteristiche della clientela e sull'andamento del mercato;

❑ *Campaign Management* (CM): funzione di gestione delle campagne di Direct Marketing per la acquisizione e qualificazione dei nominativi dei prospect;

❑ *Sales Force Automation* (SFA): funzione di gestione e supporto alla attività dei venditori, che interagisce con il Customer Data Base, alimentato sia dal Data Base dei Prospect Qualificati che dai dati raccolti attraverso la Scheda Cliente;

❑ *Key Performance Indicators* (KPI): funzione di misura e analisi delle prestazioni di tutto il sistema.

Lo schema mostrato nella figura 3 rappresenta il modo in cui i vari flussi informativi vengono integrati fra di loro: la Fabbrica dei Contatti, le Campagne promosse dal Consulente Allaxia e quelle promosse localmente dai singoli Concessionari sono tutte finalizzate ad alimentare il Data Base dei Prospect Qualificati: la funzione di Campaign Management traccia l'evoluzione di tutte le campagne lanciate permettendo di gestire le eventuali sovrapposizioni (evitare che lo stesso nominativo venga sollecitato da campagne diverse) e di ottimizzare gli impegni su tutta la rete; fornisce inoltre un supporto alla attività di affiancamento dei Concessionari da parte della rete dei Business Partner della Società di Consulenza sia nelle Campagne di qualificazione dei Prospect che per la Gestione della Scheda Cliente utilizzata dai Venditori come supporto nelle varie Trattative.
La gestione della Scheda Cliente viene supportata dal Sistema Informativo con la funzione SFA, che permette di sviluppare sia il Tracciamento della Relazione con il Cliente/Prospect che la Profilatura del Cliente; i dati raccolti nella Scheda Cliente sono forniti o in autocompilazione dai Prospect oppure compilati dai Venditori al termine di ogni incontro con il cliente: è infatti fondamentale che il Prospect non si senta "intervistato" ma piuttosto percepisca il Venditore come uno specialista al suo servizio per risolvere il problema della migliore soluzione di motorizzazione.
L'importanza della Profilatura del Cliente per il Venditore è tutta nei due seguenti aspetti:

- individuare le "chiavi di successo" per la gestione trattativa, identificando in base ai dati inseriti nella scheda cliente il **profilo di riferimento** del contatto;
- fornire al Venditore le **argomentazioni di vendita** e le **obiezioni** critiche per il profilo di

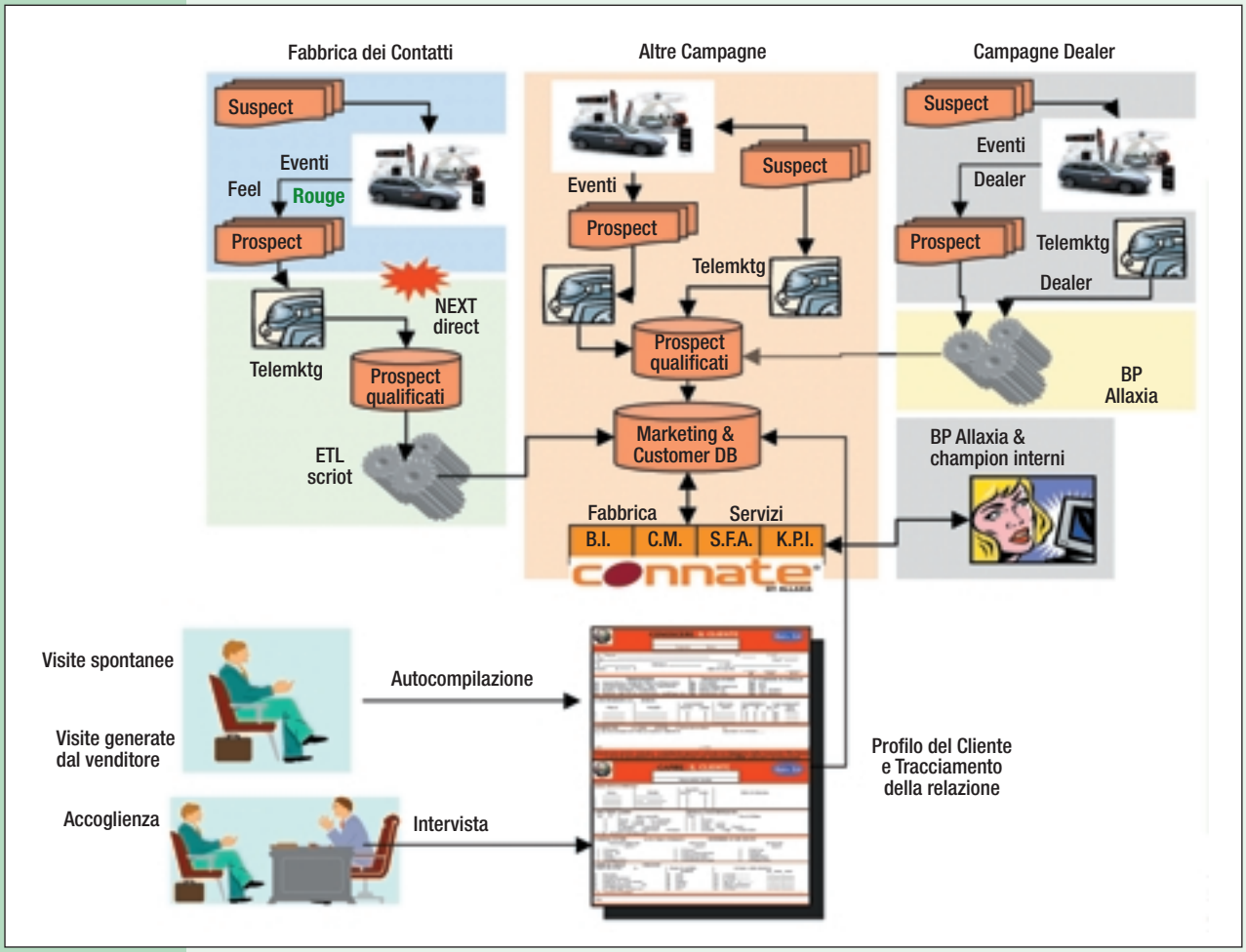


FIGURA 3
L'integrazione dei flussi informativi

clienti a cui il contatto in esame appartiene, quale derivazione dell'esperienza comune di tutti i venditori coinvolti; ricordiamo in proposito che il Profilo è caratterizzato dalla principale funzione d'uso (per lavoro, per vacanze,...) dell'auto, dai valori associati all'auto (sicura, confortevole,....), dalla classe / marca di origine e destinazione (esempio, full liner) e dalla classe / segmento di origine e destinazione (esempio, C, D,...).
Dal punto di vista funzionale, il Profilo del Cliente integra le informazioni di marketing (schede contatto, interviste ... spesso volatili), con quelle di vendita e (potenzialmente) con le informazioni gestionali (contratti, immatricolazioni, fatture, pagamenti, finanziamenti ecc.), può essere gestito sia manualmente (schedario...) che (meglio) on line, genera il Customer Database ed abilita tutte le funzionalità di analisi (Business Intelligence), gestione campagne (Campaign Management), supporto alla vendita (Sales Force Automation), misura e controllo delle performance (KPI).
Infine, dal punto di vista del Sistema informativo, non dimentichiamo l'importanza della funzione di misura e controllo: nella tabella 1 sono riportati i vari "Indicatori di Performance": gli indicatori sono articolati per fase di lavoro analitica e per ognuno di essi è indicato l'algoritmo con cui sono costruiti, lo strumento e la fonte di rilevazione.
Sono infine resi disponibili due indicatori di sintesi come

- ❑ Produttività di Marketing: Visite su Contatti (algoritmo B/D)
- ❑ Produttività delle Vendite: Ordini su Visite (algoritmo I/(D+E))

5. Risultati della Innovazione introdotta, economici e gestionali

I risultati della innovazione sono molto significativi su tutti e tre i livelli di responsabilità identificati:

a. livello concessionario

- ogni venditore è in grado di conoscere lo stato di avanzamento delle proprie attività, con il supporto anche di un benchmark che gli permette di confrontarsi con il migliore risultato e con il risultato medio;

ID	Variabile	Articolata per:	Indicatori analitici	Algoritmo	Strumenti di rilevazione	Fonte di rilevazione
Fabbrica dei contatti						
A	Liste	Campagne	N.A.	N.A.	Registro campagne	Next, Allaxsia, Concessionario
B	Contatti/Coupon/Schede	Campagne	Redemption 1° livello	B/A	Registro campagne	Concessionario
C	Contatto Qualificato	Campagne	Redemption 2° livello	C/B	Registro campagne	Concessionario
Fabbrica dei Contratti						
D	1ª Visita (da Prospect)	Campagne	Esito campagna	D/C	Profilo	Concessionario
E	1ª Visita (spontanea)	Venditore	Visite spontanee		Profilo	Concessionario
F	2ª Visita	Venditore	Interesse	F/(D + E)	Profilo	Concessionario
G	Test-drive	Venditore	Interesse Prova Auto	G/(D + E)	Profilo	Concessionario
H	Offerta / Quotazione	Venditore	Interesse offerta	H/G	Profilo	Concessionario
I	Ordine	Venditore	Conversione	I/H	Sist. Gestionale	Concessionario
J	Immatricolazione / registrazione	Venditore	Registrazione		Sist. Gestionale	Alfa Romeo

TABELLA 1
Gli indicatori di prestazione per la Rete di Vendita

• ogni responsabile vendite e/o titolare sa, periodo per periodo il risultato delle attività commerciali e come varia nel tempo la segmentazione del proprio mercato;

b. livello rete dei concessionari

• ogni funzionario e responsabile territoriale della rete è in grado di conoscere a sua volta lo stato delle trattative sul proprio territorio e il relativo benchmark territoriale;

• il responsabile della rete a livello di divisione conosce lo stato delle trattative e può formulare con notevole maggiore precisione rispetto alla situazione precedente il forecast delle vendite;

c. livello divisione

• la direzione marketing e vendite è in grado di formulare le previsioni di chiusura del periodo con molta maggiore precisione e con molto maggiore livello di dettaglio e quindi può alimentare il relativo piano di produzione con un errore notevolmente inferiore;

• la direzione ai vari livelli di specializzazione conosce l'andamento degli ordini e del mercato e le sue evoluzioni momento per momento.

Dal punto di vista dei costi, il progetto pilota ha sviluppato i suoi servizi per 30 concessionari, con un costo per concessionario di 10.500 € al mese, tutto compreso per una durata di 6 mesi; le principali voci del progetto sono:

❑ sviluppo innovazione e ridisegno dei processi di Marketing e Vendita, che assorbe un 8-9% del budget totale;

❑ supporto alla vendita, che assorbe un 80% del budget totale;

❑ project management, che assorbe il restante.

Il costo della strumentazione tecnologica è di qualche percento, annegato nelle prime due voci.

Al termine del progetto pilota era prevista una riduzione dei costi totali per concessionario dell'ordine del 30%, su una platea di almeno 80-90 concessionari, con l'obiettivo di permettere a ciascun concessionario di spesare in proprio la configurazione di servizi più adatta per i suoi obiettivi di business.

6. Conclusioni

I risultati ottenuti dal progetto per quanto riguarda il miglioramento del processo di vendita dei concessionari sono stati:

❑ identificazione e qualificazione di liste di prospect;

❑ attrazione di prospect presso il punto vendita;

❑ miglioramento della capacità di convinzione dei prospect nell'acquisto di nuovi prodotti e utilizzo dei servizi della concessionaria;

❑ ampliamento del Prospect Data Base, raccogliendo dati per successive azioni di marketing mirate;

❑ miglioramento della conoscenza del proprio mercato potenziale.

La soluzione realizzata si è articolata nei seguenti componenti:

❑ una "doppia" rete di strumenti *on-line* e di competenze locali (Business Partner) per il supporto della rete dei Concessionari;

❑ il supporto alla rete di vendita per aumentare il *sell out* tramite:

• supporto a campagne esistenti;

• sviluppo e implementazione di nuove campagne locali *one-to-one* a breve termine (qualificazione contatto);

• il miglioramento della produttività di vendita: da contatti qualificati a contratti conclusi;

❑ l'analisi e il miglioramento dei processi di Marketing e Vendita della Rete di supporto e di ciascun Concessionario, migliorando l'orientamento al cliente.

Roberto Bellini è docente nel corso di Concept Dvelopment & Industrial Engineering presso il Politecnico di Milano e docente di Marketing e Gestione della Relazione con il cliente nell'ambito del MIP, con una focalizzazione sulla innovazione nelle reti di imprese.
Presiede la Sezione AICA di Milano ed è responsabile per AICA del progetto eCCO-eCertification and COmpetence Tool, sviluppato con Federcomin e Fondazione Politecnico.
E-mail: roberto.bellini@polimi.it

Chiara Francalanci è professore associato di Sistemi Informativi al Politecnico di Milano. Ha scritto numerosi articoli sulla progettazione e sul valore economico delle tecnologie informatiche, svolto attività di ricerca e consulenza nel settore finanziario e manifatturiero sia in Italia sia presso la Harvard Business School ed è editor del Journal of Information Technology.
E-mail: francala@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.

L'informatica nella Giustizia: il processo telematico

1. Introduzione

La punta di diamante del Piano triennale per l'informatica 2004/2006 della Giustizia[1] è rappresentata dall'attuazione del cosiddetto "Processo telematico" grazie al quale i difensori, il giudice e gli operatori della giustizia potranno utilizzare gli strumenti informatici e telematici per la formazione, il deposito, lo scambio, la comunicazione e la notificazione degli atti del procedimento civile mediante documenti informatici.
In buona sostanza in un futuro, si auspica non lontano, gli atti processuali potranno essere digitalizzati secondo la disciplina dettata dal D.P.R. 13 febbraio 2001, n.123 e in conformità con le regole tecniche stabilite dal Decreto del Ministero della Giustizia del 14 ottobre 2004.
I principali benefici si manifesteranno soprattutto in termini di rapidità e risparmio nonchè, in particolare, in una riduzione dei tempi morti, in un'estensione dell'orario di funzionamento degli uffici virtuali e nella possibilità di consultare *on line* il fascicolo elettronico e lo stato della causa in linea con quanto stabilisce anche il recente codice dell'amministrazione digitale in tema di governo elettronico.
Vediamo quali sono i principali soggetti che interagiranno in questo flusso documentale del giudizio civile, nonché le principali fasi del suo funzionamento.

2. Il Sistema Informatico Civile

Il *Sistema Informatico Civile* (S.I.CI.) è il sottoinsieme del dominio giustizia (vale a dire delle risorse hardware e software che compongono la Rete Unica della Giustizia[2]) dedicato al processo civile, attraverso il quale viene veicolata l'attività di trasmissione, comunicazione o notificazione dei documenti informatici.
Secondo il D.M. 14 ottobre 2004 il S.I.CI. risulta organizzato su tre livelli:
1. la Direzione Generale per i Sistemi Informativi Automatizzati (DGSIA) del Ministero della Giustizia, sita in Roma, cura il funzionamento del Gestore Centrale;
2. il Coordinamento Interdistrettuale dei Sistemi Informativi Automatizzati (CISIA) si occupa del funzionamento dei Gestori Locali degli uffici di competenza;
3. il dirigente amministrativo di ciascun ufficio giudiziario è responsabile, per l'ufficio di propria competenza, della consistenza dei dati (Figura 1).

Nel suo insieme il Sistema Informatico Civile permette di individuare univocamente ciascun procedimento giudiziario, identificando tutti i soggetti che inseriscono, modificano o trasmettono dati. Sono abilitati all'accesso al S.I.CI. i magistrati, i loro ausiliari, il personale degli uffici giudiziari, e gli avvocati; il sistema sarà in grado di fornire i servizi di cancelleria per via telematica.

[1] Reperibile presso il sito istituzionale del Ministero della Giustizia all'indirizzo: www.giustizia.it.

[2] La R.U.G. a sua volta fa parte della R.U.P.A., la Rete Unitaria per la Pubblica Amministrazione.

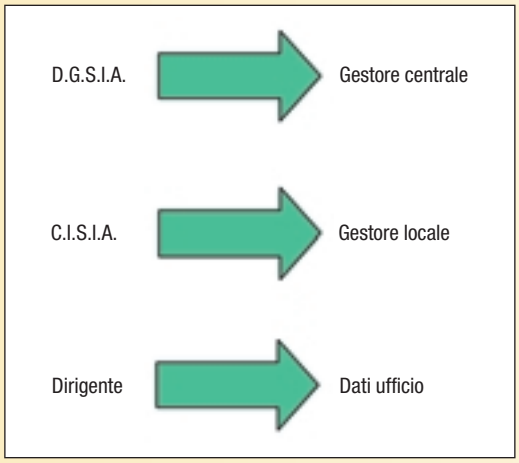


FIGURA 1
Sistema informatico civile

3. Il Punto d'Accesso

Il *Punto di Accesso* (P.d.A.) viene definito come la struttura tecnico-organizzativa che fornisce agli avvocati i servizi di connessione al gestore centrale e di trasmissione telematica dei documenti informatici relativi al processo, nonché la casella di posta elettronica certificata.
I P.d.A. possono essere attivati e gestiti esclusivamente dai seguenti soggetti:
a. i Consigli degli Ordini degli avvocati, ciascuno limitatamente ai propri iscritti;
b. il Consiglio nazionale forense;
c. l'Avvocatura dello Stato, le amministrazioni statali o equiparate e gli enti pubblici;
d. il Ministero della Giustizia;
e. i soggetti privati che abbiano forma di società per azioni, nonché capitale sociale e requisiti di onorabilità già previsti per i certificatori delle firme digitali.
Il D.M. 14 ottobre 2004 pone a carico dei Punti d'Accesso una serie di adempimenti di non poco conto, anche sotto il profilo tecnico, di talché è ragionevole presumere che per gli avvocati, almeno inizialmente, tale funzione verrà svolta dal Consiglio Nazionale Forense o dai Consigli degli Ordini più numerosi.
Ogni avvocato potrà avvalersi di un solo Punto di Accesso, al quale spetterà il compito di sottoporlo alla procedura di autenticazione (cioè di identificazione), necessaria ai fini dell'accesso al Sistema Informatico Civile; inoltre il P.d.A., se è gestito da un Consiglio dell'ordine degli avvocati o dal Consiglio Nazionale Forense, dovrà procedere alla certificazione del difensore (mediante la quale attesta il possesso della qualifica che legittima all'esercizio della difesa e l'assenza di cause ostative allo svolgimento della medesima).
Tutte le comunicazioni tra la postazione informatica dell'avvocato e il suo Punto di Accesso avvengono in modalità sicura (SSL versione 3 con chiave a 1024 bit); parimenti ciascun Punto di Accesso stabilisce con il Gestore Centrale un canale di comunicazione che consente la reciproca autenticazione e riservatezza.
Il Punto di Accesso ha un ruolo molto importante in riferimento alla sicurezza informatica, che si estrinseca anche nel verificare l'assenza di virus informatici in ogni messaggio in arrivo e in partenza.

4. Il Gestore Centrale

Il *Gestore Centrale* (G.C.) è la struttura tecnico-organizzativa appartenente al dominio giustizia che, fornendo accesso al Sistema Informatico Civile, costituisce l'unico punto di interazione tra il S.I.CI. stesso e gli avvocati; questi ultimi, come detto, possono accedere ai servizi del Gestore Centrale esclusivamente attraverso il proprio Punto di Accesso.
Il G.C. è attivo presso il Ministero della Giustizia, viene gestito dalla Direzione Generale per i Sistemi Informativi Automatizzati (DGSIA) ed è materialmente ubicato a Napoli.
Il Gestore Centrale fornisce due servizi essenziali: la consultazione del Sistema Informatico Civile e la trasmissione telematica degli atti, venendosi in entrambi i casi a trovare in posizione mediana tra i Punti d'Accesso e i Gestori Locali e dovendo, conseguentemente, smistare i relativi flussi di dati (richieste, informazioni, ricevute, atti, documenti) da una parte all'altra d'Italia.
Il G.C. dialoga con i Punti d'Accesso attraverso un canale di comunicazione sicuro, mentre con i Gestori Locali utilizza esclusivamente la Rete Unica della Giustizia.
Al Gestore Centrale, tra i vari incombenti, spetta da un lato la funzione di associare automaticamente ai documenti informatici pervenuti un'attestazione temporale (data, ora e minuto[3]) del-

[3] L'attestazione temporale fa riferimento alla scala di Tempo Universale Coordinato UTC (IEN) determinata ai sensi della L. 11 agosto 1991, n. 273 e del successivo D.M. 30 novembre 1993, n. 591, pubblicato nella G.U. 15 febbraio 1994, n. 37.

la loro ricezione e, dall'altro, il compito di tenere l'elenco di tutti gli indirizzi elettronici attivati dai punti di accesso.
Per quanto attiene ai profili di sicurezza informatica, anche il G.C. ha l'onere di verificare l'assenza di virus informatici in ogni messaggio in arrivo e in partenza.

5. Il Gestore Locale

Il *Gestore Locale* (G.L.), invece, è la parte del sistema informatico di ciascun ufficio giudiziario che si occupa della trasmissione telematica dei documenti informatici processuali tra l'ufficio stesso e il Gestore Centrale.
I Gestori Locali sono attivi presso gli uffici giudiziari sul territorio e il loro funzionamento viene curato, attraverso l'amministratore di sistema, dai Coordinamenti Interdistrettuali dei Sistemi Informativi Automatizzati (CISIA).
Per garantire la riservatezza delle comunicazioni, il G.L. utilizza un meccanismo di crittografia tramite il quale vengono cifrati i dati in uscita e decifrati quelli ricevuti; inoltre verifica automaticamente, con il controllo della firma digitale, l'autenticità e l'integrità di ogni documento informatico ricevuto[4].
Il Gestore Locale, inoltre, ha il compito di controllare che ogni file rispetti il formato stabilito dalle regole tecniche e che non contenga virus o programmi nocivi per il computer (Figura 2).

6. La Casella di Posta Elettronica Certificata per il Processo Telematico

La *Casella di Posta Elettronica Certificata per il Processo Telematico* (C.P.E.C.P.T.) è l'indirizzo di posta elettronica certificata utilizzato per il processo telematico.
Secondo quanto previsto dalle regole tecniche, per poter utilizzare i servizi di trasmissione telematica dei documenti informatici gli avvocati devono disporre di un (solo) indirizzo elettronico e della relativa casella di posta elettronica, forniti e gestiti dal punto di accesso.
La C.P.E.C.P.T. risulta abilitata a ricevere esclusivamente messaggi provenienti da altri punti di accesso (cioè da altri avvocati) e dal Gestore Centrale (quindi provenienti da uffici giudiziari) e quindi non potrà essere impiegata per usi privati.
La C.P.E.C.P.T. garantisce la ricezione dei messaggi e la loro disponibilità per trenta giorni; successivamente il messaggio viene archiviato e sostituito da un avviso.
Il servizio di posta elettronica certificata restituisce al mittente una ricevuta di avvenuta consegna per ogni documento informatico reso disponibile al destinatario, cui è associata l'attestazione temporale.
Come accennato, ogni Punto di Accesso attiva un registro contenente l'elenco di tutti gli indirizzi elettronici emessi, revocati o sospesi e lo trasmette al Gestore Centrale presso il quale viene tenuto l'elenco generale di tutti gli indirizzi elettronici attivati dai Punti di Accesso, in maniera fruibile a tutti i soggetti abilitati.
Ciascun difensore è tenuto a comunicare il proprio indirizzo elettronico (relativo alla C.P.E.C.P.T. rilasciata dal punto di accesso) sia al Consiglio dell'ordine di appartenenza, sia alla cancelleria.
A norma del D.P.R. 123/01 solamente a tale indirizzo (e non ad altri eventualmente utilizzati dall'avvocato) possono essere validamente effettuate le comunicazioni con biglietto di cancelleria, nonché le notificazioni degli atti in via telematica.

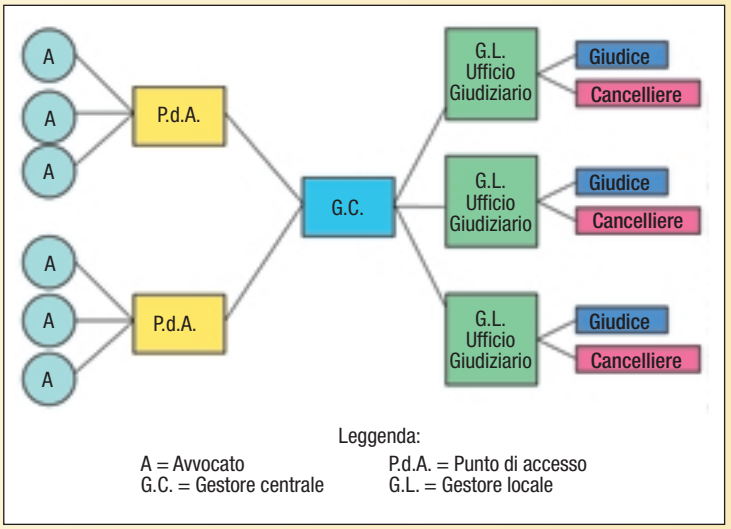


FIGURA 2
Schema generale

7. Aspetti applicativi

L'impiego degli strumenti informatici e telematici nel processo civile richiede la dotazione di una "Stazione di lavoro" (detta anche

[4] Si veda Mondo Digitale marzo 2005, "*La firma digitale tra crittografia e diritto*".

"Consolle") che permetta di redigere un atto, firmarlo digitalmente, crittografarlo e, infine, spedirlo.
La postazione di lavoro, allo stato attuale della tecnica, deve possedere i seguenti requisiti minimi: processore Pentium III 500 MHz; memoria RAM 128 MB (consigliata 256 MB); sistema operativo Windows 2000 o Windows XP; software applicativi MS Word 97/2000/XP; Internet Explorer 4.0 o superiore.
Inoltre risulta necessario utilizzare un ambiente di redazione (chiamato "Redattore di Atti"), che si integra con il tradizionale software di videoscrittura Microsoft Word e permette la successiva trasformazione dell'atto in un formato XML[5]; in questo modo il sistema informatico che riceve l'atto può riconoscerne facilmente i campi semantici.
Al documento viene apposta la firma digitale mediante la smart card rilasciata da un certificatore accreditato (per esempio, dal Consiglio Nazionale Forense), pertanto il file generato si presenterà con l'estensione ".p7m" tipica della struttura PKCS#7.
Da ultimo l'atto viene crittografato in modo da risultare leggibile solamente dal destinatario[6] e spedito tramite il Punto di Accesso che immediatamente lo inoltra al Gestore Centrale; a sua volta il Gestore Centrale provvede al deposito dell'atto presso l'Ufficio giudiziario indicato, il quale verifica i contenuti pervenuti e aggiorna il fascicolo informatico e il registro del Sistema Informatico Civile.
In questo momento il deposito risulta perfezionato e risulta visibile tramite i servizi di consultazione (Figura 3).

8. Tempi di realizzazione

Secondo le intenzioni del legislatore, il D.P.R. 123/01 avrebbe dovuto applicarsi ai giudizi iscritti al ruolo dopo il 1° gennaio 2002, mentre le regole tecnico-operative erano attese per il 30 ottobre 2001; in realtà queste ultime sono state approvate con ben 3 anni di ritardo.
Nel frattempo è stata opportunamente avviata una fase di sperimentazione presso sette sedi pilota individuate in tutta Italia (Bari, Bergamo, Bologna, Catania, Genova, Lamezia Terme e Padova), nelle quali vengono sviluppate le infrastrutture informatiche e testate le funzionalità dei sistemi del processo telematico.
Presso ciascuna sede pilota opera un laboratorio di sperimentazione composto da avvocati, magistrati, cancellieri ed esperti informatici.
Sono stati fino a oggi realizzati, installati e collaudati tutti i sistemi informatici previsti dalla prima fase del progetto e, in particolare, il Gestore Locale, il Gestore Centrale, il Punto di Accesso del Ministero, la postazione di lavoro dell'avvocato, l'applicativo per le cancellerie.
Presso il Centro di gestione del Ministero della Giustizia di Napoli sono installati:
- il Gestore Centrale
- il Punto di Accesso del Ministero

Presso le sale server delle sette sedi pilota sono installati:
- il Gestore Locale
- i sistemi informatizzati di gestione dei registri di cancelleria, il repository documentale (la base dati contenente anche il fascicolo informatico).

Si stanno, inoltre, mettendo a punto i sistemi

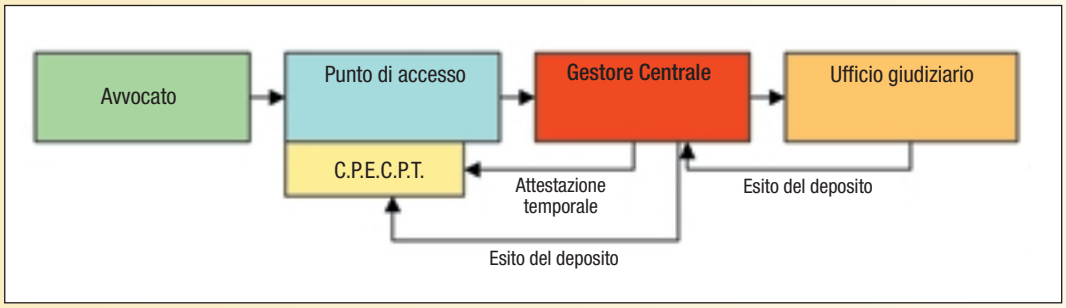


FIGURA 3
Flusso delle informazioni

[5] L'eXtensible Markup Language (XML) è un meta-linguaggio di marcatura e si compone di un insieme standard di regole sintattiche per modellare la struttura di documenti e dati. Tali regole, dette più propriamente specifiche, sono state definite dal W3C (Worl Wide Web Consortium) e sono consultabili all'indirizzo http://www.w3.org/XML.

[6] L'algoritmo utilizzato per l'operazione di cifratura simmetrica del file si chiama 3DES e le chiavi simmetriche di sessione vengono cifrate utilizzando la chiave pubblica contenuta nel certificato dell'ufficio giudiziario destinatario.

che prevedono l'estensione del processo telematico alle Sezioni Lavoro, nonché la consolle del giudice (agenda, gestione dell'udienza, produzione provvedimenti ecc.).
In realtà l'attuazione del Processo Telematico implica non solo la risoluzione delle problematiche tecniche derivanti dall'applicazione al mondo della giustizia civile delle tecnologie informatiche e telematiche, ma anche l'esigenza di riconfigurare radicalmente le modalità attraverso le quali i processi vengono trattati e di agire sui modelli organizzativi.

ANTONIO PIVA laureato in Scienze dell'Informazione, Membro del Consiglio Nazionale e Presidente della commissione di informatica giuridica dell'ALSI (*Associazione Nazionale Laureati in Scienze dell'Informazione ed Informatica*).
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 e ispettore AICA ECDL base ed advanced.
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

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.

I linguaggi per la programmazione

Carlo Ghezzi

1. Introduzione

Il linguaggio costituisce il mezzo primario per la comunicazione tra le persone. In assenza di un linguaggio col quale comunicare in forma orale e scritta non esisterebbe una società organizzata: non sarebbe possibile l'interazione e la cooperazione tra gli individui né sarebbe possibile l'accumulo e la trasmissione di conoscenze. Lo studio del linguaggio è pertanto sempre stato strettamente connesso con gli studi sociali, antropologici e filosofici.
A partire dalla seconda metà del secolo scorso, la nascita dell'informatica ha fatto nascere una nuova dimensione di sviluppo dei linguaggi: i linguaggi per la comunicazione tra l'uomo e il computer. In realtà la comunicazione tra uomo e macchina è un fenomeno più generale, nato prima e indipendentemente dall'informatica; ma è solo attraverso le "macchine informatiche" che la comunicazione raggiunge livelli di sofisticazione che richiedono l'uso di linguaggi di complessità paragonabile a quella dei linguaggi per la comunicazione umana. In questo articolo ci concentreremo sui linguaggi per la programmazione, lasciando ad altri articoli un approfondimento di altre forme linguistiche tipiche dell'informatica (ad esempio, i linguaggi per il trattamento di dati, i linguaggi per la specifica dei requisiti ecc.).

2. Perché i linguaggi di programmazione?

Il computer è una macchina in grado di eseguire programmi scritti nel proprio linguaggio macchina: è un interprete del linguaggio macchina. I programmi consentono di specializzare il computer all'esecuzione di compiti specifici; una volta rappresentati nella forma binaria richiesta dalla tecnologia digitale del computer e inseriti nella memoria centrale, questi vengono interpretati dall'unità centrale di elaborazione (CPU).
Fin dalla nascita dei primi computer, ci si rese conto che programmare direttamente in linguaggio macchina non sarebbe stato agevole, se non per piccoli e semplici programmi. La rappresentazione del programma in linguaggio macchina avrebbe comportato la scomposizione del problema da risolvere in un algoritmo espresso in istruzioni da formulare in un lin-

guaggio binario: istruzioni elementari per il trasferimento di dati dalla e nella memoria, per l'esecuzione di semplici operazioni aritmetico-logiche, per il trasferimento esplicito del controllo mediante la modifica del "program counter". In sintesi, il livello di astrazione al quale il programmatore doveva esprimersi era troppo lontano dal problema che voleva risolvere.

Una prima soluzione fu quella di rendere simbolico il linguaggio macchina, definendo così i linguaggi di tipo assembler. In tal modo, anziché usare un ostico alfabeto binario, il programmatore poteva dare nomi simbolici mnemonici alle operazioni elementari del computer e alle celle di memoria contenenti dati o istruzioni. Di fatto, si manteneva una corrispondenza biunivoca tra istruzioni in linguaggio macchina e istruzioni assembler: semplicemente, si rendevano queste ultime più facilmente usabili, in quanto più facilmente comprensibili alle persone umane.

Ma la storia dei linguaggi di programmazione era solo agli inizi. A partire dagli anni 50 del secolo scorso iniziò infatti un flusso continuo di proposte e realizzazioni di nuovi linguaggi di programmazione, definiti allo scopo specifico di facilitare la programmazione dei computer. Si tratta di linguaggi che si possono complessivamente definire di *alto livello*. Con ciò si intende che il livello espressivo fornito dal linguaggio si allontana dal livello di dettagli minuti imposti dalle operazioni svolte dai circuiti digitali che costituiscono l'elaboratore per avvicinarsi alle capacità espressive umane. Senza questi linguaggi possiamo senz'altro affermare che l'informatica non si sarebbe diffusa fino a diventare la tecnologia fondante della vita contemporanea: non si sarebbero sviluppate le applicazioni che rendono i computer utili o addirittura indispensabili per il funzionamento quotidiano della nostra società.

La motivazione principale alla base della continua evoluzione dei linguaggi è proprio la necessità di rispondere meglio ai requisiti posti dalle applicazioni e dal loro processo di ingegnerizzazione. Le applicazioni hanno invaso settori applicativi che vanno dalla gestione aziendale al calcolo scientifico, dai giochi interattivi all'elaborazione di documenti, dal controllo di impianti in tempo reale alle gestione distribuita di servizi in rete. Le esigenze poste al linguaggio nei diversi casi sono spesso assai diverse. Allo stesso tempo sono diventati sempre più critici i requisiti di affidabilità delle applicazioni, di una loro facile adattabilità al mutare dei requisiti con il passare del tempo, di un loro sviluppo rapido da parte di team di programmatori che lavorano in parallelo. Anche i computer sono andati evolvendo nel tempo in termini di prestazioni e di memoria. Tutto ciò ha portato a una continua evoluzione dei linguaggi e fa presumere che questa evoluzione sia destinata a continuare nel prossimo futuro.

La tabella 1 sintetizza gli sviluppi storici dei linguaggi di programmazione, limitando la visione della Babele linguistica esistente a una piccola porzione che vuole mettere in luce i linguaggi che riteniamo più influenti. La tabella indica il periodo storico in cui i linguaggi sono stati definiti, evitando di mostrare le versioni successive che ne hanno accompagnato l'evoluzione. Nel seguito di questo articolo ci concentreremo invece non tanto sullo sviluppo storico dei linguaggio—compito impossibile per una trattazione sintetica come questa—quanto su alcuni concetti fondamentali che consentono una loro classificazione.

3. Classi di linguaggi e paradigmi di programmazione

I linguaggi di programmazione possono essere caratterizzati in base allo stile di programmazione che essi supportano, detto anche *paradigma di programmazione*. Un paradigma caratterizza il modo in cui il programma si presenta, come vengono espresse le computazioni e come il programma viene organizzato in parti.

1950-1960	1960-1970	1970-1980	1980-1990	1990-oggi
FORTRAN	BASIC	Smalltalk	Ada	Java
ALGOL 60	SIMULA 67	Modula-2	C++	C#
COBOL	ALGOL 68	PROLOG	Eiffel	PERL
APL	Pascal	Scheme	Common LISP	Python
LISP			CLOS	

TABELLA 1
Tappe fondamentali dell'evoluzione storica dei linguaggi

L'esistenza di diversi stili linguistici caratterizza anche altre forme espressive: dal linguaggio poetico a quello architettonico o musicale. In architettura si parla infatti di stile romanico o gotico, di "art nouveau" o di stile "bauhaus". Ciascuno stile è nato in risposta ad esigenze costruttive (e, ovviamente, estetiche). L'analogia vale anche per i linguaggi di programmazione, anche se la motivazione "estetica", pur presente, non è certo prevalente. Fondamentale è invece la capacità dei linguaggi di favorire lo sviluppo di software di buona qualità.

I paradigmi dei linguaggi di programmazione possono essere distinti in due classi fondamentali: paradigmi computazionali e strutturali. Una loro comprensione e la classificazione dei diversi linguaggi secondo i diversi paradigmi possono aiutare a meglio capire sia i linguaggi che le applicazioni che con essi vengono sviluppate.

Un *paradigma computazionale* caratterizza il modello di calcolo definito dal linguaggio. Il paradigma computazionale di gran lunga prevalente è quello imperativo, il cui modello di calcolo è il cosiddetto "modello di Von Neumann". Con ciò si intende che i programmi scritti utilizzando il linguaggio riflettono il modo di operare del computer soggiacente, il quale a sua volta riflette il modo di operare del computer primigenio, ideato da John Von Neumann. In altre parole, i linguaggi imperativi non sono altro che astrazioni della macchina di Von Neumann, il cui modello di calcolo consiste nell'esecuzione sequenziale, passo-passo, di istruzioni che modificano lo stato della memoria della macchina. Sono astrazioni, in quanto mascherano i dettagli di basso livello mentre mantengono i tratti fondamentali del funzionamento da cui astraggono. I linguaggi imperativi operano su variabili, intese come astrazioni di celle di memoria il cui contenuto viene modificato nel corso dell'esecuzione, e forniscono istruzioni per l'esecuzione condizionale (tipo *if...then...else*) e per le iterazioni (tipo *while...do...*). A questa classe appartengono linguaggi ormai obsoleti, ma tuttora in uso, come FORTRAN e COBOL, ma anche linguaggi più moderni come C, C++, Java e C#.

Val la pena osservare che con il passare del tempo, in realtà, la "macchina" in grado di eseguire i programmi è andata evolvendo rispetto al primitivo computer ideato da Von Neumann. In particolare, si è passati dal modello di calcolo della singola macchina di Von Neumann a quello di un insieme di macchine di Von Neumann operanti in parallelo. Ciò è stato reso possibile, da un lato, dalla definizione di meccanismi per poter "multi-programmare" i computer, fornendo l'astrazione del parallelismo logico dei programmi, e, dall'altro, dalla possibilità di interconnettere e coordinare più computer, realizzando sistemi fisicamente paralleli (sistemi multi-processore o sistemi distribuiti, interconnessi da una rete di comunicazione). Linguaggi moderni, come Java o C#, forniscono un supporto linguistico per la programmazione concorrente e distribuita, permettendo di definire sia più unità concorrenti che operano all'interno di un singolo nodo di elaborazione, sia unità che possono essere allocate su nodi diversi di una rete.

Anche se la concorrenza e la distribuzione delle computazioni ha reso più complesso il modello computazionale complessivo di linguaggi tipo Java o C#, i singoli nodi dell'architettura soggiacente il linguaggio continuano a seguire il paradigma di Von Neumann. Paradigmi computazionali diversi sono invece offerti dai linguaggi funzionali e dai linguaggi logici (o dichiarativi). In entrambi i casi si raggiunge un più elevato livello di astrazione rispetto al paradigma imperativo, che maschera i dettagli esecutivi del computer soggiacente, a scapito però di una minore efficienza di esecuzione di programmi.

Un *paradigma funzionale* esprime il calcolo effettuato da un programma come una funzione matematica. Le sue basi concettuali risiedono dunque nella matematica, e in particolar modo nella teoria delle funzioni ricorsive. I linguaggi funzionali consentono di esprimere in maniera sintetica ed elegante il procedimento di risoluzione di problemi anche complessi. Il capostipite dei linguaggi funzionali è il LISP, usato in passato prevalentemente per lo sviluppo di applicazioni nell'ambito dell'intelligenza artificiale. Altri linguaggi funzionali sono ML, Scheme[1], e Haskell.

Il modello computazionale dei *linguaggi logici* (*o dichiarativi*) invece si basa sulla logica (più precisamente, su sottoinsiemi del calcolo dei predicati del I ordine) per la descrizione dei programmi e

[1] Molte università hanno adottato Scheme come primo linguaggio di programmazione dei curricula di Informatica per la valenza formativa delle astrazioni fornite dai linguaggi funzionali (in particolar modo, l'uso della ricorsione). Esistono inoltre ottimi testi didattici basati sull'uso di Scheme.

sulle regole logiche di inferenza per la loro valutazione. Anche in questo caso, si tratta dunque di un paradigma fortemente radicato nella matematica. Il più noto linguaggio logico è il Prolog.
Passiamo ora ad analizzare i *paradigmi strutturali*. Questi classificano i linguaggi di programmazione in funzione delle modalità che essi offrono per dare una struttura ai programmi, scomponendoli in unità logiche. I due principali paradigmi strutturali sono quello procedurale e quello orientato agli oggetti (object-oriented). Entrambi i paradigmi prevedono una scomposizione di un programma complesso in parti, possibilmente caratterizzate da elevati livelli di autonomia e sviluppabili in maniera indipendente le une dalle altre. Entrambi i paradigmi, pertanto, rispondono al classico principio "divide et impera"; entrambi costituiscono degli strumenti per la modularizzazione di un programma. La scomposizione in moduli fa sì che il programma complessivo risulti più facile da leggere e da scrivere: risulta dunque più facile sia lo sviluppo iniziale del programma che la sua successiva manutenzione. La possibilità di sviluppo separato e indipendente delle diverse parti consente poi di ridurre i tempi complessivi di sviluppo delle applicazioni, rispetto ai tempi necessari per lo sviluppo di un'applicazione monolitica. I due paradigmi si differenziano per le astrazioni rese possibili dai meccanismi di modularizzazione. Come illustreremo tra breve, il paradigma procedurale consente una scomposizione per funzioni, mentre il paradigma object-oriented consente una scomposizione per oggetti.
Secondo il *paradigma procedurale*, un programma viene scomposto in sottoprogrammi. Tipici esempi sono le SUBROUTINE e FUNCTION del Fortran, le procedure del Pascal, le function del C. Un sottoprogramma consente di definire e dare nome a un'operazione complessa: per esempio, una funzione *sort* per ordinare un insieme di valori, una funzione *solve* per risolvere un sistema di equazioni, o una funzione *draw_map* per disegnare una mappa. Una volta definita, l'operazione complessa può essere utilizzata tutte le volte che ciò risulti necessario attraverso una semplice operazione di chiamata del sottoprogramma. Dal punto di vista della modularizzazione che così viene ottenuta, si può osservare dunque che risultano ben isolate le parti di programma che utilizzano le operazioni complesse da quelle che le definiscono. Eventuali cambiamenti interni al sottoprogramma, per esempio la codifica di un diverso algoritmo di ordinamento, all'interno del sottoprogramma *sort*, non hanno riflessi imprevisti sulle altre parti del programma e possono dunque essere da queste ignorate.
Il *paradigma object-oriented* scompone invece un programma in classi di oggetti. Una classe di oggetti è definita da un insieme di operazioni attraverso le quali gli oggetti della classe possono essere manipolati. Come semplice esempio, si supponga di voler rappresentare in un programma la classe di oggetti "poligoni regolari". Le operazioni che si vogliono definire per i poligoni regolari potrebbero essere il calcolo del perimetro, dell'area e il disegno sullo schermo del computer. Deve essere poi possibile creare diversi poligoni regolari, definendone la dimensione del lato. Un altro esempio è la classe fax, che definisce oggetti (i diversi dispositivi fisici fax), mediante i quali è possibile trasmettere e ricevere documenti. Infine, un altro esempio è costituito dai conti correnti bancari, per i quali le operazioni possibili sono depositi, prelievi e richieste di saldo, e dai correntisti di una banca, le cui operazioni possibili sono l'apertura e chiusura di un conto corrente o la richiesta di un mutuo.
Dagli esempi forniti, si può osservare che un oggetto è caratterizzato da uno stato (i dati nei quali vengono memorizzati valori che possono essere modificati) e da operazioni. In un linguaggio object-oriented pertanto gli oggetti sono definiti da un costrutto speciale, detto *classe*, di cui possono essere generati esemplari. Una classe raggruppa la definizione della struttura di dati e dei sottoprogrammi che implementano le operazioni. Inoltre una classe si comporta come un tipo dei convenzionali linguaggi procedurali. Così come è possibile generare, per esempio in FORTRAN o C, diverse variabili di tipo *integer*, allo stesso modo si possono generare diversi conti correnti e correntisti, esemplari delle rispettive classi. Inoltre, così come è possibile manipolare gli *integer* mediante apposite operazioni definite per essi dal linguaggio, allo stesso modo si possono manipolare conti correnti e correntisti con le operazioni per essi definite dalle rispettive classi. Si dice pertanto comunemente che il costrutto di classe consente di definire *tipi di dati astratti*.
I vantaggi in termini di strutturazione e modularità dei linguaggi object-oriented rispetto ai linguaggi procedurali sono notevoli. La possibilità

non solo di associare una struttura di dati alle operazioni che ne consentono la manipolazione, ma anche di rendere invisibile tale struttura di dati all'esterno del costrutto di classe (e cioè ai moduli di programma clienti) consente di ottenere sia una più razionale suddivisione in moduli che una più facile modificabilità dei programmi. Eventuali modifiche della struttura di dati che realizza lo stato degli oggetti non hanno effetto sui moduli clienti. La modifica della struttura di dati resta nascosta (incapsulata) all'interno del costrutto di classe.

Un'ulteriore caratteristica fondamentale dei linguaggi object-oriented è il costrutto di *sottoclasse*. Mediante questo costrutto è possibile definire oggetti che specializzano il comportamento della classe originaria. Per esempio, avendo definito la classe "fax", è possibile definire la sottoclasse "fax con telefono", che permette di comunicare non solo documenti, ma anche messaggi vocali. La sottoclasse definisce solo le nuove funzionalità che gli oggetti della sottoclasse forniscono, in quanto tutte le altre risultano *ereditate* dalla classe. La sottoclasse può inoltre ridefinire alcune delle funzionalità fornite dalla classe.

Il meccanismo di sottoclasse è un potente strumento linguistico a supporto della evoluzione del software. Per esempio, la modifica che aggiunge a un fax la funzionalità di trasmissione dei messaggi vocali viene ottenuta senza dover apportare modifiche globali al software esistente: basta definire le nuove funzionalità in una sottoclasse. Inoltre i programmi esistenti continuano ad operare correttamente anche se gli oggetti "fax" su cui operano non sono dei semplici fax, ma dei "fax con telefono", in quanto le operazioni di trasmissione e ricezione di testi continuano ad essere da questi supportate[2].

Il paradigma object-oriented è stato introdotto per la prima volta dal linguaggio SIMULA 67; esso è stato riscoperto più tardi da Smalltalk e infine diffuso, in una forma ibrida, dal C++. C++ è riuscito a diffondere i concetti della progettazione del software object-oriented in quanto essi sono stati aggiunti ai concetti della tradizionale programmazione procedurale offerta dal linguaggio C, consentendo in tal modo un'adozione graduale del nuovo paradigma. I linguaggi Java e C#, che stanno avendo una larga diffusione, adottano in maniera più coerente il paradigma object-oriented.

4. Ulteriori classificazioni e conclusioni

I paradigmi computazionali e strutturali consentono di classificare i linguaggi di programmazione, e quindi di meglio coglierne le caratteristiche distintive principali. Altre forme di classificazione sono pure possibili, e spesso molto utili. Si potrebbe distinguere i linguaggi a seconda dello stile mediante il quale si definiscono i programmi. In tal caso, si potrebbe distinguere tra linguaggi testuali e linguaggi grafici, o visuali. Altre distinzioni potrebbero essere fatte tra i linguaggi compilati e linguaggi interpretati, o tra i linguaggi con tipizzazione statica e quelli con tipizzazione dinamica. Nel primo caso, si vuole distinguere tra i linguaggi per i quali sono prevalenti le implementazioni basate sulla traduzione verso un linguaggio macchina (o un linguaggio vicino ad essa) e quelli che vengono implementati mediante una diretta interpretazione dei programmi nella forma sorgente[3]. Tipici linguaggi interpretati sono LISP e molti cosiddetti "linguaggi di script" che oggi sono molto diffusi (per esempio, Python o più recentemente Ruby). Normalmente invece i linguaggi sono compilati, anche se sono comuni situazioni ibride, come quella adottata da Java, che non viene di solito compilato in linguaggio macchina, ma in un linguaggio intermedio, detto Bytecode, che viene poi interpretato. La seconda classificazione invece distingue tra linguaggi in cui ogni variabile ha associato in maniera statica un tipo, che caratterizza gli oggetti a cui può fare riferimento, rispetto a quelli in cui questo tipo può cambiare dinamicamente. Nel primo caso, il linguaggio consente di verificare, prima dell'esecuzione, che i dati vengano manipolati in maniera corretta; nel secondo, questo

[2] Tecnicamente, questo importante risultato è dovuto a due caratteristiche dei linguaggi object-oriented: il binding dinamico e il polimorfismo.

[3] Si potrebbe obiettare che stiamo in realtà distinguendo due modi per implementare i linguaggi piuttosto che diverse tipologie di linguaggi. In realtà, approfondendo questa questione si arriverebbe a osservare che proprio alcune caratterstiche dei linguaggi favoriscono l'una o l'altra soluzione implementativa.

non risulta possibile (se non imponendo vincoli specifici). Intuitivamente, i linguaggi che ricadono all'interno della prima categoria supportano una maggiore affidabilità dei programmi, in quanto possono garantire a priori l'assenza di certi errori durante l'esecuzione.
Per concludere questa breve trattazione sui linguaggi di programmazione, sottolineiamo come sia fondamentale che uno specialista di informatica padroneggi i concetti che ne stanno alla base. I linguaggi di programmazione sono gli strumenti principali per la produzione di software, e questo sta alla base delle applicazioni che si usano quotidianamente in ogni settore della vita odierna. La qualità delle applicazioni e, in primis, la loro affidabilità, è fortemente dipendente dal linguaggio di programmazione che è stato utilizzato per lo sviluppo. Capire a fondo i linguaggi significa dunque non solo poterli usare al meglio nello sviluppo delle applicazioni ma, in ultima analisi, sviluppare prodotti di qualità.

Bibliografia

Elencare i riferimenti ai diversi linguaggi di programmazione porterebbe a una bibliografia sterminata. Si riporta dunque soltanto il riferimento a un testo di carattere generale che contiene un'analisi dei concetti dei linguaggi, oltre ai riferimenti bibliografici ai diversi linguaggi analizzati.

[1] Ghezzi C., Jazayeri M.: *Programming Language Concepts*. (III Edition), J. Wiley and Sons, 1997 (la traduzione della II edizione di questo testo è disponibile in lingua Italiana col titolo Concetti dei Linguaggi di Programmazione, Editore Franco Angeli, 1989).

Carlo Ghezzi è Professore Ordinario di Ingegneria del Software, membro del Senato Accademico del Politecnico di Milano e responsabile scientifico dell'area di ricerca sull'Ingegneria del Software presso il CEFRIEL. È Fellow dell'ACM (*Association for Computing Machinery*) e dell'IEEE e Editor in Chief della rivista ACM *Transactions on Software Engineering and Methodology*. È autore di numerosi articoli scientifici e libri. Ha interessi di ricerca nell'ambito dei linguaggi di programmazione e dell'ingegneria del software, con particolare riguardo ai fondamenti teorici, metodologici e tecnologici dello sviluppo delle applicazioni distribuite su rete.
E-mail: carlo.ghezzi@polimi.it

Tecnologie alla ribalta
RETI DI MICRODISPOSITIVI INTELLIGENTI

Mirko Cesarini
Carlo Ghezzi
Fabio Schreiber
Letizia Tanca

Le *Wireless Sensor Network* ed i *tag RFID* sono tecnologie recenti con grandi potenzialità applicative che causeranno un aumento del grado di penetrazione dei dispositivi elettronici e dei sistemi informatici nella vita di ogni giorno. Al di la dei timori suscitati da alcuni recenti interventi sui mass media, questi nuovi strumenti hanno tutte le carte in regola per incidere profondamente non solo nel mondo dell'ICT, ma anche in molti dei tradizionali settori industriali. Questo articolo presenta una panoramica di queste tecnologie.

1. INTRODUZIONE

Negli ultimi anni il progresso compiuto nel campo della miniaturizzazione degli apparati di elaborazione, memorizzazione e comunicazione ha permesso la creazione di nuove famiglie di dispositivi che per le loro caratteristiche possono essere distribuiti in grandi quantità nell'ambiente per svolgere funzioni di rilevazione dati, di controllo della movimentazione delle merci e in generale per realizzare nuove e avanzate forme di interazione con il mondo circostante. Esempi d'uso si trovano nella logistica, nei sistemi di telecontrollo ed in generale nelle applicazioni di "ambient intelligence".
Tra gli esempi più noti e interessanti di questi sistemi periferici intelligenti e pervasivi troviamo le reti wireless di sensori (*Wireless Sensor Network*, WSN) e le etichette (*tag*) RFID (*Radio Frequency IDentification*). Su di essi si concentra lo studio riportato in questo articolo. L'impiego delle WSN e dei *tag* RFID permette di far uscire le informazioni digitali dai confini dei server aziendali per distribuirle sugli oggetti e negli ambienti fisici. Grazie a ciò si aprono nuove possibilità per quanto riguarda la progettazione di software e di sistemi informativi aziendali. Nel corso dell'articolo forniremo una panoramica sugli aspetti più salienti delle due tecnologie e ci focalizzeremo su come questi possono influenzare il processo di sviluppo del software ed i sistemi per la gestione dei dati.
Nella loro forma più nota, le WSN sono reti formate da tanti piccoli elementi chiamati *mote*, dal nome utilizzato per i primi prototipi di WSN sviluppati dall'Università della California Berkeley (USA). Un *mote* è un dispositivo di elaborazione in miniatura dotato di memoria, di un apparato ricetrasmittente, di sensori per la rilevazione di dati ambientali e di una batteria per l'alimentazione. I *mote* (Figura 1) vengono progettati in modo che, una volta accesi e inseriti nel luogo di funzionamento, siano in grado di operare a lungo senza intervento umano.
I *mote* sfruttano le trasmissioni radio per comunicare tra loro e per trasmettere i dati raccolti. L'assenza di cavi di alimentazione e di trasmissione dati facilita il dispiegamento del-

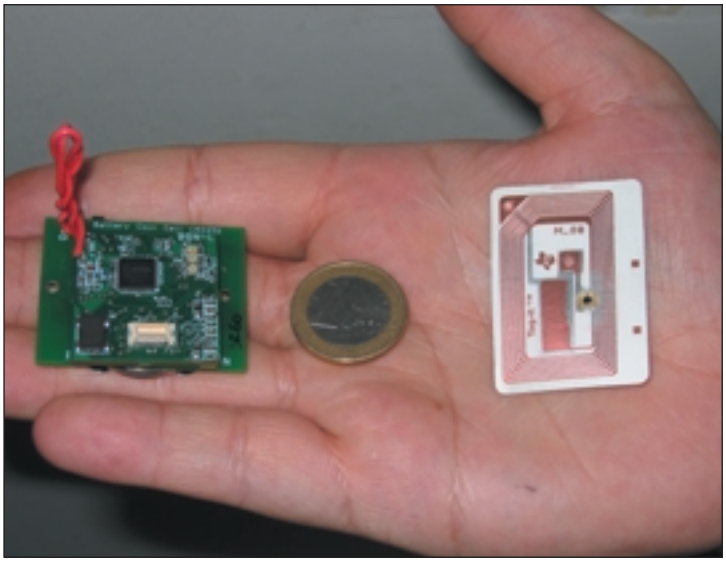


FIGURA 1

Da sinistra verso destra sono visibili rispettivamente un mote, una moneta da 1 euro ed un RFID

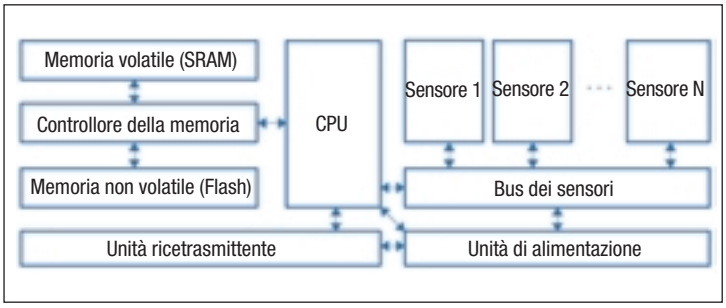


FIGURA 2

Schema hardware di un mote

le WSN, consentendone l'utilizzo in tutte quelle situazioni dove i tradizionali sensori cablati non possono essere impiegati efficacemente. Un *tag* RFID è una memoria, accoppiata ad una antenna di trasmissione, che può essere interrogata a distanza da un apposito lettore. Un *tag* per funzionare sfrutta l'energia irradiata dai dispositivi con i quali interagisce e per questo motivo normalmente non ha bisogno di una batteria[1]; ciò rende possibile produrre dei *tag* con le dimensioni ed il volume di un francobollo (Figura 1) e con costi molto bassi. Una volta posizionati su oggetti fisici, gli RFID memorizzano informazioni in formato digitale che possono essere lette a distanza. Gli RFID potenzialmente possono rivoluzionare molti settori applicativi: si tratta di una tecnologia che offre funzionalità di identificazione e di memorizzazione equivalenti a quelle fornite dai codici a barre e che supera alcune delle limitazioni che affliggono le tecnologie ottiche (i primi sistemi RFID sono stati sviluppati proprio con l'obiettivo di sostituire i codici a barre).

Il resto di questo articolo è organizzato nel modo seguente: nel paragrafo 2 analizzeremo le reti wireless di sensori, nel paragrafo 3 analizzeremo in dettaglio la tecnologia RFID. In entrambi i paragrafi ci focalizzeremo sulle problematiche connesse alla gestione dei dati negli scenari d'impiego delle nuove tecnologie. Infine nel paragrafo 4 verranno tratte alcune conclusioni e descritti i possibili sviluppi futuri.

2. RETI WIRELESS DI SENSORI

2.1. Architettura hardware

Un *mote* di una rete di sensori è descritto dallo schema riportato nella figura 2. I componenti principali sono: l'unità di elaborazione, le memorie, l'unità ricetrasmittente e l'insieme di sensori di cui il *mote* è corredato.

Nel seguito di questo paragrafo analizzeremo con qualche dettaglio i singoli componenti dell'architettura hardware delle WSN. Successivamente al punto 2.2 descriveremo l'architettura software e infine al punto 2.3 illustreremo un caso applicativo.

2.1.1. Unità di elaborazione

Rispetto ai sensori tradizionali, i componenti di una WSN hanno la capacità di elaborare le informazioni: un *mote* può rimuovere il rumore di fondo dai segnali raccolti, trasformare i dati raccolti in informazioni di più alto livello e più sintetiche (esempio, il calcolo della media delle temperature rilevate in un periodo) o aggregarle con informazioni ricevute dagli altri *mote* prima di trasmetterle al destinatario. Queste operazioni sono svolte dall'unità di elaborazione, la quale oltre a elaborare i dati raccolti dai sensori si occupa anche di gestire le risorse di un *mote*. I microprocessori impiegati devono essere ottimizzati per consumare poca energia, che costituisce una risorsa scarsa nelle normali condizioni di utilizzo, infatti i *mote* difficilmente possono essere recuperati una volta dispiegati nell'ambiente di lavoro; per

[1] Per essere precisi, all'interno della famiglia dei tag RFID, solo i tag passivi non usano batterie. Per una classificazione dettagliata si rimanda al punto 3.1.

questo motivo prolungare la durata della batteria è importante. Le unità di elaborazione attualmente disponibili sono a 8, 16 e 32 bit. Come criterio per il confronto si possono considerare sia le prestazioni di punta del processore sia l'indice MIPS/Watt che fornisce un'indicazione di quale sia il costo della potenza di calcolo erogata in termini di energia consumata. In linea generale si può affermare che i microprocessori a 32 bit offrono elevate prestazioni di calcolo ma consumano più energia; al contrario, i processori a 8 bit sono parchi nel consumo di energia, ma a scapito delle prestazioni.

2.1.2. Memorie

I *mote* usano memorie SRAM come memoria di lavoro e memorie flash come memoria di massa. Le memorie SRAM generalmente sono integrate nel processore; le memorie flash hanno capacità di poche centinaia di kbyte, arrivano ad alcuni Mbyte solo nei *mote* più dotati. Le memorie flash conservano il loro contenuto anche quando non vengono alimentate e sono utilizzate per ospitare sia i programmi sviluppati dagli utenti per i *mote* sia i dati rilevati dai sensori in attesa di essere elaborati o trasmessi.

2.1.3. Unità ricetrasmittente

Sono disponibili diverse soluzioni per realizzare il sistema di comunicazione di un *mote*. Alcuni *mote*, per esempio i *mote* Crossbow [40] nella serie Mica2, utilizzano dei protocolli di comunicazione proprietari, altri utilizzano protocolli standard come Bluetooth [37], IEEE 802.15.4/ZigBee [63] e IEEE 802.11 (WiFi) [66]. IEEE 802.11 (WiFi) è un protocollo per wireless LAN che risulta carente sotto il profilo del risparmio energetico e per questo motivo è scarsamente utilizzato nelle WSN. Bluetooth e IEEE 802.15.4/ZigBee sono protocolli per WPAN (*Wireless Personal Area Network*). Le WPAN sono nate per sostituire, con collegamenti radio, i cavi di connessione dei vari dispositivi "indossati" da una persona e implementano strategie di risparmio energetico. Nel riquadro è riportato un confronto tra Bluetooth e ZigBee.
Ogni *mote* poi mette a disposizione dei programmatori servizi per crittografare i dati trasmessi.

Confronto tra Bluetooth e ZigBee

Concepita inizialmente per sostituire i cavi di collegamento tra PC, telefoni e altri dispositivi elettronici, la tecnologia Bluetooth si è affermata anche per realizzare piccole reti locali in grado di riconfigurarsi automaticamente man mano che si presentano nuovi dispositivi nell'area di copertura. La tecnologia Bluetooth, regolata dalle norme IEEE 802.15.1, utilizza la frequenza radio ISM (*Industrial Scientific Medical*) a 2,45 GHz, di libero uso in tutto il mondo. I dispositivi hanno un raggio utile di comunicazione di circa 10 m in linea d'aria (100 m con Bluetooth 2) che si riduce in presenza di ostacoli. Bluetooth è stato concepito per connettere tra loro un numero limitato di dispositivi (se sono presenti più di 8 dispositivi, la rete si segmenta in gruppi di massimo 8 elementi) garantendo una banda di 1 Mbyte/s, sufficiente per trasmettere voce e video. I dispositivi sono ottimizzati per limitare il consumo energetico delle batterie: le potenze impiegate per trasmettere sono basse e i dispositivi di comunicazione rimangono per la maggior parte del tempo in modalità di funzionamento a basso consumo energetico.
IEEE 802.15.4/ZigBee è una specifica per reti di dispositivi a basso consumo energetico che offre una banda di 250 kbit/s, dà la possibilità di gestire connessioni che coinvolgono fino a 255 altri dispositivi, utilizza la frequenza radio ISM (*Industrial Scientific Medical*) a 2,45 GHz. La massima distanza di comunicazione da un dispositivo all'altro, come per Bluetooth è di circa 10 m in linea d'aria, un po' meno con ostacoli in mezzo. A differenza di Bluetooth, ZigBee è ottimizzato per connettere un elevato numero di dispositivi che comunicano sporadicamente tra di loro e che per le loro comunicazioni non necessitano di banda elevata.

2.1.4. Sensori

I *mote* hanno sia sensori built-in sia un bus di espansione per mezzo del quale è possibile aggiungere altri sensori necessari per svolgere funzioni specifiche. Sono disponibili molti tipi di sensori, per esempio rilevatori di temperatura, rumore, ricevitori GPS ecc.. È tuttavia possibile costruire un sensore ad hoc specifico per le proprie esigenze.
Un sensore è composto generalmente da un trasduttore e da un convertitore di segnali analogici in digitali. I trasduttori sono costruiti sfruttando le caratteristiche di certi materiali che variano le loro caratteristiche elettriche al variare delle condizioni ambientali. Un ADC (*Analog to Digital Converter*, convertitore di segnali analogici in segnali digitali) converte il valore di tensione su un trasduttore in un valore binario che verrà poi utilizzato per le successive elaborazioni.
Molti dei trasduttori utilizzati sui *mote* sono dei MEMS (*MicroElectroMechanical Systems*): si tratta di dispositivi in grado di rilevare una vasta gamma di fenomeni fisici in maniera efficiente ed economica dal punto di vista del consumo energetico; confrontati con i sensori piezoelettrici ad alta precisione i MEMS forniscono una discreta precisione pur avendo un costo di produzione notevolmente inferiore.

Un MEMS viene costruito utilizzando processi di incisione (*etching*) del silicio (in maniera analoga al procedimento utilizzato per stampare circuiti integrati) per costruire delle piccolissime strutture meccaniche con dimensioni dell'ordine 5 – 10 μm. Su queste microstrutture, man mano che vengono costruite, vengono anche stampati dei circuiti a semiconduttore. La forza di gravità o le accelerazioni possono flettere le strutture di silicio, causando delle variazioni alle caratteristiche elettriche dei circuiti semiconduttori. Il primo esempio di impiego su larga scala di un sensore MEMS è l'accelerometro che causa l'apertura degli airbag nelle automobili [9].

2.1.5. Considerazioni riassuntive

Nella tabella 1 riportiamo un confronto delle principali caratteristiche hardware di alcuni sensori disponibili in commercio. Nella tabella 2 abbiamo inserito un elenco di progetti di hardware per WSN con relativi riferimenti per l'approfondimento. Questi progetti descrivono prototipi sviluppati da università e centri di ricerca, alcuni dei quali sono diventati dei prodotti commerciali; per alcuni sono disponibili gli schemi circuitali di realizzazione.

2.2. Architettura software

Un *mote* è un elaboratore dotato di risorse hardware molto limitate, che deve gestire un apparato ricetrasmittente ed il routing dei pacchetti dati. Il sistema operativo e le applicazioni sviluppate devono essere in grado di esercitare un controllo fine delle risorse hardware per poter implementare politiche di risparmio energetico efficaci che consentano di far durare il più possibile la batteria che alimenta il *mote*. I sistemi operativi tradizionali (come Linux, OSE, QNX, . . .) impiega-

Nome piattaforma	IMote	Mica2	MicaZ	Telos B	Stargate
CPU (produttore) Modello Velocità Bit	ARM ARM7TDMI 12 MHz 32 bit	Atmel ATMEGA128 8 MHz 8 bit	Atmel ATMEGA128 8 MHz 8 bit	Texas Instruments MSP430 8 MHz 16 bit	Intel PXA255 400 MHz 32 bit
SRAM (KB)	64	4	4	10	64.000
Flash (KB)	512	128 programmi 512 dati	128 programmi 1024 dati	48 programmi	32.000
Radio	Bluetooth	Proprietario 315/433/915MHz	ZigBee	ZigBee	Espandibile con periferiche di comunicazione via porta PCMCIA, CF, RS 232, USB, Ethernet
Banda (kbit/s)	720	15	250	250	

TABELLA 1

Confronto tra diversi hardware per WSN

Crossbow	http://www.xbow.com/
Rockwell	http://wins.rsc.rockwell.com/
BTnode	sviluppati al Politecnico di Zurigo, commercializzati da Vitronics http://www.vitronics.com/
μamps	http://www-mtl.mit.edu/researchgroups/icsystems/uamps/
Medusa	http://nesl.ee.ucla.edu/projects/ahlos/hardware.htm
Intel IMote	http://www.ieee.or.com/Archive/Intel_mote.pdf
EyesIFX Platform	http://doc.utwente.nl/41407
Sensoria WINS	http://www.sensoria.com/
Telos	http://www.moteiv.com/
EmberNet	http://www.ember.com/company/index.html

TABELLA 2

Elenco di progetti su hardware per WSN

ti nei sistemi embedded sono carenti per quanto riguarda i requisiti appena introdotti; per questo sono stati sviluppati dei sistemi operativi ad hoc come TinyOS [60], EYES OS [10] e Contiki [11].
L'implementazione delle politiche di risparmio energetico e di routing delle informazioni costituiscono un fattore cruciale di una WSN. Il risparmio di energia può essere ottenuto spegnendo i sensori di rilevazione quando non vengono utilizzati, "addormentando" l'unità di elaborazione quando non deve svolgere calcoli e soprattutto usando oculatamente l'unità ricetrasmittente, il componente che in assoluto consuma più energia. Un *mote* può decidere di spegnere l'unità ricevente per brevi intervalli di tempo, può sfruttare la comunicazione *multihop*[2] e soprattutto può cercare di elaborare localmente le informazioni prima di trasmetterle. Un sensore che deve calcolare la temperatura media di un ambiente in un giorno, può memorizzare i valori man mano che vengono raccolti e a fine giornata calcolare e trasmettere la media. In questo caso il consumo energetico per memorizzare i valori, elaborarli e trasmetterli solo a fine giornata è notevolmente inferiore rispetto a quanto consumerebbe l'apparato di trasmissione per trasmettere singolarmente ognuno dei valori rilevati.

2.2.1. TinyOS e NesC

Analizziamo ora NesC e TinyOS, rispettivamente il primo linguaggio utilizzato per scrivere le applicazioni per *mote* ed il primo sistema operativo per *mote* (scritto in NesC), entrambi sviluppati presso l'Università di Berkeley. TinyOS è diventato uno standard industriale, sia per essere stato il primo sistema operativo per WSN sia perché i suoi sorgenti sono pubblici e liberamente utilizzabili.

I *mote* sono dei dispositivi di elaborazione particolari, perché "dormono" per la maggior parte del tempo. L'elaborazione svolta è guidata dagli eventi: solo quando il sensore acquisisce nuovi dati oppure quando vengono ricevuti dei messaggi si avvia un processo di calcolo.
L'unità base di programmazione e di compilazione nel linguaggio NesC è il "componente", ogni programma è ottenuto assemblando uno o più componenti tra loro. Ogni componente è specificato da una o più interfacce, le quali, in modo simile a quanto fatto da altri linguaggi, dichiarano un insieme di funzioni e procedure da implementare chiamate "comandi". A differenza di linguaggi come C e JAVA, le interfacce NesC offrono dei costrutti sintattici espliciti sia per specificare il codice di gestione degli eventi (la porzione di codice che deve essere avviata al verificarsi di un particolare tipo di evento) sia per gestire l'accoppiamento tra i componenti che producono eventi e i componenti interessati alla notifica.
I componenti NesC a loro volta si dividono in due categorie: i moduli e le configurazioni. I moduli contengono il codice che implementa una o più interfacce; un componente di configurazione (più brevemente, configurazione) viene utilizzato invece per specificare come assemblare più moduli tra loro. L'obiettivo principale di quest'approccio è poter sviluppare separatamente componenti che possono essere poi facilmente composti tra loro.
TinyOS è formato da un insieme di routine di sistema (implementate per mezzo di componenti NesC) e da un componente che si occupa di pianificare l'esecuzione dei "thread". In un *mote* ci sono solamente due thread in esecuzione: l'esecutore dei "task" e l'esecutore dei "gestori degli eventi hardware". I task sono blocchi di codice applicativo, i gestori degli eventi hardware si occupano di gestire le interruzioni generate dall'hardware del *mote*, in particolare le interruzioni generate dai sensori e dall'unità ricetrasmittente.
Le chiamate di sistema TinyOS che richiedono elaborazioni lunghe in termini di tempo non sono bloccanti, ma vengono eseguite in modo asincrono: la loro invocazione restituisce subito il controllo al chiamante ed il completamento del comando è segnalato per mezzo di un evento. Per esempio, l'interfaccia che offre il comando per trasmettere dati via radio ri-

[2] Si ha comunicazione *multihop* quando un messaggio non viene trasmesso direttamente dal mittente al destinatario, ma per mezzo di un insieme di nodi, ognuno dei quali riceve un messaggio e lo ritrasmette al vicino, formando un percorso che congiunge il mittente con il destinatario. La trasmissione *multihop* permette di coprire lunghe distanze pur avendo a disposizione apparati di comunicazione a basso consumo energetico in grado di coprire solo piccole distanze.

chiede al chiamante di gestire l'evento "sendDone", generato dal sistema operativo, che informa che il pacchetto è stato inviato.

2.2.2. Macchine virtuali

I programmi per WSN vengono sviluppati su desktop, provati con dei simulatori [61] e poi trasferiti sui *mote*. Per quest'ultima operazione si utilizza la "programming board", una scheda che si collega ad una porta del PC, sulla quale si può inserire un *mote* e che permette di scrivere direttamente sulla memoria flash del *mote* stesso. Con questo metodo è necessario riprogrammare i *mote* uno ad uno ed inoltre non è possibile riprogrammare quelli già disposti sul campo. Sarebbe auspicabile utilizzare le capacità di comunicazione dei *mote* per distribuire un nuovo programma, tuttavia la maggior parte dell'hardware non permette di riscrivere via radio la memoria delle applicazioni.
Questo inconveniente può essere superato eseguendo su ogni *mote* una "macchina virtuale": un interprete di codice macchina che gira come un normale programma. La macchina virtuale esegue del codice che può essere trasmesso come se fosse una normale sequenza dati, quindi sfruttando le potenzialità di comunicazione di una rete di *mote*. Le istruzioni interpretate dalle macchine virtuali sono generalmente istruzioni di più alto livello di quelle offerte dall'hardware e da TinyOS, per questo motivo il codice prodotto ha dimensioni inferiori ed è quindi più facile da trasmettere. A differenza di un programma NesC, il quale viene eseguito direttamente dall'hardware del *mote* senza alcun controllo, la macchina virtuale può controllare che il programma interpretato non esegua operazioni pericolose che possano portare a bloccare il *mote*, come scrivere in zone di memoria al di fuori di quelle assegnate. Esempi di applicazioni o sistemi operativi che implementano al loro interno delle macchine virtuali sono: Mate [19], Bombilla [38], Sensorware [57], MagnetOS [51].

2.2.3. La gestione dei dati: TinyDB

I *mote* di una WSN hanno la capacità di acquisire informazioni autonomamente dall'ambiente esterno e hanno capacità di elaborazione che possono essere utilizzate per filtrare ed in generale per pre-elaborare localmente i dati raccolti. Queste capacità sono state sfruttate in modo interessante da TinyDB [20], un software che permette di creare una rappresentazione astratta di un insieme di *mote* basata sul paradigma del database. TinyDB permette di recuperare informazioni eseguendo interrogazioni espresse in linguaggio SQL come se ci si trovasse di fronte ad un database vero e proprio. L'architettura software sottostante a TinyDB si occupa di inviare la query ai *mote* interessati, di farla eseguire in locale, di raccogliere le informazioni e di presentarle all'interrogatore, eventualmente in forma aggregata. Essa si occupa anche di minimizzare il traffico di rete, per esempio cercando di inviare le query solamente ai nodi che dispongono delle informazioni cercate e aggregando i dati raccolti dai nodi vicini prima di ritrasmetterli. Il linguaggio di TinyDB include anche un paradigma attivo, che prevede la possibilità di scrivere query che vengono scatenate periodicamente o in presenza di determinati eventi.

2.3. Un caso applicativo

Le reti di sensori possono essere impiegate nei più svariati ambiti e contesti; a titolo di esempio nel riquadro si riportano i riferimenti ad alcune applicazioni, suddivise per cate-

Esempi di applicazioni basate su WSN

Controllo di spazi
Controllo di ambienti ed habitat naturali [44] [16].
Agricoltura di precisione: si rilevano dati ambientali per determinare interventi da compiere, periodo di raccolta o di vendemmia [8].
Climatizzazione di interni [2] [47] [46].
Sorveglianza [54].

Controllo di oggetti
Controllo strutture [39] [17] [32] [45].
Ecophysiology: si utilizzano i sensori per controllare che i parametri di un ambiente rientri nei normali valori di funzionamento [6].
Gestione e controllo motori ed equipaggiamenti [3].
Diagnostica medica [26].
Mappatura di territori [22].

Controllo dell'interazione tra oggetti, persone ed ambiente
Ambienti domestici [59] [15].
Interazione tra persone [28].
Controllo traffico veicoli [62].
Controllo habitat selvaggi [29].
Gestione di catastrofi [42].
Interventi di emergenza [33].
Ubiquitous computing environments [18].
Asset tracking: localizzazione di container, beni di valori [34].
Sanità [21] [27].
Flussi di processi produttivi [12].

gorie, che utilizzano le WSN. In questo paragrafo descriveremo come le reti di sensori sono state utilizzate sul Golden Gate di San Francisco (USA).
Poiché il ponte vibra per effetto del traffico e del vento, arrivando ad oscillare di alcune decine di centimetri quando sottoposto a forti raffiche di vento o a terremoti, sulla struttura sono stati disposti dei *mote* dotati di accelerometri in grado di rilevare le sollecitazioni subite dal ponte. I dati raccolti vengono inviati ad un'apposita stazione per essere consultati dal personale che sovrintende alla manutenzione del ponte [45].
Per misurare le oscillazioni del ponte è stata preferita una WSN ad altri strumenti di misura basati su cavi poiché i *mote* possono essere posizionati con maggior facilità nei punti più difficilmente accessibili.
L'elevata mole di dati originata dal processo di rilevamento e la ridotta banda di trasmissione disponibile sono stati gli aspetti più critici da gestire nello sviluppo del software di controllo della WSN impiegata sul Golden Gate. In 5 min di rilevazioni, 100 *mote* disposti sul ponte raccolgono 24 Mbyte di dati, i quali devono essere tutti trasmessi ad un *mote* che funge da "gateway" tra la rete di sensori e il centro di rilevazione dati. Il *mote* gateway costituisce un collo di bottiglia, in quanto il canale radio attraverso cui riceve i dati dagli altri *mote* della rete ha una banda effettiva[3] di 1,2 kbyte/s e per trasmettere i dati raccolti con tale banda a disposizione sarebbero necessarie più di 5 h [50]. Anche aumentando il numero di gateway tra rete wireless e stazione di raccolta dati non si riuscirebbero ad ottenere prestazioni accettabili. In altri contesti applicativi sarebbe possibile sfruttare le capacità di elaborazione locali di ogni *mote* per ridurre le dimensioni dei dati da trasmettere, tuttavia nel caso del Golden Gate i dati raccolti sui movimenti della struttura devono essere trasmessi per intero.

[3] Sono stati utilizzati dei *mote* Mica2 che offrono una banda massima teorica di 15 kbit/s (circa 2 kbyte/s) che, per effetto della ritrasmissione dei pacchetti corrotti e dell'inserimento di informazioni di controllo da parte del protocollo di comunicazione impiegato, si riduce ad una banda effettiva di 1,2 kByte/s.

Per superare i problemi esposti ogni *mote* dispiegato sul ponte è stato programmato in modo da memorizzare temporaneamente i dati raccolti nella memoria flash e per trasmetterli verso la stazione di rilevamento solo alla fine del periodo di rilevazione (o all'esaurirsi della memoria).
Dato che i dati sono trasferiti alla stazione di rilevamento in differita, è necessario contrassegnare ogni rilevamento con un *time stamp* (orario di rilevamento). Poiché gli orologi interni dei *mote* non sono sincronizzati tra loro, dopo l'invio dei dati è necessario ricondurre i vari *time stamp* ad un unico orario. Un protocollo sviluppato *ad hoc* si occupa di determinare la differenza di orario tra i *mote* e la stazione di rilevamento, in modo da poter correggere i *time stamp*.
Per svolgere le rilevazioni i *mote* sono stati depositati sulle strutture del ponte per poi essere rimossi una volta esaurita la batteria. Sono allo studio soluzioni per dotare il ponte di sensori permanenti. In questo caso il problema da risolvere è come fare in modo che i *mote* possano funzionare per lungo tempo senza essere connessi alla rete elettrica (soluzione che richiederebbe la messa in posa di cavi).
I *mote* fino ad ora sono stati disposti in modo tale da essere in contatto diretto con la stazione di rilevamento. Sono allo studio forme di comunicazione *multihop*: dato che i *mot* vengono spesso disposti "in fila" (per esempio sui tiranti del ponte), questa topologia permette di implementare un semplice algoritmo di routing: è sufficiente che un *mote* inoltri i pacchetti ricevuti al *mote* successivo della fila per fare in modo che le informazioni giungano alla stazione di rilevazione. È in corso di valutazione la possibilità di impiegare dei *super-mote* che raccolgano i dati raccolti dai *mote* normali circostanti e li inviino alla stazione di raccolta per mezzo di un canale radio a banda maggiore.

3. RFID

Un sistema RFID è composto da due elementi principali: il lettore (o *reader*) ed il *transponder*. I *transponder* spesso sono montati su etichette e per questo motivo vengono comunemente chiamati etichette o con il corrispondente termine inglese "*tag*". Il lettore è

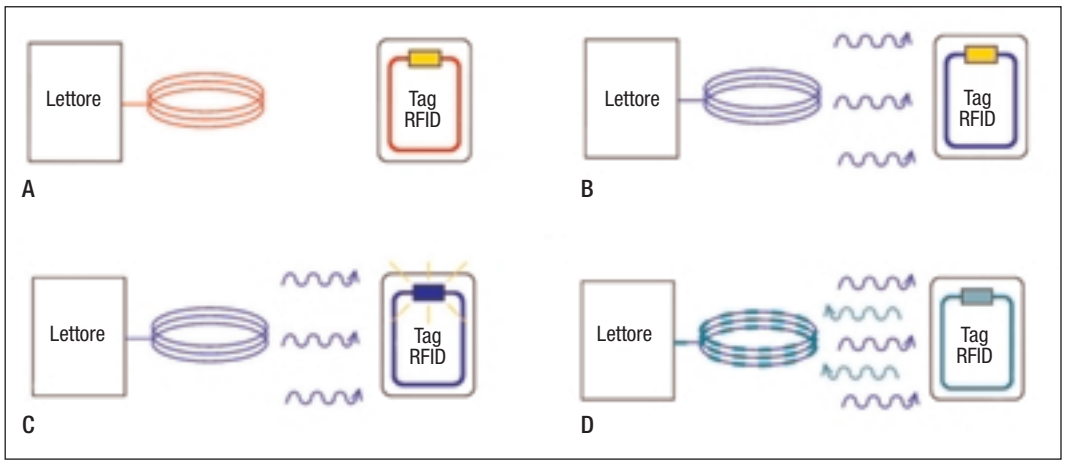


FIGURA 3

Esempio di funzionamento di un transponder RFID: **A** *Situazione a lettore spento.* **B** *Il lettore genera un campo elettromagnetico che alimenta la bobina del transponder.* **C** *Il circuito del transponder esegue delle elaborazioni.* **D** *Il transponder restituisce i risultati delle elaborazioni*

un dispositivo ricetrasmittente che svolge una duplice funzione: fornisce energia, con la quale un *transponder* può funzionare, e dialoga con il *transponder* trasmettendo comandi e raccogliendo risposte. Un *transponder* è un piccolo chip accoppiato ad una antenna; quando il *transponder* entra nel campo elettromagnetico generato dal lettore, sull'antenna del *transponder* (che in molti casi assomiglia ad una bobina) viene indotta una corrente che alimenta il chip. Il chip elabora il comando ricevuto e trasmette indietro il risultato delle elaborazioni. Il processo è illustrato nella figura 3.

Ogni *transponder* (*tag* d'ora in poi) è dotato di un identificatore univoco (ID), che lo distingue dagli altri. Ad ogni operazione di interrogazione, un *tag* risponde fornendo l'ID che lo contraddistingue; in questo modo diventa possibile identificare oggetti e merci usando i *tag* RFID come se fossero dei marcatori, in maniera analoga ad un codice a barre. Un codice a barre per essere letto deve essere illuminato con una certa precisione da un lettore laser, in caso contrario la lettura non riesce[4]; un RFID non soffre di questo problema perché utilizza le onde radio le quali non richiedono che trasmittente e ricevente siano tra loro visibili. L'impiego di onde radio permette addirittura di rilevare tutti i *tag* RFID contenuti all'interno di uno scatolone senza doverlo aprire. Un'altra differenza fondamentale tra codici a barre ed RFID è il tipo di codifica dei prodotti adottata. Un codice a barre permette di identificare i prodotti solamente a livello di categoria (esempio, le marmellate alla fragola prodotte dall'azienda del signor Rossi hanno tutte lo stesso codice a barre), mentre l'ID di un RFID permette anche di distinguere tra loro prodotti della stessa categoria.

Questa caratteristica, unita con la possibilità di leggere i dati di un *tag* RFID a distanza, apre innumerevoli possibilità, prima fra tutte la possibilità di eseguire gli inventari in modo quasi istantaneo. È sufficiente far passare gli oggetti da inventariare vicino ad un lettore RFID per identificare tutti gli oggetti corredati di *tag*. Con i codici a barre sarebbe necessario leggere ad uno ad uno i codici degli oggetti; inoltre ci sarebbe sempre il rischio di contare un oggetto più volte, se l'operatore passasse per sbaglio il lettore su un oggetto "già letto".

3.1. Categorie di tag

Esistono vari tipi e varie categorie di *tag*: si distinguono in base alla presenza o meno di una fonte di alimentazione a bordo, alla capacità ed alla riscrivibilità della memoria, al tipo di codice identificativo adottato, alla frequenza di lavoro e allo standard utilizzato per

[4] È esperienza comune vedere il cassiere di un supermercato costretto a passare più volte un prodotto sopra il lettore ottico della cassa prima che la lettura riesca.

la trasmissione dati. I *tag passivi* non hanno batterie o fonti di alimentazione a bordo, i *tag attivi* hanno una fonte di alimentazione e trasmettono continuamente informazioni, i *tag semipassivi* hanno una fonte di alimentazione e trasmettono informazioni solo quando sono interrogati. Per esempio, il sistema telepass utilizzato nelle autostrade italiane utilizza dei *tag* semipassivi. Invece il dispositivo arva [35] usato dagli escursionisti alpini per rintracciare le persone sepolte dalle slavine costituisce un esempio di *tag* attivo. Nel seguito di questo articolo ci occuperemo solo dei *tag* passivi, per i quali ci si attende una notevole diffusione nel mercato nei prossimi anni, soprattutto a causa delle dimensioni e dei costi di produzione mediamente inferiori rispetto agli altri tipi di *tag*. Le dimensioni ridotte facilitano l'inserimento dei *tag* passivi all'interno degli oggetti ed il basso costo di produzione è un fattore chiave per la diffusione di massa della tecnologia.

I *tag* con memoria a sola lettura (*tag Read Only*, RO in breve) memorizzano solamente il codice identificativo univoco impostato in fabbrica. I *tag* scrivibili (*Read Write*, RW in breve), che pure sono sono dotati di un codice identificativo univoco, dispongono anche di memorie flash sulle quali un utente può leggere e scrivere informazioni. La scrittura delle informazioni, in maniera analoga a quello che accade per la lettura, è comandata a distanza dal lettore che con il suo segnale fornisce sia i comandi sia l'energia necessaria per far svolgere al *tag* l'operazione.

Esistono due categorie di standard per il codice identificativo di un *tag*, una portata avanti dall'ISO e un'altra portata avanti da EPC Global[5]. Gli approcci seguiti dai due organismi sono molto diversi tra loro, come riassunto nel riquadro di p. 26. Stando alle più recenti informazioni rese disponibili dalla stessa EPC Global, è ormai probabile che lo standard EPC confluisca in un apposito standard ISO, attualmente in corso di negoziazione tra le due parti.

Per quanto riguarda la frequenza delle onde elettromagnetiche utilizzate, i *tag* si possono classificare in 4 categorie: tag a bassa frequenza o LF (da 125 a 134 kHz), *tag* ad alta frequenza o HF (13.56 MHz), *tag* UHF (ultra high frequency 868 – 956 MHz) e *tag* a microonde (2.45 GHz). I *tag* a bassa ed alta frequenza possono essere usati in tutto il mondo liberamente e senza licenze. Per i *tag* UHF invece ci sono delle regole discordanti da Paese a Paese. In Europa i *tag* maggiormente utilizzati sono quelli HF. L'impiego di *tag* UHF e a microonde in Europa è reso difficoltoso dalle norme che disciplinano la telefonia cellulare, l'assegnazione di frequenze televisive e la sicurezza del lavoro. Nel mercato nord americano, una differente disposizione delle frequenze della telefonia cellulare e della tv e leggi più permissive nel campo della sicurezza del lavoro facilitano l'utilizzo di *tag* UHF.

All'aumentare della frequenza aumenta la distanza massima di lettura e scrittura di un *tag*. I *tag* LF, pur avendo delle distanze di lettura inferiori alle altre categorie di *tag*, presentano una maggior insensibilità alla presenza di metalli e liquidi. Per questo motivo vengono utilizzati per la marcatura di animali (il cui corpo presenta una elevata quantità di acqua).

3.2. Casi d'uso

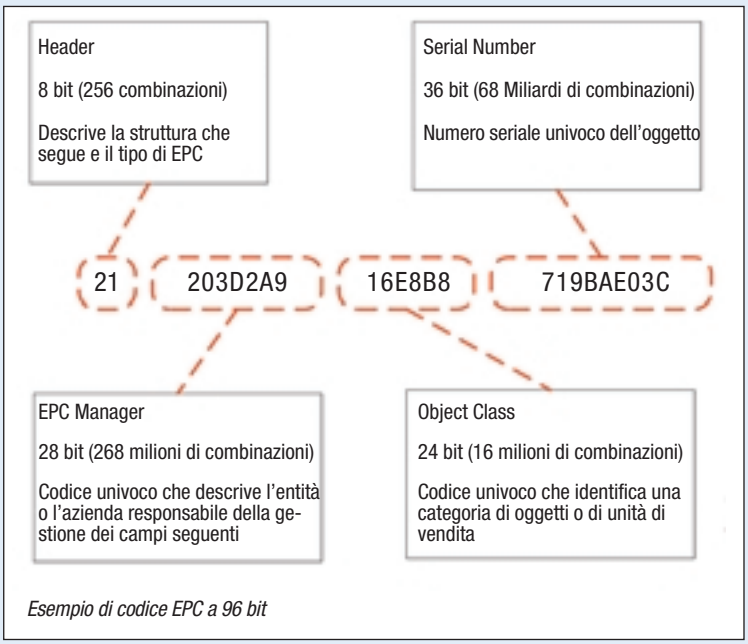
Numerose applicazioni e prototipi innovativi che sfruttano la tecnologia RFID sono stati sviluppate da industrie e centri di ricerca; questi lavori possono essere classificati in base a diversi criteri: il settore industriale di riferimento, il grado di maturità dell'applicazione della tecnologia, il tipo o i tipi di processi aziendali che l'applicazione va ad influenzare.

Nelle tabelle 3 sono riportati dei casi d'uso raggruppati a seconda dell'attività per la quale è stata impiegata la tecnologia RFID. La classificazione è stata ispirata da [1]. Le attività sono state divise principalmente in tre categorie: nella prima i tag RFID sono utilizzati per identificare e tracciare il flusso o lo stato di oggetti (o di animali), nella seconda per tracciare e determinare le azioni di individui, nella terza per migliorare le funzionalità

[5] EPC Global Inc. è un'organizzazione no-profit nata con lo scopo di standardizzare i codici di prodotto impiegati all'interno degli RFID. EPC Global è nata da un'iniziativa del MIT (Massachusetts institute of technology di Boston) e attualmente è un network di aziende ed organizzazioni. Il rappresentante italiano di EPC Global è Indicod-Ecr [49].

Confronto tra ISO ed EPC

L'obiettivo principale di EPC Global è la creazione dell'EPC (*Electronic Product Code* "Codice prodotto elettronico"), cioè un codice che permetta di identificare facilmente ogni prodotto, fornendo informazioni sul produttore, sul Paese di provenienza ecc., in maniera analoga ad un codice a barre. Lo standard EPC definisce tre schemi di codici: a 96, a 128 e a 256 bit. Lo schema del codice a 96 bit, riportato nella figura fornisce un identificatore univoco diviso in tre parti più un header: azienda, classe di oggetto e prodotto. La prima parte del codice identifica l'azienda produttrice del bene, e permette di identificare 268 milioni di aziende. Ogni azienda può gestire 16 milioni di classi di oggetti, ognuna delle quali può avere 68 miliardi di numeri seriali diversi. Secondo lo schema EPC, ogni prodotto è contrassegnato da un tag il quale ha come identificatore univoco l'EPC del prodotto stesso. Un *tag* pur potendo ospitare delle informazioni aggiuntive, secondo lo schema EPC deve memorizzare solamente il codice EPC, le altre informazioni che contraddistinguono un prodotto devono essere fornite da una rete e da un sistema informativo on-line che permettono di recuperare le informazioni associate al *tag*. In merito a quest'ultimo punto EPC definisce in modo rigido un'infrastruttura per interconnettere i sistemi informativi di più soggetti al fine di condividere le informazioni associate ai *tag*.



Esempio di codice EPC a 96 bit

Lo standard ISO è concepito in modo completamente diverso: le informazioni aggiuntive rispetto ad un ID che contraddistinguono un oggetto risiedono per la maggior parte sul *tag* stesso, o al più su un database e su delle applicazioni locali all'azienda. In questa architettura non vi è una standardizzazione del sistema che si occupa dello scambio di informazioni tra diversi sistemi informativi. Infine per quanto riguarda i criteri per specificare l'ID dei tag, ISO definisce uno standard aperto, delegando al singolo produttore di *tag* la definizione delle caratteristiche dell'ID utilizzato dal *tag* stesso. Nella tabella è riportato l'elenco degli standard ISO di interesse per i *tag* RFID. Sia ISO sia EPC Global si sono occupati della standardizzazione delle frequenze radio utilizzate oltre alla standardizzazione dei codici identificativi dei *tag*. EPC Global, a differenza di ISO, ha svolto in questo campo un lavoro marginale il cui principale obiettivo è stato la creazione di un hardware standard che possa ospitare l'EPC. Gli sforzi di EPC Global nel campo delle frequenze si sono concentrati principalmente sulla banda UHF, per la quale nel mercato nord americano esistono meno vincoli per l'utilizzo rispetto al mercato europeo.

ELENCO DEGLI STANDARD ISO DI INTERESSE PER RFID

ISO/IEC 18001	Information Technology, AIDC Techniques, RFID for Item Management, Application Requirement Profiles	Published Standard 2004
ISO/IEC 18000_1	Generic Parameters for Air Interface Communication for Globally Accepted Frequencies	Published Standard 2004
ISO/IEC 18000_2	Parameters for Air Interface Communications below 135kHz	Published Standard 2004
ISO/IEC 18000_3	Parameters for Air Interface Communications at 13.56 MHz	Published Standard 2004
ISO/IEC 18000_4	Parameters for Air Interface Communications at 2.45 GHz	Final Draft Intern. Standard
ISO/IEC 18000_6	Parameters for Air Interface Communications at 860_930 MHz	Published Standard 2004
ISO/IEC 15961	RFID for Item Management _ Data protocol: Application Interface	Published Standard 2004
ISO/IEC 15962	RFID for Item Management, Protocol: Data encoding rules and logical memory functions	Published Standard 2004
ISO/IEC 15963	RFID for Item Management, Unique Identification of RF Tag	Final Draft Intern. Standard

Supporto all'attività produttiva principale	Settore agricolo e dell'allevamento • Identificazione animali: contrassegnando gli animali con dei tag si snelliscono le procedure burocratiche connesse con la loro compravendita [1]. • Controllo animali: contrassegnando ogni animale con un tag si possono riconoscere e fornire automaticamente alimentazione, cure veterinarie personalizzate all'animale [1]. • Riconoscimento lotti prodotti agricoli: all'ingresso di serre sono applicati tag RFID su cui vengono registrati i dati del lotto di sementi utilizzate e l'insieme delle attività svolte sulle colture [1].
	Settore manifatturiero • gestione dell'avanzamento lavori lungo le linee di assemblaggio: i lotti di semilavorati vengono contrassegnati con tag sui quali si registrano le lavorazioni effettuate e le lavorazioni da effettuare [1] [30] [65].
	Settore dei servizi • Automazione operazione di ingresso/uscita/rifornimento mezzi di trasporto: l'applicazione di un *tag* agli automezzi facilita il riconoscimento dei mezzi ai varchi di accesso al deposito e ad aree del centro storico, oppure serve per selezionare automaticamente il carburante più adatto nelle stazioni di rifornimento [1]. • Life cycle management di attrezzature: macchinari sottoposti a manutenzione periodica vengono contrassegnati con tag sui quali si registrano le operazioni di manutenzione effettuate [1]. • Gestione medicinali in ospedale: tag sono applicati sulle confezioni dei farmaci, in questo modo può essere automatizzata la contabilità dei prelievi e restituzione dei farmaci nei singoli armadietti da parte dei medici [1]. • Gestione delle supply chain (catene di approvvigionamento) [64 - 53 - 36]. • Utilizzo dei *tag* RFID per aumentare l'affidabilità e la velocità di elaborazione dei centri di smistamento pacchi di corrieri e spedizionieri [56]. • Gestione bagagli negli aeroporti [25 - 7 - 14]. • Gestione biblioteche: Inserendo un tag in ogni libro, è possibile velocizzare le operazioni di prestito e restituzione [5].
Logistica di magazzino	• Identificazione univoca delle unità di movimentazione nei magazzini: applicando un *tag* ai contenitori di magazzino è possibile snellire le procedure di inventario, contabilità degli ingressi e delle uscite [1]. • Controllo ricevimento merci in ingresso: applicando un *tag* ad ogni prodotto è possibile conoscere il contenuto di un contenitore di prodotti (o di una qualsiasi altra unità di movimentazione) senza doverlo aprire [1]. • Impiego in magazzini con condizioni climatiche difficili: l'utilizzo dei *tag* consente di velocizzare le operazioni di riconoscimento prodotti, di avere meno manodopera umana all'interno del magazzino ed infine i tag RFID (non dovendo stare all'esterno delle confezioni come i codici a barre) subiscono meno il degrado causato dalle condizioni ambientali [1].
Tracciabilità dei prodotti lungo una filiera	Su un *tag* che contrassegna un prodotto, ogni componente di una filiera produttiva può memorizzare le sue generalità e le lavorazioni svolte, in tal modo il *tag* diventa un registro della filiera produttiva [1].
Gestione di asset	I *tag* vengono utilizzati per contrassegnare e tracciare oggetti di valore, per esempio contenitori (damigiane, container, ...) che transitano tra più operatori [1].
Gestione dei punti vendita	Contrassegnando con un *tag* i beni venduti da un negozio, è possibile automatizzare le operazioni di pagamento in cassa e di inventariazione della merce. Attualmente in Italia sono allo studio soluzioni per ottenere questi obiettivi, però non esiste nessuna realizzazione pratica di larga scala [1], negli Stati Uniti sono stati computi degli esperimenti: [23 - 43].
Sensori di stato	*Tag* RFID che cambiano irreversibilmente stato se sottoposti a condizioni ambientali particolari. Sono adatti per verificare l'interruzione della catena del freddo di un prodotto [31 - 24].

TABELLA 3/1

RFID associati ad oggetti ed animali

Bigliettazione elettronica	Si vuole sostituire il tradizionale biglietto o abbonamento cartaceo per i mezzi pubblici con un badge avente un tag RFID che memorizza il contratto di abbonamento e la validazione delle singole corse. Il tag permette di automatizzare le operazioni di pagamento dell'abbonamento, diventa inoltre possibile creare biglietti ricaricabili con credito a scalare. Le amministrazioni pubbliche possono infine ottenere informazioni aggregate sui tragitti giornalieri compiuti dalle persone e utilizzare queste informazioni per migliorare il servizio [1 - 55].
Car sharing nel trasporto pubblico locale	Sono allo studio l'utilizzo di tag RFID per velocizzare le operazioni di ritiro dei mezzi prenotati [1].
Ticketing	Si utilizzano dei badge con tag RFID per gestire l'ingresso a particolari eventi, manifestazioni, centri sportivi e per la fruizione dei servizi accessori [58 - 1].
Customer Relationship Management ed edutainment	Si tratta di un'evoluzione delle operazioni di ticketing: il riconoscimento della persona e il tracciamento del suo comportamento consentono di fornire all'utente servizi personalizzati o migliorare il livello di servizio fornito [1].
Controllo accessi nei luoghi di lavoro	Si utilizzano dei badge con tag RFID per il controllo accessi, i quali possono essere utilizzati anche per accedere ad altri servizi interni all'azienda (es. mensa). In alcune applicazioni, i badge memorizzano dati biometrici cifrati da utilizzare per la validazione dell'identità della persona. Dato che le informazioni non sono memorizzare nei server aziendali, per l'azienda non sussistono problemi per la gestione della privacy [1 - 48].
Identificazione dei pazienti in ambito ospedaliero	L'applicazione di tag sui pazienti (per mezzo di braccialetti) congiuntamente all'applicazione su materiali (esempio, sacche per trasfusione) possono fornire al personale utili indicazioni per la corretta gestione dei processi ospedalieri e prevenire somministrazioni scorrette di farmaci o applicazione di trattamenti errati [1].

TABELLA 3/2

RFID associati a individui

Misura pressione pneumatici	Un tag RFID munito di sensore di pressione è inserito all'interno di un pneumatico, un lettore RFID è inserito all'interno del parafango, in questo modo è possibile determinare la pressione di una gomma in tempo reale [67].
Utilizzo di lettori RFID all'interno di lavatrici e frigoriferi	Una lavatrice è in grado di leggere le istruzioni di lavaggio contenute nei tag attaccati ai vestiti. Il frigorifero può controllare la presenza di alimenti scaduti ed eventualmente ordinarne di nuovi [52].
Autenticazione documenti cartacei	Nel processo di firma digitale di documenti per garantire l'univocità (oltre che l'autenticità) di un documento [41].

TABELLA 3/3

RFID utilizzati per migliorare tecnologie esistenti

di tecnologie esistenti. All'interno di ognuna di queste categorie sono state individuate delle sottocategorie, per ognuna delle quali sono state riportate informazioni sui singoli casi applicativi e dei riferimenti per l'approfondimento.

Ora analizzeremo in dettaglio un caso d'uso della tecnologia RFID.

Sistema di messaggistica. Al Politecnico di Milano, è stato svolto un progetto sperimentale [13] nel quale si sono utilizzati i *tag* RFID per sostituire la procedura di autenticazione basata su username e password di un sistema di messaggistica. Il sistema utilizza dei pannelli a messaggi variabili per fornire informazioni ad utenti che si muovono all'interno di un certo ambiente, quale un campus universitario, una stazione o un aeroporto. Esso consente di veicolare messaggi diretti ad un vasto gruppo di persone (esempio, "I passeggeri del volo XYZ 123 sono pregati di presentarsi al gate 2" oppure "La lezione di Informatica A è stata spostata nell'aula So1").

Ogni volta che una persona si avvicina ad un

pannello a messaggi variabili, il pannello rileva il *tag* RFID contenuto nel badge di riconoscimento della persona e visualizza i messaggi destinati all'utente. Il *tag* RFID permette di rilevare la presenza dell'utente e di identificarlo senza un intervento esplicito dell'utente stesso, come invece digitare lo username o strisciare un badge magnetico su un lettore. Se più persone sono vicine ad un pannello, lo schermo del pannello si divide in parti uguali, ognuna delle quali mostra le informazioni dirette ad un singolo utente. Qualora vicino al pannello siano presenti un numero di utenti tale da portare ad una eccessiva frammentazione dello schermo, vengono visualizzati solo i messaggi comuni ai vari utenti. Il sistema gestisce la suddivisione di utenti in gruppi e permette di inviare messaggi di gruppo, che verranno poi notificati a tutti i membri, oppure messaggi con destinatario non specificato, che verranno recapitati a tutte le persone che condividono certi interessi.
Il *tag* contenuto nel badge ospita informazioni sugli interessi dell'utente, le quali sono utilizzate dal sistema per decidere quali messaggi proporre all'utente stesso. Le informazioni memorizzate sui *tag* sono mantenute ed aggiornate personalmente dall'utente (avvalendosi di appositi chioschi che permettono di leggere e scrivere il contenuto dei *tag*) e sono ospitate solamente sul badge. In questo modo un utente può ricevere messaggi tarati sui suoi gusti senza che il sistema debba gestire centralmente le sue informazioni di preferenza. Il sistema è particolarmente indicato per veicolare informazioni generali che interessano un largo pubblico, può essere usato per inviare informazioni anche a singole persone, tuttavia non è adatto per trasmettere informazioni di carattere strettamente personale dato che i messaggi vengono visualizzati da schermi posti in luoghi pubblici.

3.3. Gestione distribuita dei dati

I *tag* RFID permettono di realizzare in modo semplice ed economico un sistema informativo "distribuito" e pervasivo cui partecipano, oltre a componenti fissi, diversi dispositivi mobili come palmari, telefoni cellulari e smartcard ed eventualmente componenti per la raccolta di informazioni del territorio come i *mote*.
I *tag* RFID consentono di memorizzare le informazioni in formato elettronico direttamente sugli oggetti, dando modo agli operatori di recuperarle e (nel caso di *tag* RW) aggiornarle agevolmente. Per esempio un *tag* RFID attaccato ad un prodotto permette di memorizzare il percorso compiuto, man mano che il *tag* avanza, lungo la filiera produttiva.
Il fatto che dei "dati elettronici" viaggino assieme alla merce che caratterizzano, diventa un comodo strumento per scambiare informazioni tra i sistemi informativi delle società e degli enti coinvolti nella vita di un prodotto. Salvare le informazioni alla "periferia del sistema informativo" significa metterle "nelle mani" degli attori che principalmente le utilizzano. Ciò offre delle semplificazioni dal punto di vista dell'accesso alle informazioni, ma solleva delle problematiche riguardanti la sicurezza e la riservatezza delle stesse. Non entriamo qui nel dettaglio di questa discussione; vogliamo però segnalare che le nuove tecnologie offrono possibilità per affrontare questi problemi. La sede di New York dell'American Express utilizza un sistema composto da *tag* RFID e lettore di impronte digitali per gestire il controllo degli accessi dei dipendenti [48]. L'aspetto innovativo sta nel fatto che le impronte digitali di un dipendente sono memorizzate su un *tag* RFID presente all'interno del badge assegnato al dipendente stesso[6]. Quando un utente deve passare per un punto di controllo, preme il suo dito su un lettore di impronte, l'impronta viene rilevata e confronta con le informazioni memorizzate sul *tag* RFID contenuto nel badge dell'utente. Il sistema di accessi della società non memorizza l'impronta dell'utente ma solo un codice di controllo dei dati del badge per poter riconoscere eventuali manomissioni. Questo approccio evita alla società di dover gestire direttamente le informazioni di natura biometrica dell'utente, i quali rappresentano "dati sensibili" che richiederebbero un trattamento particolare per garantire la riservatezza.

[6] Per essere precisi sul badge non è salvata una descrizione completa dell'impronta di una persona ma solo uno schema (un sottinsieme di punti dell'impronta) che può essere usato per la verifica dell'impronta. Lo schema non permette di risalire all'impronta dal quale è stato generato, ma permette di identificarla.

Salvare i dati su dispositivi mobili, con connessioni transitorie e di capacità variabile, ha come rovescio della medaglia problemi di sincronizzazione e di consistenza dei dati. Per esempio il ricevimento di un pacco da parte di un fornitore richiede lo svolgimento di alcune procedure amministrative e per far ciò è necessario sincronizzare le informazioni gestite dal sistema informativo dell'amministrazione con le informazioni contenute nel *tag* RFID che contrassegna il pacco. Se gli operatori del magazzino non hanno lettori di *tag* collegati con il sistema informativo, l'afflusso delle informazioni a quest'ultimo può subire dei ritardi.
Se più copie della stessa informazione sono memorizzate in dispositivi diversi, c'è il rischio che diversi operatori modifichino le informazioni in modi differenti generando problemi di consistenza delle informazioni. Anche per questo motivo, nel caso in cui i *tag* debbano essere letti ed aggiornati da diversi sistemi informativi, i dati devono essere espressi in un formato comune intelligibile ai vari sistemi coinvolti.
Infine le informazioni contenute nei *tag* possono essere manomesse oppure i *tag* possono subire dei danni accidentali. Nel primo caso le informazioni devono essere protette con opportuni strumenti (crittografia, codici di controllo, . . .). Nel secondo caso una copia delle informazioni può essere sufficiente. La copia può essere memorizzata su un altro *tag* attaccato all'oggetto, oppure su un server del sistema informativo.

3.4. RFID: una metodologia di progettazione

Per avvantaggiarsi delle possibilità offerte dalle nuove tecnologie è necessario ripensare lo sviluppo dei sistemi informativi. L'approccio tradizionale (gestire centralmente tutte le informazioni), dovrà coesistere con la possibilità di memorizzare dati sugli oggetti fisici messa a disposizione dalle nuove tecnologie. Come conseguenza nasce il problema di decidere, data un'informazione, dove memorizzarla e in che maniera gestire le eventuali repliche. Occorre scegliere le informazioni che devono essere disponibili localmente (memorizzate su *tag* RFID oppure su *mote*) in modo tale da minimizzare l'accesso a fonti di informazioni non locali, tenendo tuttavia conto dei limiti di memorizzazione dei dispositivi.
Illustriamo ora un esempio in cui applicheremo una metodologia [4] utile per risolvere il problema accennato.
L'esempio riguarda la gestione delle informazioni mediche e cliniche di un gruppo di pazienti. Ad ogni paziente viene fornito un *tag* RFID che contiene le sue informazioni mediche personali. Le informazioni che il sistema informativo nel suo complesso deve gestire riguardano la storia medica, le patologie ed i ricoveri dei pazienti. Una porzione di queste informazioni deve essere replicata nel *tag* in mano alla singola persona: tale porzione deve essere scelta in modo tale che gli operatori che assistono la persona (per esempio i medici in corsia oppure il medico di base oppure il farmacista) trovino nel *tag* del paziente tutte le informazioni di cui hanno bisogno per svolgere le loro operazioni. Purtroppo la memoria di un *tag* RFID (e dei dispositivi di memorizzazione in generale) è limitata e non è possibile ospitare al suo interno tutte le informazioni inerenti un paziente. Occorre quindi modificare il tipo di informazioni contenute nel *tag* al variare del contesto in cui il paziente si trova (esempio, ricoverato, malato cronico, ...).
Vediamo ora come determinare le possibili combinazioni di informazioni che devono essere disponibili localmente. Per prima cosa introduciamo il concetto di ambiente locale (o più brevemente ambiente): l'insieme delle caratteristiche personali ed ambientali che determina la porzione di dati che deve essere disponibile localmente. L'ambiente guiderà la scelta di quale sottinsieme delle informazioni globali sarà reso disponibile anche localmente. La metodologia di progetto prevede i seguenti passi:
1. Si individuano tutte le informazioni più rilevanti del dominio applicativo e, sfruttando le tecniche e metodologie tradizionali, si disegna lo schema concettuale dell'intero database. Questo schema dovrà contenere tutte le informazioni di interesse, senza distinguere tra quelle che devono essere disponibili localmente e quelle che sono disponibili solo presso il server del sistema informativo.
2. Si identificano le "dimensioni di analisi" che permetteranno di costruire l'ambiente del dispositivo, e quindi individuare i dati che devono essere disponibili localmente. Una

dimensione di analisi è un aspetto comune a più problemi e situazioni, utilizzato con scopi di classificazione ed analisi. Elenchiamo ora alcune dimensioni, che possono essere usate e integrate con altre a seconda del dominio applicativo considerato:

- L'*utilizzatore* del dispositivo fisico al quale sono attaccate le informazioni. A seconda del tipo di utilizzatore cambiano le informazioni che un soggetto desidera avere memorizzate. Per esempio i medici desiderano accedere a informazioni su tutti i loro pazienti, mentre un paziente desidera accedere solo alle sue informazioni personali, magari con un maggior livello di dettaglio.
- L'*argomento di interesse* dell'utilizzatore in un particolare istante. Nell'esempio precedente per un medico gli argomenti possono essere: "ricette" oppure "malattie croniche".
- La *situazione* fa riferimento al fatto che durante la vita dell'utilizzatore, per un determinato argomento, le informazioni desiderate possono cambiare in base alla sua situazione. Nel nostro esempio individuiamo la situazione "paziente ospedalizzato" che si riferisce a persone ricoverate e "paziente regolare" che si riferisce allo stato ordinario di una persona.

Il risultato di questo passo è un elenco di dimensioni di analisi, che fungerà da guida per la scelta delle informazioni da ospitare localmente. Nel nostro esempio l'elenco è: *<utilizzatore, argomento di interesse, situazione>*

3. In questa fase, si sceglie la principale dimensione di analisi (per esempio, rispetto all'utilizzatore) e per ogni valore che essa può assumere viene creata una vista concettuale dei dati di interesse. Nel nostro caso viene creata una vista per i medici, una per i pazienti e una per ogni altro tipo di utente (esempio, amministratore dell'ospedale, infermiere ecc.). Le viste devono essere congruenti con lo schema concettuale globale del database precedentemente creato. Per soddisfare questo requisito, in questa fase può essere necessaria una modifica dello schema concettuale globale.

4. In questa fase si combinano i valori delle dimensioni di analisi, eliminando le combinazioni di valori senza senso. Queste combinazioni vengono chiamate *chunk* (pezzo).

Esempi di *chunk* per un database di informazioni sanitarie sono:

<paziente, malattia cronica, ospedale>
questo *chunk* rappresenta tutte le informazioni di cui un paziente ha bisogno relativamente alle sue malattie croniche quando ricoverato,
<paziente, ricetta, regolare>
questo *chunk* contiene tutte le informazioni di cui un paziente ha bisogno relativamente alle sue ricette in una situazione normale,
<dottore, ricetta, regolare>
questo *chunk* contiene tutte le informazioni sulle ricette che un dottore ha effettuato su pazienti regolari.
Si noti che il *chunk*
<dottore, informazioni amministrative, ospedale>
costituisce invece un esempio di combinazione senza senso nel dominio applicativo considerato.

5. A questo punto, tenendo conto dei *chunk* prodotti nelle situazioni precedenti, si possono disegnare lo schema logico (per esempio, relazionale) del database centrale e gli schemi logici dei *chunk*, che dovranno essere ospitati nei singoli dispositivi locali (esempio, *tag* RFID, smartcard o computer palmari).
Per ogni *chunk* si genera una vista dei dati associati al *chunk* a partire dalle viste associate ai singoli valori della dimensione principale.
I *chunk* generati in questa fase sono raggruppati in modo da definire quali informazioni devono essere immagazzinate in un singolo dispositivo locale. Per esempio, l'RFID di un paziente dovrà contenere tutti i *chunk* relativi alla situazione di un paziente regolare più quelli collegati alle sue malattie croniche e alle sue ricette. Ovviamente, il *chunk* dello specifico paziente è una istanza di quello definito per il paziente generico, e contiene solo i suoi dati. Quando un paziente è ricoverato, i *chunk* relativi alla situazione regolare sono rimossi per lasciare spazio a quelli relativi alla situazione "ospedalizzato".

6. Poiché i *chunk* verranno implementati come viste sullo schema logico globale, occorre tener conto anche di alcune caratteristiche fisiche dei dispositivi locali, come le dimensioni di memoria, e alcune caratteristiche di contesto, come la disponibilità di collegamento con il database centrale, il tempo ed il luogo in cui

queste informazioni sono necessarie e i diritti d'accesso degli utenti. Queste informazioni di contesto possono aiutare a ridurre ulteriormente le informazioni che devono essere disponibili localmente.

4. CONCLUSIONI

Sia la tecnologia RFID sia le WSN aumentano la capacità di interazione dei dispositivi di elaborazione con il mondo reale, riducendo in questo modo il divario tra "mondo digitale" e "mondo fisico".
Le WSN permettono di raccogliere informazioni su un ambiente senza richiedere la messa in posa di cavi. Per questo motivo le WSN possono essere impiegate in tutte quelle aree che, per la difficoltà, per il costo, per i tempi richiesti dalle operazioni di cablaggio, difficilmente possono essere dotate dei tradizionali sensori a filo.
Gli RFID sono dei contenitori di informazioni che possono essere disseminati all'interno di un ambiente, possono essere facilmente identificati, la loro memoria dati (quando presente) può essere letta ed aggiornata da dispositivi elettronici.
Vorremmo far notare che entrambe le tecnologie sono utilizzate per gestire dati: le WSN raccolgono informazioni sul mondo reale (temperatura, suoni, umidità, ...), i *tag* RFID permettono invece di associare oggetti reali con dei dati che possono essere facilmente letti, elaborati ed aggiornati da dispositivi elettronici.
Alcune operazioni possono essere svolte da entrambe le tecnologie: un *mote* di una WSN può essere agganciato ad un oggetto per conservare informazioni, in modo simile ad un *tag* RFID. Ciò è reso possibile dall'alta versatilità dei *mote* che possono essere programmati in modo da comportarsi come *tag* RFID attivi. Un *tag* RFID passivo ha un costo e delle dimensioni inferiori rispetto ad un *mote* di una WSN, per questo nelle situazioni dove le maggiori performance dei *tag* attivi non servono, si preferisce l'utilizzo dei *tag* passivi.
Proponiamo ora un confronto simbolico tra le due tecnologie basandoci su 4 dimensioni di analisi:

- costo unitario
- dimensioni[7]
- massima distanza di comunicazione
- versatilità e prestazioni offerte

I valori delle prime tre dimensioni sono rappresentati negli assi della figura 4, all'interno della quale delle aree di diverso colore evidenziano il posizionamento delle due tecnologie all'interno dello schema.
Come si può osservare dallo schema, il costo ridotto di un *tag* RFID permette di sviluppare applicazioni che ne utilizzano grandi quantità. Le ridotte dimensioni facilitano l'inserimento dei *tag* nell'ambiente o negli oggetti che devono essere contrassegnati. Le maggiori distanze di comunicazione (fino a 100 m) favoriscono invece l'impiego dei *mote*. La maggior capacità di elaborazione (non riportata nello schema) e le maggiori distanze di comunica-

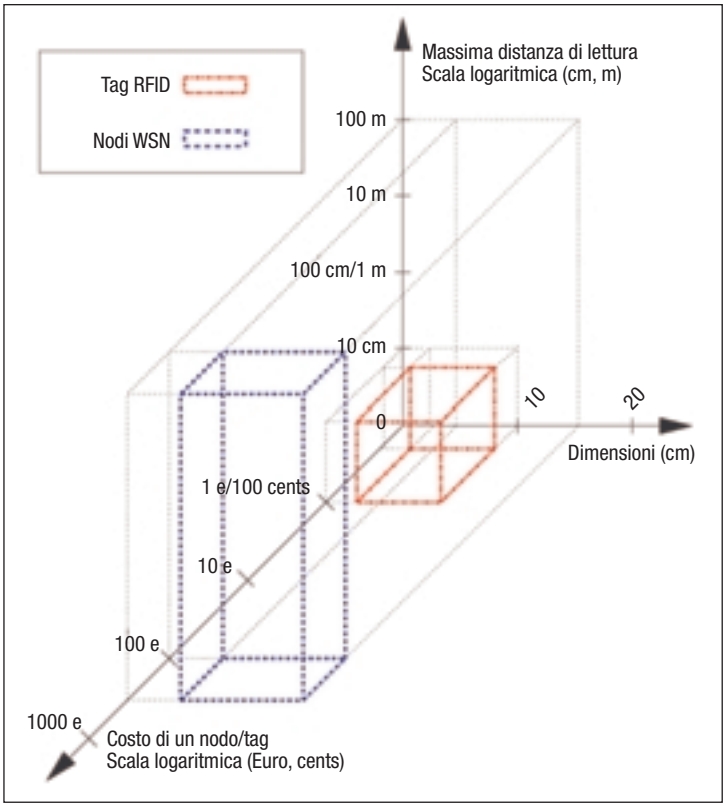


FIGURA 4

Riepilogo delle dimensioni, dei costi e delle distanze di lettura di mote di WSN e di RFID

7 Come dimensioni abbiamo considerato il raggio della più piccola (ipotetica) sfera, capace di contenere l'oggetto in considerazione. Sebbene il volume sarebbe stata una unità di misura più significativa, non è utilizzabile come strumento di confronto poiché gli RFID hanno tutti una altezza quasi nulla e quindi un volume quasi nullo.

zione dei mote tuttavia incidono sul costo di un singolo *mote*, che risulta notevolmente superiore a quello di un singolo *tag*.
Le tecnologie impiegate dagli RFID e dalle WSN sono in costante evoluzione ed è lecito attendersi per i prossimi anni delle novità significative. Ci si aspetta che le dimensioni dei dispositivi, soprattutto dei *mote* delle WSN, diminuiranno sensibilmente fino ad ottenere dei dispositivi dalle dimensioni di pochi mm. Dispositivi di dimensioni minori di quelle attuali molto probabilmente si diffonderanno negli ambienti di lavoro e domestici.
Ambienti attrezzati con *tag* RFID e WSN permetteranno di raccogliere una gran quantità di informazioni, su processi logistici, processi di business e comportamenti abituali delle persone. Ciò richiederà ai sistemi informativi e ai database di gestire un numero molto maggiore di informazioni rispetto a quelle rese disponibili attualmente da tecnologie simili (per esempio, i lettori di codici a barre). La pervasività di queste tecnologie, la quantità di informazioni che una WSN, dispiegata anche all'interno di un ambiente domestico, può raccogliere solleva complesse problematiche connesse anche con la tutela dei dati personali. In questo settore si stanno muovendo i primi passi e ci si attendono sviluppi futuri interessanti.

Bibliografia

[1] AA.VV.: *RFID tra presente e futuro*. Collana Quaderni AIP, Politecnico di Milano _ Dipartimento di Ingegneria Gestionale, 2005. http://www.osservatori.net.

[2] Borodulkin L., Ivanov B., Zhelondz O., Ruser H.: *Distributed smart sensor system for indoor climate monitoring*. In KONNEX Scientific Conference, ott. 2002.

[3] Bai H., Atiquzzaman M., Lilja David: *Wireless sensor network for aircraft health monitoring*. Proceedings of the First International Conference on Broadband Networks, IEEE Computer Society, 2004, p. 748-750.

[4] Bolchini C., Schreiber F. A., Tanca L.: A context-aware methodology for very small data base design. *SIGMOD Rec.*, Vol. 33, n. 1, 2004, p. 71-76.

[5] Boss R.: *Library RFID technology*. Library Technology Reports, nov.-dic. 2003.

[6] Brennan S. M., Mielke A. M., Torney D. C.: Radiation detection with distributed sensor networks. *IEEE Computer*, Vol. 37, n. 8, ago. 2004, p. 57-59.

[7] Brewin B.: *Delta has success in RFID baggage tag test*. Computerworld, 18 dic. 2003.

[8] Burrell J., Brooke T., Beckwith R.: Vineyard computing: Sensor networks in agricultural production. IEEE Pervasive Computing, Vol. 3, n. 1, gen.-mar. 2004, p. 38-45.

[9] Culler D., Estrin D., Srivastava M.: Guest editors' introduction: Overview of sensor networks. *Computer*, Vol. 37, n. 8, 2004, p. 41-49.

[10] Dulman S., Havinga P.: *Operating system fundamentals for the eyes distributed sensor network*. In: Proceedings of the 3-rd PROGRESS Workshop on Embedded Systems, Utrecth, The Netherlands, ott. 2002. http://wwwhome.cs.utwente.nl/~dulman/docs/paper_progress.pdf

[11] Dunkels A., Gronvall B., Voigt T.: *Contiki - a lightweight and flexible operating system for tiny networked sensors*. In LCN '04: Proceedings of the 29-th Annual IEEE International Conference on Local Computer Networks, Washington, DC, USA, 2004, p. 455-462. IEEE Computer Society.

[12] Manges W. W., e altri: Intelligent wireless sensors for industrial manufacturing. *Sensors Magazine*, Vol. 17, n. 4, apr. 2000.

[13] Mocci A., Montalto M.: *Sharp²anel: un pannello a messaggi variabili basato su tecnologia RFID*. Tesi di laurea, Politecnico di Milano, 2004.

[14] Feder B. J.: *Delta to invest in radio tags for luggage at airports*. The New York Times, 1 lug. 2004.

[15] Kalawsky R.S., Neild I.: *Sensing activities of daily living on a limited power budget*. In 2-nd International Workshop on Body Sensor Network, Imperial College, London, apr. 2005.

[16] Kaiser W.J., Pottie G.J., Srivastava M., Sukhatme G.S., Villasenor J., Estrin D.: *Networked infomechanical systems (nims) for ambient intelligence*. Relazione Tecnica n. 31, UCLA Center for Embedded Networked Sensing, dic. 2003.

[17] Kurata N., Spencer B. F. Jr., Ruiz-Sandoval M., Miyamoto Y., Sako Y.: *A study on building risk monitoring using wireless sensor network mica-mote*. Proceedings of the First International Conference on Structural Health Monitoring and Intelligent Infrastructure, nov. 2003.

[18] LaMarca A., Brunette W., Koizumi D., Lease M., Sigurdsson S. B., Sikorski K., Fox D., Borriello G.: *Making sensor networks practical with robots*. Proceedings of the First International Conference on Pervasive Computing, London, UK, Springer-Verlag, 2002, p. 152-166.

[19] Levis P., Culler D.: *Mate: a tiny virtual machine for sensor networks*. Proceedings of the 10-th international conference on Architectural support for programming languages and operating systems, New York, NY, USA, 2002, p. 85-95. ACM Press.

[20] Madden S.R., Franklin M.J., Hellerstein Joseph M., Hong Wei: TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database System*, Vol. 30, n. 1, 2005, p. 122-173.

[21] Malan D., Fulford-Jones T.R.F., Welsh M., Moulton S.: Codeblue: *An ad hoc sensor network infrastructure for emergency medical care*. Proceedings of the MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004), Boston, MA, giu. 2004. Articolo presentato anche a International Workshop on Wearable and Implantable Body Sensor Networks, London, April 2004.

[22] Maroti M., Simon G., Ledeczi A., Sztipanovits J.: Shooter localization in urban terrain. *IEEE Computer*, Vol. 37, n. 8, 2004, p. 60-61.

[23] McGinity M.: RFID: is this game of tag fair play? *Communications of the ACM*, Vol. 47, n. 1, 2004, p. 15-18.

[24] Nambi S., Nyalamadugu S., Wentworth S.M.: *Radio frequency identification sensors*. Proceedings of th 7-th World Multiconference on Systemics, Cybernetics and Informatics, lug. 2003.

[25] Rosencrance L.: *Las Vegas airport to implement RFID baggage-tag system*. Computerworld, nov. 2003.

[26] Shih E., Bychkovsky V., Curtis D., Guttag J.: *Continuous medical monitoring using wireless microsensors*. In: SenSys '04: Proceedings of the 2-nd international conference on Embedded networked sensor systems, New York, NY, USA. ACM Press, 2004, p. 310-310.

[27] Shnayder V., Chen B., Lorincz K., Fulford-Jones T.R.F., Welsh M.: *Sensor networks for medical care*. Relazione Tecnica n. TR-08-05, Harvard University, apr. 2005.

[28] Srivastava M.B., Muntz R.R., Potkonjak M.: *Smart kindergarten: sensor based wireless networks for smart developmental problem-solving enviroments*. Proceedings of the 7-th international conference on Mobile Computing and Networking, 2001, p. 132 - 138.

[29] Szewczyk R., Osterweil E., Polastre J., Hamilton M., Mainwaring A., Estrin D.: Habitat monitoring with sensor networks. *Communications of the ACM*, Vpl. 47, n. 6, 2004, p. 34-40.

[30] Todorovich P.: *RFID, come rivoluzionare l'identificazione*. Approfondimenti di Computer World Online, 9 lug. 2003.

[31] Want R.: Enabling ubiquitous sensing with RFID. *IEEE Computer*, Vol. 37, n. 4, 2004, p. 84-86.

[32] Xu N., Rangwala S., Chintalapudi K., Ganesan D., Broad A., Govindan R., Estrin D.: *A wireless sensor network for structural monitoring*. Proceedings of the ACM Conference on Embedded Networked Sensor Systems(Sensys04), nov. 2004.

[33] Ying Z., Ackerson L., Duff D., Eldershaw C., Yim M.: Tsam: a system of tracking and mapping in real environments. *Wireless Communications, IEEE*, Vol. 11, n. 6, dic. 2004.

[34] Zhao F., Guibas L.: *Wireless Sensor Networks*. Elsevier, Morgan Kauffman, 2004. ISBN: 1-55860-914-8.

Riferimenti a siti WEB

I siti web di seguito elencati sono stati visitati in data 29 luglio 2005.

[35] http://www.caifirenze.it/arva/

[36] http://www.rfidjournal.com/article/article-view/344/1/1/

[37] http://www.bluetooth.com/

[38] http://www.ece.osu.edu/~greenc/Tiny%20OS%20Info/

[39] http://www.citris.berkeley.edu/applications/disaster_response/smartbuildings.html

[40] http://www.xbow.com/

[41] http://www.autentica.it/

[42] http://firebug.sourceforge.net/

[43] http://www.ti.com/tiris/docs/solutions/epc/retail.shtml

[44] http://envisense.org/

[45] http://www.geospatial-online.com/geospatial-solutions/article/articleDetail.jsp?id=93546.

[46] http://www.cbe.berkeley.edu/research/xyz/FederspiellCEBO2002.pdf

[47] http://www.calit2.net/technology/features/sensorNetClass/AirVentCalibration.pdf.

[48] http://www.rfidjournal.com/article/view/309/1/1/

[49] http://www.indicod-ecr.it/

[50] http://www.eecs.berkeley.edu/~binetude/course/cs294-1/paper.pdf

[51] http://www.cs.cornell.edu/People/egs/magnetos/

[52] http://www.rfidjournal.com/article/article-view/369/1/1/

[53] http://www.metrogroup.de/servlet/PB/menu/1014631/index.html

[54] http://wins.rockwellscientific.com/

[55] http://www.semiconductors.philips.com/markets/identification/products/mifare/

[56] http://www.semiconductors.philips.com/markets/identification/articles/success/s46/

[57] http://www.ee.ucla.edu/~boulis/phd/SensorWare.html

[58] http://www.oracle.com/corporate/press/2005_mar/dolomiti.html

[59] http://www.intel.com/research/prohealth/cs-aging_in_place.htm

[60] http://www.tinyos.net/

[61] http://www.tinyos.net/nest/doc/tutorial/tossim-lesson.html

[62] http://www.mobilitytechnologies.com/ntdc/

[63] http://www.zigbee.org/

[64] http://informationweek.com/shared/printableArticle.jhtml?articleID=20600021

[65] http://www.rfidjournal.com/article/articleview/527/1/26/

[66] http://wi-fi.org/

[67] http://www.bartecautoid.com/applications-bartec.html

Mirko Cesarini ha conseguito il dottorato di ricerca in Ingegneria dell'Informazione presso il Politecnico di Milano nel 2005. I suoi interessi di ricerca riguardano: context aware programming, pervasive computing, sistemi di localizzazione e tecnologia RFID.
E-mail: mirko.cesarini@polimi.it

Carlo Ghezzi è Professore Ordinario di Ingegneria del Software, membro del Senato Accademico del Politecnico di Milano e responsabile scientifico dell'area di ricerca sull'Ingegneria del Software presso il CEFRIEL. È Fellow dell'ACM (Association for Computing Machinery) e Editor in Chief della rivista ACM Transactions on Software Engineering and Methodology. È autore di numerosi articoli scientifici e libri. Ha interessi di ricerca nell'ambito dei linguaggi di programmazione e dell'ingegneria del software, con particolare riguardo ai fondamenti teorici, metodologici e tecnologici dello sviluppo delle applicazioni distribuite su rete.
E-mail: carlo.ghezzi@polimi.it

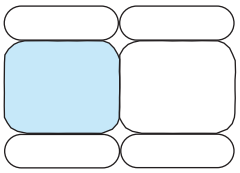
Fabio A. Schreiber è Professore Ordinario di Tecnologie dei Sistemi Informativi presso la Facoltà di Ingegneria del Politecnico di Milano. I suoi interessi di ricerca riguardano i Sistemi di Elaborazione Distribuiti e i Sistemi Informativi. Attualmente si occupa in particolare di problematiche relative ai Sistemi Informativi per applicazioni mobili, dipendenti dal contesto e comprendenti dispositivi utente molto piccoli (PDA, telefoni cellulari ecc.). È autore di numerosi articoli scientifici ed è associate editor di Data & Knowledge Engineering e di Decision Support Systems.
E-mail: schreibe@elet.polimi.it

Letizia Tanca è Professore Ordinario di Basi di Dati presso il Politecnico di Milano, e presidente del Consiglio di Corso di Studi di Ingegneria Informatica del Politecnico di Milano, campus Milano Leonardo. È autrice di diverse pubblicazioni internazionali sulle basi di dati e sulla teoria delle basi di dati, e del libro "Logic Programming and Databases", scritto con S. Ceri e G. Gottlob. Ha partecipato a vari progetti nazionali e internazionali.
I suoi interessi di ricerca riguardano tutta la teoria delle basi di dati, in particolare le basi di dati deduttive, attive e orientate agli oggetti, i linguaggi a grafi per basi di dati, la rappresentazione e l'interrogazione di informazione semistrutturata, le basi di dati per piccoli dispositivi.
E-mail: tanca@elet.polimi.it

LA SCIENZA DEL CAOS E IL COMPUTER

Giancarlo Storti Gajani

L'approccio riduzionista considera naturale descrivere con modelli ciò che si vede in natura. I limiti della matematica rendono impossibile racchiudere in equazioni il comportamento di entità, anche molto comuni, quali la forma delle nuvole. La dinamica nonlineare, impensabile senza i computer, dà nuove possibilità di affrontare questi problemi. L'articolo introduce alcuni aspetti della dinamica nonlineare, limitandosi ai sistemi in cui il tempo è una variabile continua.



1. INTRODUZIONE

Definire la "scienza del caos" in modo rigoroso e tale da non scontentare nessuno tra coloro che se ne occupano è ancora abbastanza difficile, trattandosi di una scienza giovane. Malgrado si possa senz'altro fare di meglio, possiamo iniziare con il dire che la scienza del caos si occupa in prima istanza di sfatare il mito secondo il quale determinismo sia uguale a determinabilità e di scoprire come sia più norma che accidente il fatto che, in molti fenomeni fisici, condizioni iniziali impercettibilmente differenti possano corrispondere a risultati finali enormemente diversi. La realtà è nonlineare e anche se si può dire molto su un fenomeno "facendo finta" che sia lineare (e molta ricerca scientifica fino ai tempi più recenti si è basata su questo assunto) si finirà inevitabilmente col perdere la percezione di aspetti della realtà comuni e sotto gli occhi di tutti. La forma delle nuvole, il lancio di un dado, la turbolenza in un fluido o la crescita di una pianta sono tutti fenomeni difficilmente spiegabili con modelli semplici. Fortunatamente, per vedere esempi di comportamento deterministico, ma difficilmente determinabile, non è necessario considerare situazioni così complesse. Fenomeni fisici apparentemente molto semplici e i cui modelli matematici sono costituiti da pochissime equazioni possono avere comportamenti anche estremamente complicati. In questo articolo ci siamo limitati a sistemi e modelli in cui il tempo può essere considerato una variabile continua. Ci sono molti casi in cui è opportuno considerare il tempo come una variabile discontinua, generalmente definita solo su una successione di istanti equispaziati tra loro ed è il caso dei sistemi a tempo discreto. Questi ultimi hanno caratteristiche molto interessanti, ma possono portare ancora più facilmente a comportamenti complessi.

2. MODELLI DELLA REALTÀ E SISTEMI DINAMICI

"*...Questo grandissimo libro che continuamente ci sta aperto innanzi agli occhi (io dico l'universo), non si può intendere se prima non s'impara a intender la lingua, e conoscer*

i caratteri ne' quali è scritto. Egli è scritto in lingua matematica, e i caratteri son triangoli, cerchi, ed altre figure geometriche, senza i quali è impossibile a intendere umanamente parola; senza questi è un aggirarsi vanamente per un oscuro laberinto".

Galileo Galilei, Opere VI

Molti, e non a torto, considerano Galileo il padre del metodo scientifico occidentale: il "grande libro" di cui parla è la natura e l'idea di poterne scrivere le leggi in termini rigorosi ha da sempre caratterizzato tutta la ricerca scientifica. L'analisi sperimentale di un fenomeno fisico ne permette la riduzione ad un modello matematico, ulteriori esperimenti permettono di convalidare e perfezionare questo modello che, una volta ben definito, può essere usato per dedurre una teoria che spieghi altri fenomeni fisici, possibilmente più generali e anche diversi da quelli dai quali si è partiti. Tutto ciò che noi oggi chiamiamo scienza si basa su questo principio metodologico, e tutto ciò che chiamiamo tecnologia deve tantissimo, anzi, tutto, al rigore del metodo scientifico galileiano.
La realtà evolve nel tempo, farne dei modelli implica nella maggior parte dei casi riassumere in un certo numero di equazioni, le cui variabili misurino in qualche modo lo "stato" in cui si trova l'entità che si sta studiando, le leggi che governano l'evoluzione temporale del fenomeno. Le soluzioni di queste equazioni rappresentano quindi il comportamento del modello e permettono, se le equazioni sono corrette, di tentare di prevederne l'evoluzione. In generale non si usano tutte le possibili variabili che rappresentano un fenomeno: se per esempio volessimo studiare il movimento dei pianeti nel sistema solare potremmo, in prima approssimazione, rappresentare il Sole e i pianeti come punti dotati di massa e la loro posizione con le coordinate di ognuno di questi punti nello spazio tridimensionale rispetto ad un qualche sistema di riferimento. Le equazioni che descrivono il moto dei pianeti, si riducono allora alle ben note leggi della dinamica ed alla legge di gravitazione di Newton; se si vuole maggiore precisione si possono aggiungere le correzioni relativistiche, ma a nessuno verrà mai in mente di complicare il modello includendo, per esempio, lo stato della atmosfera o le condizioni politiche delle nazioni terrestri. Il modello migliore risponde quindi ad un principio di economia: deve includere *tutte e sole* le informazioni che permettono di risolvere il problema che ci interessa.
Consideriamo ora un semplice sistema che evolve nel tempo, essendo il sistema puramente ideale è di fatto già un modello. Prendiamo una pallina, immaginiamola dotata di peso ed appoggiamola su una superficie, non necessariamente piana. Se immaginiamo che tra pallina e superficie ci sia una qualche forma di attrito e se pensiamo ad una superficie a forma di scodella tutti sappiamo che la pallina finirà per stabilirsi sul fondo della scodella. Il fondo della scodella è il punto di equilibrio di questo semplice sistema, e, visto che perturbando leggermente la pallina questa tornerà comunque sul fondo, il punto di equilibrio è stabile (Figura 1 **A**). Modifichiamo la scodella, cambiandone la curvatura fino a trasformarla in un piano. Ora la pallina resta ferma in qualsiasi punto venga appoggiata, e piccole perturbazioni porteranno la pallina in posizioni differenti: tutti i punti del piano sono di equilibrio indifferente (Figura 1 **B**). Se infine trasformiamo la scodella in una collina (invertendone la curvatura) la pallina sfuggirà da dove la poniamo, a meno di non posizionarla, con estrema precisione, sul punto più alto. La cima della collina è ancora un punto di equilibrio, ma in questo caso instabile (Figura 1 **C**) e qualsiasi perturbazione, con il nostro semplice modello, comporta per la pallina un viaggio verso l'infinito.

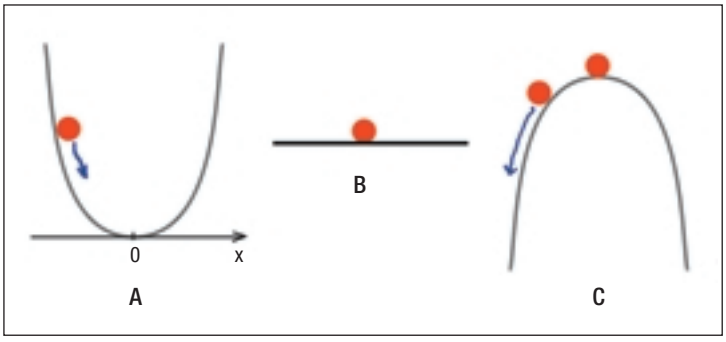


FIGURA 1

Il sistema "pallina + scodella" per diversi valori di curvatura della scodella: **A** *Curvatura positiva;* **B** *Curvatura nulla;* **C** *Curvatura negativa*

Come si può notare, malgrado le tre situazioni siano descritte dallo stesso modello di base, si hanno comportamenti decisamente diversi tra loro, cambiando solo un *parametro* del modello: la curvatura della superficie. È cioè possibile scrivere una *unica* equazione con un parametro che rappresenta la curvatura della superficie che descrive i tre diversi comportamenti di figura 1. I cambiamenti strutturali nella dinamica di un sistema al variare di uno o più parametri sono chiamati *biforcazioni*, lo studio delle biforcazioni di un sistema è fondamentale per capirne il funzionamento. Sistemi anche più complicati possono avere soluzioni stabili (a volte dette genericamente soluzioni di *regime*, dopo l'eventuale transitorio iniziale) oppure instabili, solo che in molti casi è molto più difficile classificarli.

Il caso instabile di figura 1 **C** è poi un po' inquietante, è infatti esperienza comune che i fenomeni fisici con cui abbiamo a che fare sono sempre in qualche modo stabili, almeno a lungo termine o su scala sufficientemente grande: persino un fenomeno violento come una esplosione prima o poi giunge in uno stato di quiete, e la materia espulsa da una supernova formerà plausibilmente nebulose ed eventualmente nuove stelle.

Accettare questo dato dell'esperienza corrisponde ad affermare che qualsiasi fenomeno naturale, localmente instabile, può essere descritto da un modello matematico *lineare* solo in un piccolo intorno delle sue condizioni iniziali, perché se un sistema lineare è instabile in un punto allora è necessariamente instabile in modo *globale*. Modelli nonlineari dotati di comportamento instabile in una regione circoscritta, ma stabili rispetto ad una regione più grande, sono la norma, anche nelle applicazioni pratiche. Qualsiasi oscillatore stabile (come per esempio gli oscillatori necessari al funzionamento dei calcolatori, degli orologi e di molti altri sistemi elettronici) funziona proprio grazie al fatto di essere instabile in un intorno dell'origine dello spazio di stato (in questo modo si "accende"), ma stabile in grande (in questo modo le traiettorie si stabilizzano su una ampiezza costante). La nonlinearità è dunque la norma, e non un'eccezione.

Sempre pensando al sistema pallina + scodella, rappresentiamo su un piano le variabili che rappresentano lo stato del nostro sistema e tracciamo una curva (detta *traiettoria*) che descrive l'evoluzione del nostro modello istante per istante. Le variabili che possiamo considerare sono velocità e posizione della pallina, se consideriamo il primo dei tre casi visti sopra (quello di Figura 1 **A**) abbiamo, per una generica posizione iniziale, la traiettoria mostrata nella figura 2 **A**. Se ora variamo un altro parametro del nostro modello, il coefficiente d'attrito, otteniamo, aumentandolo, le traiettorie di figura 2 **B**, ma, riducendo a zero l'attrito, le traiettorie in figura 2 **C**. Si noti come in quest'ultimo caso l'energia iniziale del sistema resti costante e la pallina continui indefinitamente ad oscillare all'interno della scodella.

Per rappresentare le traiettorie abbiamo usato un piano perché abbiamo solo due variabili di stato, ovviamente, se avessimo tre variabili di stato, dovremmo usare una rappresentazione tridimensionale e per dimensioni superiori saremmo costretti, se proprio volessimo avere una percezione geometrica visiva di quanto succede, a fare sezioni o proiezioni di queste traiettorie su sottospazi di dimensione due o tre.

Quest'ultima osservazione è molto meno banale di quanto non si possa pensare: se stiamo infatti considerando un fenomeno fisico deterministico pretendiamo senz'altro che qualsiasi suo modello sia altrettanto deterministico, e quindi che ad uno stato del modello corrisponda una ed una sola possibile evoluzione. In termini geometrici ciò implica che le traiettorie di un sistema non possono in alcun modo incrociarsi perché, se si incrociassero, in quel punto il sistema potrebbe evolvere in due modi distinti. Le conseguenze di questo fatto sono decisamente straordinarie: un modello con due sole variabili di stato potrà avere traiettorie che convergono su uno o più punti di equilibrio stabili, che sfuggono da un equilibrio instabile, oppure dei cicli periodici (traiettorie chiuse percorse all'infinito). Le traiettorie che nascono all'interno di un ciclo non potranno quindi fare altro che restare confinate all'interno del ciclo, al massimo adagiandosi su di esso, e quelle all'esterno

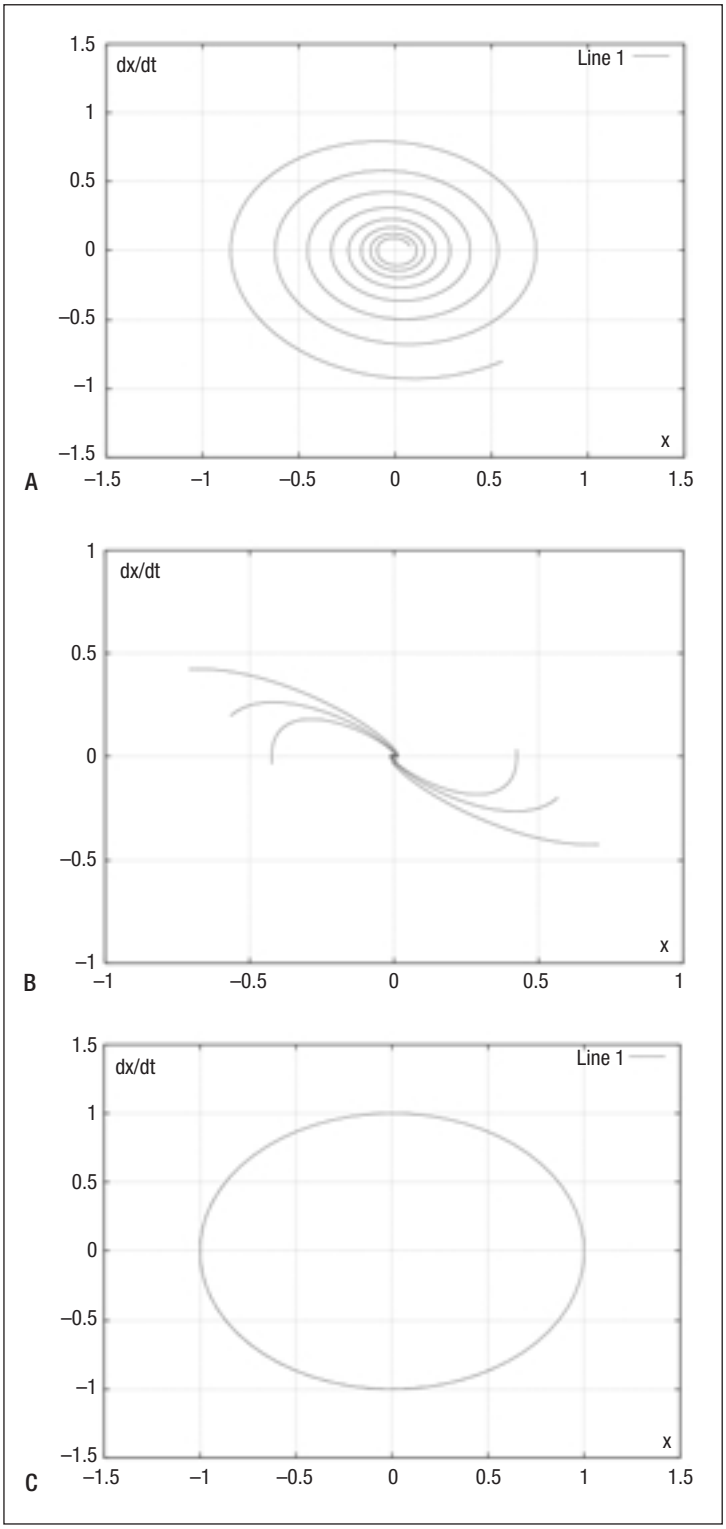


FIGURA 2

Traiettorie del sistema "pallina + scodella", lo stato iniziale è indicato con un puntino rosso; **A** *attrito "normale";* **B** *attrito "elevato";* **C** *attrito nullo*

resteranno per sempre all'esterno (uno dei pionieri nella analisi topologica delle soluzioni delle equazioni differenziali nonlineari è stato Henri Poincaré [14]).

Gli oscillatori elettronici sono sistemi con quest'ultimo tipo di traiettoria e sono presenti intorno a noi in grande numero in qualsiasi orologio, telefono cellulare, calcolatore ecc.. In figura 3 è mostrato il *ritratto di fase* dell'oscillatore di Van Der Pol, cioè alcune sue traiettorie tracciate nel piano definito dalle sue variabili di stato, la tensione sul condensatore e la corrente sull'induttore. Questo oscillatore è uno dei più semplici oscillatori elettronici, composto solo da un condensatore, un induttore ed un resistore nonlineare facilmente sintetizzabile con pochi transistor.

Talvolta, facendo misure su un fenomeno, magari finalizzate a validarne un modello, si trovano traiettorie che si incrociano: o il fenomeno è non deterministico, oppure ci si è dimenticati di una o più variabili di stato. Traiettorie che apparentemente si incrociano in due dimensioni possono infatti non incrociarsi se di dimensioni se ne hanno tre (o più), ed è immediatamente evidente che in tre dimensioni si possono avere situazioni anche molto più complicate di quelle che si hanno nelle due dimensioni.

Un semplice sistema meccanico in grado di mostrare comportamenti anche molto complessi è la cosiddetta "ruota idraulica di Lorenz" (lo schema in Figura 4). Si tratta di una ruota a pale con perdite, di notevole interesse storico, in quanto descritta da un sistema di equazioni equivalenti al modello di Lorenz, il sistema di equazioni da cui è nato il concetto di caos deterministico, e facilmente costruibile in casa (ne esistono realizzazioni perfettamente funzionanti costruite utilizzando solo bicchierini di plastica bucati).

Le variabili di stato sono in questo caso la posizione della ruota (misurata come angolo rispetto ad un certo riferimento), la velocità e l'accelerazione angolare; i principali parametri caratteristici sono la portata di fluido in ingresso e l'attrito del perno della ruota. Si nota anche solo qualitativamente (in altre parole, ad occhio) che ci troviamo in presenza di una situazione più complicata di quelle viste sopra. Le traiettorie di questo sistema, per tutto un insieme di valori dei parametri, non sono né instabili né periodiche, ma si adagia-

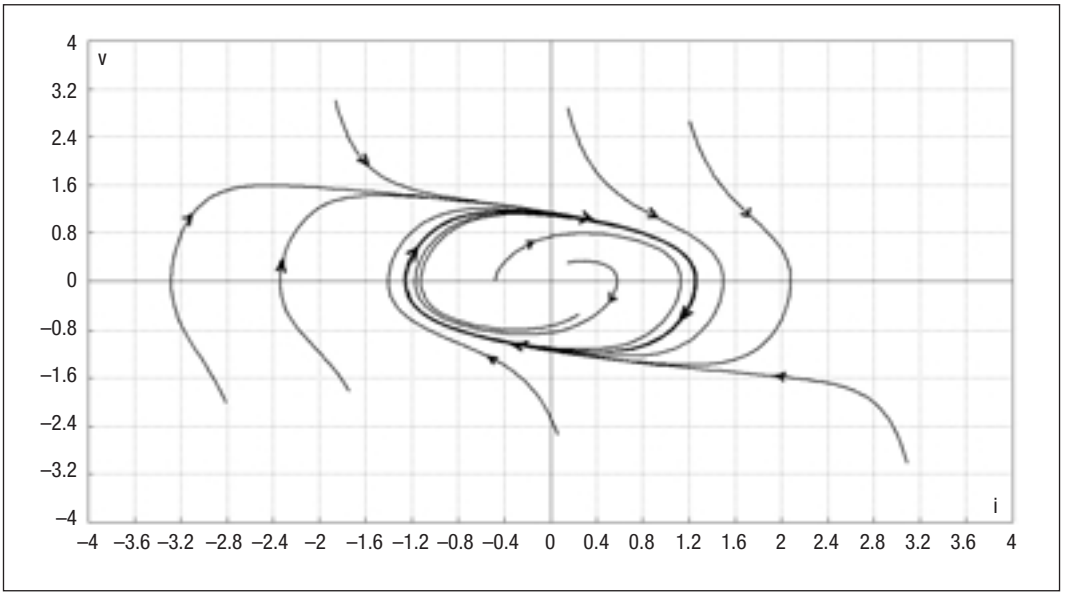
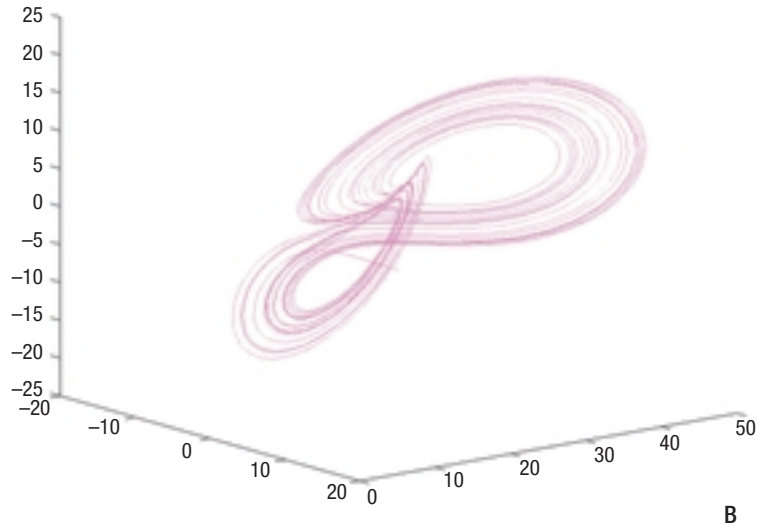


FIGURA 3
Ritratto di fase dell'oscillatore di Van Der Pol

A



B

FIGURA 4
A *La ruota idraulica di Lorenz;* **B** *Una sua traiettoria caotica*

no su un insieme di punti dello spazio di stato dalla geometria molto complessa. Questo insieme è chiamato *attrattore strano* in quanto qualsiasi traiettoria finirà per entrare nell'insieme, come nei normali attrattori, quali punti di equilibrio stabile o cicli periodici stabili, ma è dotato di una geometria decisamente "complicata" (e quindi "strana"). Nessun punto di un attrattore strano è visitato più di una volta (non siamo quindi in un caso periodico), ma la traiettoria non sfugge dall'insieme di punti (e quindi non è instabile).
Le equazioni di Lorenz sono state originariamente scritte da Edward Norton Lorenz durante la ricerca di modelli meteorologici semplificati. Secondo quanto racconta lo stesso Lorenz [10, 11, 12] ciò che lo sorprese non fu tanto la complessità geometrica delle traiet-

torie del suo sistema, quanto la forte dipendenza dalle condizioni iniziali da lui scoperta quasi per caso. Il suo modello infatti "girava" su un calcolatore ENIAC (capace di eseguire ben 60 moltiplicazioni al secondo!) e, nel trascrivere manualmente le condizioni iniziali per il suo sistema tra due esperimenti numerici successivi, aveva troncato i numeri alla quarta cifra decimale. Le traiettorie del secondo esperimento, dopo un primo tratto quasi perfettamente in comune con il primo, proseguivano in modo diverso, ma sempre restando nella stessa "zona di funzionamento", restando, cioè, confinate nell'attrattore.
Ovviamente un sistema nonlineare può avere più di un attrattore (strano e non); in questo caso, partendo da diverse condizioni iniziali, è possibile finire su attrattori differenti. L'insieme di punti dello spazio di stato che, presi come condizioni iniziali, portano allo stesso attrattore viene chiamato *bacino d'attrazione* dell'attrattore (e i bacini di attrazione di alcuni "semplici" sistemi dinamici a tempo discreto costituiscono alcuni dei "frattali" più conosciuti, come, per esempio, il celebre insieme di Mandelbrot).

3. IL MODELLO DI LORENZ E IL RUOLO DEI CALCOLATORI ELETTRONICI

L'equazione di Lorenz rappresenta uno tra i più semplici, e senz'altro il più conosciuto, modello nel quale sono presenti attrattori caotici. La sua importanza è anche storica: proprio dalla simulazione numerica di questa equazione Lorenz si accorse della forte dipendenza di questo genere di sistemi dalle condizioni iniziali, battezzata dallo stesso Lorenz con il nome "*effetto farfalla*".
L'equazione che descrive l'evoluzione temporale del modello di Lorenz è un sistema differenziale di ordine tre, le variabili di stato sono x_1, x_2 e x_3 e le derivate rispetto al tempo sono indicate con un puntino sopra la rispettiva variabile:

$$\dot{x}_1 = ad\,(x_2 - x_1)$$
$$\dot{x}_2 = x_2 + d\,(bx_1 - x_2 - x_3\,x_1)$$
$$\dot{x}_3 = x_3 + d\,(x_1\,x_2 - cx_3)$$

I parametri sono a,b,c e d, ognuno dei quali rappresenta, nel modello di Lorenz, una qualche proprietà fisica del suo modello fluidodinamico. Le soluzioni di questa equazione dipenderanno, oltre che dai valori dei parametri, dalla condizione iniziale del sistema.
Come per la maggior parte dei casi nonlineari, non è possibile scrivere la soluzione del sistema di Lorenz in forma chiusa, ma se ne possono comunque studiare le proprietà analizzando il sistema di partenza e risolvendo numericamente le equazioni per mezzo di un calcolatore. Come già detto, l'effetto farfalla è stato scoperto fortuitamente grazie alla trascrizione manuale ed approssimata delle condizioni iniziali tra due successivi esperimenti su calcolatore elettronico. In realtà i metodi numerici per la soluzione delle equazioni differenziali esistono da molto prima dei primi calcolatori elettronici e la scoperta fondamentale di Lorenz avrebbe potuto avvenire anche senza la necessità di strumenti di calcolo, ma il tempo e lo sforzo necessari sarebbero stati immensi, anche ammettendo di avere a disposizione una macchina calcolatrice non programmabile per le operazioni aritmetiche di base. Anche se l'ingegno di numerosi studiosi "pre-computer" hanno permesso la messa a punto di strumenti teorici molto raffinati per cercare di far luce nel comportamento dei modelli nonlineari, e Poincaré si era spinto a ipotizzare la possibile esistenza di quelli che poi furono chiamati "attrattori strani", è difficile pensare allo sviluppo di questa scienza e, a maggior ragione, delle sue applicazioni, senza l'ausilio dei moderni strumenti di calcolo.
Per un insieme (abbastanza vasto) dei valori dei parametri le soluzioni del modello di Lorenz finiscono tutte, cioè per qualsiasi condizione iniziale e dopo un eventuale transitorio iniziale, su un attrattore strano (la forma di questo attrattore cambia in modo anche molto significativo al variare dei parametri). Una rappresentazione molto elegante di uno di questi attrattori, adatta a metterne in luce la complessità geometrica, si ottiene disegnandone una mappa di densità (L'attrattore in Figura 5 si ottiene per i valori dei parametri: a = 7.906, b = 8.79, c = 3.379 e d = 0.079, ed è equivalente a quello mostrato in Figura 4). Per meglio apprezzarne la complessità e la forma lo stes-

so attrattore visualizzato da diversi punti di vista è mostrato nella figura 6. Ognuna di queste figure si ottiene calcolando numericamente l'equazione di Lorenz in alcuni milioni di punti, pochi secondi sul PC che sto usando per scrivere questo testo, parecchie ore sull'ENIAC che usava Lorenz, e un tempo difficilmente quantificabile per chi volesse provare con carta e penna.

Ogni singola traiettoria del sistema di Lorenz concorre alla formazione della mappa di densità. È interessante notare che seguire una singola traiettoria per lungo tempo o tante traiettorie diverse (definite da diverse condizioni iniziali scelte a caso) da luogo alla stessa mappa. Ogni traiettoria arriverà infatti prima o poi arbitrariamente vicina a qualsiasi punto dell'attrattore, ma senza ovviamente intersecare altre traiettorie.

Gli attrattori strani come quelli mostrati in figura 5 e figura 6 sono classici esempi di *oggetti frattali*. Nel nostro caso la dimensione geometrica di questi attrattori è compresa tra 2 e 3 (per un sistema dinamico con tre variabili di stato), è cioè un numero *non intero*. La difficoltà nel comprendere che un oggetto possa avere una dimensione non intera deriva dalla nostra abitudine a trattare con le entità della geometria Euclidea. La quantità di informazione necessaria per stabilire la posizione di un punto su una qualsiasi figura della geometria classica è infatti direttamente correlata al numero di coordinate utilizzate, eventualemente definite rispetto ad un sistema di riferimento "locale". Ma quanta "informazione" è necessaria per descrivere la posizione di un punto appartenente a un attrattore strano? Si può dare una risposta a questa domanda estendendo la misura classica di dimensione con la definizione data da Hausdorf [3]: si ricopre l'oggetto di cui si vuole misurare la dimensione con n sfere di raggio r e si calcola il rapporto logaritmico tra il numero di sfere necessarie ed il loro raggio facendo tendere a zero il raggio stesso:

$$D_{\mathrm{H}} = \lim_{\mathrm{r}\to 0} \frac{\log n\,(r)}{\log {}^{1}\!/_{r}}$$

Questa nuova e più generale definizione di dimensione coincide, ovviamente, con la misura classica per gli "oggetti geometrici" semplici. Il nome *frattale* viene usato per designare gli insiemi di punti la cui dimensione secondo Hausdorf è un numero non intero. I frattali più conosciuti sono quelli che descrivono i bacini di attrazione di sistemi dinamici a tempo discreto, come il famoso "insieme di Mandelbrot", ma si possono generare frattali anche in altri modi, per esempio iterando all'infinito delle trasformazioni geometriche su oggetti inizialmente anche molto semplici. Qualsiasi sia il sistema scelto, il numero di operazioni necessarie per ottenere frattali anche molto semplici è comunque proibitivo senza strumenti di calcolo automatico.

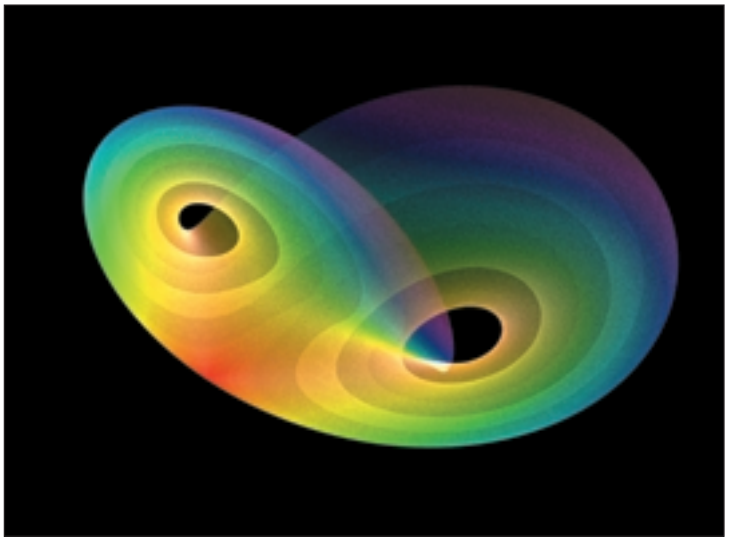


FIGURA 5

Mappa di densità di un attrattore del modello di Lorenz. I colori corrispondono a diverse velocità delle traiettorie

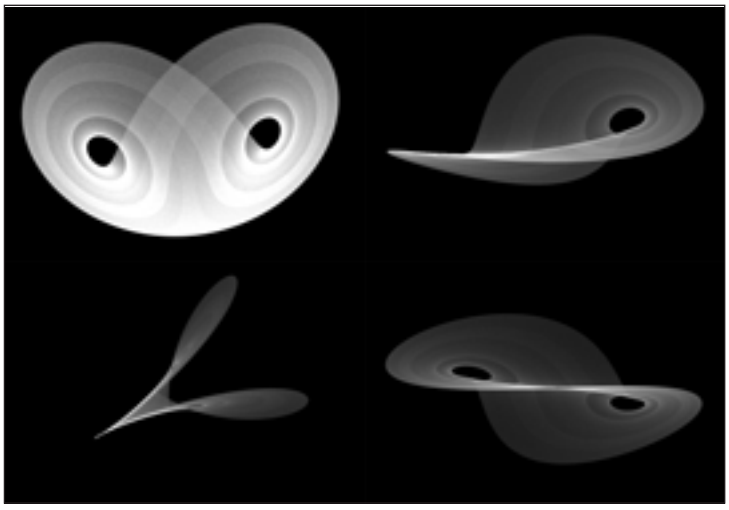


FIGURA 6

Un attrattore strano del modello di Lorenz da diversi punti di vista

4. SENSITIVITÀ ALLE CONDIZIONI INIZIALI E DETERMINABILITÀ

Se un sistema è deterministico ad ogni condizione iniziale corrisponde una singola traiettoria e a condizioni iniziali diverse corrispondono traiettorie diverse. Secoli di scienza basata su modelli lineari o linearizzati ci hanno abituato a pensare che solo la presenza di eventi casuali, cioè non deterministici, e, soprattutto, macroscopici può giustificare risultati molto diversi a partire da condizioni iniziali simili tra loro. Viene spontaneo pensare che qualsiasi piccola perturbazione dello stato iniziale di un sistema corrisponda necessariamente a variazioni altrettanto piccole nel comportamento finale. Come già visto i primi risultati di Lorenz sconfessarono questa visione del mondo: non è in generale vero che in un sistema deterministico condizioni iniziali vicine tra loro corrispondono sempre a traiettorie simili e vicine, dove "vicino" è definito secondo una qualche metrica coerente al sistema che si sta considerando. Anche piccolissime variazioni nelle condizioni iniziali possono portare due identici sistemi caotici, posti a confronto, in stati totalmente diversi a un tempo determinato. Un esempio di questo comportamento si può vedere nella figura 7, dove l'evoluzione del modello di Lorenz è confrontata, per una delle variabili di stato, a partire da due condizioni iniziali molto simili (la differenza è di 0.0001 su una sola variable di stato, $x_3(0)$, a partire dalle condizioni iniziali nominali $x_1(0) = 20$, $x_2(0) = 5$ e $x_3(0) = -5$).

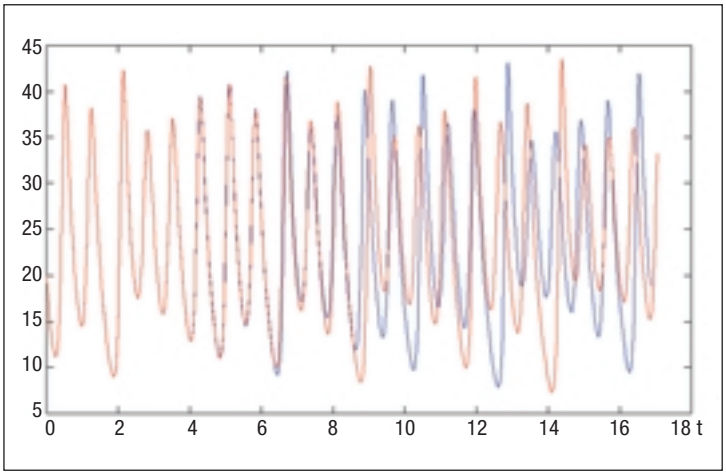


FIGURA 7

Comportamento di una delle variabili di stato del modello di Lorenz a partire da due condizioni iniziali "leggermente" differenti

Dopo un breve periodo in cui le due situazioni sono molto simili le traiettorie si allontanano, pur restando, ovviamente, confinate all'interno dell'attrattore.
Questa proprietà rappresenta per molti la definizione stessa di sistema caotico.
Il problema è allora molto serio: conoscere con precisione *infinita* lo stato iniziale di un sistema è impossibile (richiede, per lo meno, energia infinita), e quindi è impossibile determinare con sicurezza come si comporterà un sistema deterministico, ma caotico.
Essendo i sistemi caotici più diffusi di quanto non si sospettasse in passato, al punto di poter dire che sono più la norma, che un'eccezione, i concetti stessi di "*casuale*" e "*deterministico, ma non determinabile*" sembrano sempre più sul punto di confondersi.

5. APPLICAZIONI

L'abitudine a credere che i sistemi semplici possano avere solo comportamenti semplici e che solo con sistemi molto complessi (eventualmente composti dalla interconnessione di molti sistemi semplici) si possano avere comportamenti complessi, ha fatto sì che molti, tuttora, credano che i sistemi caotici siano essenzialmente curiosità accademiche.
In realtà sono molto più diffusi di quanto non si creda, anzi, aver compreso che molti fenomeni naturali sono riconducibili ad una descrizione relativamente semplice, ma fortemente non lineare, ha permesso di risolvere in modo originale alcuni problemi. Le applicazioni della dinamica non lineare in generale, e della teoria del caos in particolare, sono infatti sempre più numerose. Alcune possibili applicazioni sono ancora in uno stato infantile, per esempio quelle relative ai sistemi economici, per la previsione dei mercati finanziari, o ai sistemi sociali, per la previsione delle scelte politiche; altre hanno invece già prodotto risultati interessanti.
Per tentare di dare una minima tassonomia agli usi della dinamica nonlineare e della teoria del caos considereremo tre classi principali di applicazioni: prima di tutto si farà qualche cenno ad alcune applicazioni della dinamica non lineare dove la presenza di comportamenti caotici è solo una eventua-

lità, ma non rappresenta una condizione indispensabile per l'applicazione stessa. Una seconda classe considera poi i casi dove la presenza di situazioni caotiche è intrinseca all'applicazione, ma dove è necessario controllare il sistema, per eliminare gli effetti del caos o per scegliere uno specifico attrattore tra i tanti possibili. L'ultima classe rappresenta le applicazioni più innovative: quelle che non sarebbero pensabili senza la presenza o, addirittura, la generazione di segnali caotici da parte delll'applicazione stessa.

5.1. Applicazioni "tradizionali" della dinamica non lineare

Lo studio di molti problemi classici della fisica, anche di alcuni tra i più semplici problemi di meccanica, richiedono la soluzione di equazioni differenziali non lineari. È quindi spontaneo affermare che i campi applicativi della dinamica non lineare sono *tutti*. O almeno tutti quelli in cui è presente un qualche tipo di dinamica. Ovviamente non tutti i problemi di dinamica non lineare implicano la presenza di soluzioni caotiche. Ma, come tutti sanno, anche il classico problema dei tre corpi può avere soluzioni molto complicate (e i suoi punti di equilibrio sono i ben noti "punti di Lagrange"). Il problema è che la maggior parte delle equazioni non si possono risolvere in forma chiusa, e l'unica possibilità, per chi non disponeva di strumenti di calcolo, consisteva nell'inferire alcune proprietà della soluzione a partire dalle equazioni di partenza o semplificare notevolmente il problema linearizzando. L'avvento dei calcolatori, e quindi l'analisi numerica di sistemi di equazioni di dimensioni rilevanti, ha permesso di ampliare notevolmente la ricerca e le applicazioni già esistenti e di trovarne di nuove.

Tra i campi applicativi della dinamica non lineare in cui è possibile la presenza di soluzioni caotiche, forse i più conosciuti sono quelli tipici della fluidodinamica. I moti turbolenti nei fluidi e nei gas possono in molti casi essere riassunti e descritti in modo "macroscopico" in modo da non dover trattare nello specifico la dinamica "microscopica" sottesa. Queste semplificazioni hanno permesso, anche in epoca "pre-calcolatore" la soluzione di molti problemi importanti di fluidodinamica, descrivendo, per esempio, gli effetti di un fluido in moto turbolento in un condotto usando solo pochi parametri riassuntivi. Una descrizione più precisa di questi fenomeni è invece ora possibile e, grazie ai calcolatori, si possono per esempio progettare profili alari, turbine ed eliche, molto più efficienti.

Sempre pensando alla fluidodinamica, si è già visto come le stesse equazioni di Lorenz, oltre ad un immenso valore teorico, avevano l'intento di risolvere dei problemi di meteorologia. Una delle prime applicazioni dei grandi calcolatori è stata, ed è tuttora, proprio la soluzione di enormi sistemi di equazioni che cercano di descrivere e prevedere il tempo. Proprio a causa della rilevante sensitività allo stato iniziale le previsioni del tempo sono tanto più affidabili quanto più sono a breve termine (Figura 8).

Coloro che si occupano di studiare le vibrazioni nelle macchine e negli edifici hanno da tempo iniziato a fare uso degli strumenti della dinamica nonlineare e, ancora una volta, la possibilità di simulare strutture anche molto complesse in tempi ragionevoli ha dato nuovo impulso ad applicazioni di notevole importanza economica e umana.

Tra le altre applicazioni della dinamica non lineare sono attualmente molto importanti quelle che riguardano l'analisi dei sistemi ecologici e dell'ambiente. Anche in questi campi la ricerca è molto attiva, e c'è sicuramente molto spazio per nuove scoperte.

FIGURA 8

L'uragano Epsilon (3 dic 2005, courtesy of Nasa/Goddard space flight center)

5.2. Evitare gli attrattori "troppo strani", il controllo del caos

Molti fenomeni sono necessariamente caotici e può nascere, in alcuni casi, l'esigenza di scegliere con cura in quale attrattore si vuole che il sistema funzioni, oppure, in altri casi, si ha l'esigenza di eliminare, del tutto o in parte, i comportamenti caotici.

Malgrado una notevole regolarità di fondo, il funzionamento del cuore è, secondo molti ricercatori, essenzialmente caotico. Osservando il tracciato di un elettrocardiogramma si nota come ogni battito sia diverso dagli altri, e un cuore in buona salute è in molti casi più caotico di uno malato [4, 5, 6]. Durante un episodio di fibrillazione atriale o ventricolare, il funzionamento del cuore è ancora caotico, ma le sue variabili di stato si portano su un attrattore strano la cui dinamica è inefficace a pompare il sangue. I defibrillatori, resi famosi, spesso in modo ridicolo, dai programmi televisivi di "fiction" medica, sono macchine basate sull'idea di dare un forte shock al sistema di sincronizzazione cardiaca utilizzando una scarica elettrica di grande intensità. Si spera in questo modo di riportare il "sistema cuore" verso l'attrattore giusto. Il meccanismo spesso funziona, anche se non è privo di pericoli.

In alternativa si può pensare di sfruttare proprio la caratteristica tipica dei sistemi caotici di avere una forte dipendenza dalle condizioni iniziali. Una forte dipendenza dalle condizioni iniziali indica che piccole perturbazioni possono variare in modo anche notevole il funzionamento di un sistema. Questa specifica caratteristica è stata utilizzata per la messa a punto di un "pace-maker" basato sui principi del controllo caotico. Piccole perturbazioni, di entità controllata e applicate in istanti determinati con precisione, riescono a riportare il cuore sull'attrattore strano giusto. Per ora sono stati effettuati numerosi esperimenti su animali che hanno dimostrato la fattibilità e la correttezza di questo approccio.

Un'altra applicazione recente, sempre in campo biomedico, riguarda la previsione ed eventualmente la prevenzione degli episodi epilettici. Dagli studi compiuti su pazienti umani si è scoperto che è possibile prevedere episodi di epilessia anche con un notevole anticipo dall'analisi di alcune proprietà dei segnali di un EEG. La dimensione frattale del segnale a rischio di epilessia subisce infatti una *diminuzione* qualche tempo prima dell'insorgere dell'episodio epilettico (ancora una volta il comportamento normale pare essere *più* caotico di quello patologico). È ancora in fase di studio la possibilità di utilizzare dei "micro-elettroshock" localizzati per prevenire le fasi epilettiche acute.

5.3. Usare gli attrattori anche molto strani

Produrre caos è facile, sia con i calcolatori che con i circuiti elettronici. Il più semplice e studiato circuito dotato di attrattori strani è il *circuito di Chua*, utilizzato per la sua semplicità e il suo bassissimo costo in molte applicazioni [9]. Consideriamo ora applicazioni dove è necessario produrre attrattori strani, bacini di attrazione frattali o, uno o più segnali caotici.

L'uso forse più diffuso e più maturo commercialmente della teoria del caos, riguarda la generazione di oggetti geometrici frattali, in particolare per la generazione di paesaggi virtuali, cioè per la grafica e per le animazioni 3D. Moltissimi effetti speciali cinematografici di sintesi, soprattutto quelli che generano paesaggi virtuali, sono infatti basati sulla generazione di oggetti frattali. Qualsiasi montagna, albero, costa o roccia ha un aspetto molto più realistico se generata in modo frattale, piuttosto che utilizzando primitive geometriche classiche (come si faceva agli inizi della grafica in 3D). Nessuno dei film di animazione più recenti sarebbe possibile senza i *landscape generator* attualmente presenti anche nei prodotti di grafica commerciali e *open-source* più conosciuti ed alla portata di tutti.

Un'altra interessante caratteristica dei sistemi caotici è la loro capacità di sincronizzarsi: se si costruiscono due sistemi caotici identici (o anche solo molto simili) e si lascia che una delle variabili di stato del primo sistema sia in qualche modo accoppiata alla analoga variabile di stato del secondo sistema, si possono ottenere, purché siano soddisfatte alcune semplici condizioni, due sistemi le cui variabili di stato sono, istante per istante, identiche tra loro [14]. Immaginiamo allora di costruire due circuiti elettrici caotici identici (entro i limiti di tolleranza dei componenti) e di collegarli tra loro scegliendo una delle variabili di stato (nei circuiti elettronici le variabili di stato sono in genere le tensioni sui condensatori e le correnti

che fluiscono negli induttori); si scelga quindi un condensatore in un circuito e si colleghi uno dei suoi terminali, tramite una resistenza, al terminale equivalente dell'altro circuito.
I due circuiti si sincronizzano. Se ora sovrapponiamo un debole segnale *contenente informazione* al filo che collega tra loro i due circuiti questi restano sincronizzati, ma il segnale, confuso nel "rumore" dei sistemi caotici è totalmente inintelligibile. Sfruttando le proprietà di sincronizzazione dei due circuiti è però possibile ricostruire il segnale originale, ottenendo in questo modo la trasmissione sicura dello stesso. Se i primi esperimenti sono stati fatti utilizzando circuiti elettronici, recentemente alcuni ricercatori italiani hanno esplorato con successo la possibilità di sfruttare alcune proprietà dei laser [1]. In questa applicazione il segnale mascherato è un segnale video trasmesso via fibra ottica da un laser caotico e decrittato a destinazione utilizzando un altro sistema laser caotico.
Tra i sistemi naturali il cui funzionamento è riconducibile a modelli con dinamica caotica ci sono anche i fenomeni in cui si ha turbolenza. La turbolenza è comunemente presente nel funzionamento di molti strumenti a fiato, e la sintesi dei suoni di questi strumenti è per questo abbastanza complessa. L'uso di circuiti elettronici caotici è stato proposto per ovviare a questo problema e cercare di generare con sistemi molto semplici suoni anche molto ricchi da un punto di vista armonico [2].

6. CONCLUSIONI

La dinamica nonlineare ha ancora molte zone inesplorate e sulle quali è attiva la ricerca. Forse le più interessanti sono quelle relative ai sistemi di grandi dimensioni, tipicamente formati da un enorme numero di sistemi dinamici nonlineari interconnessi tra loro. Caratterizzare i fenomeni di sincronizzazione o dare una misura dei complicatissimi attrattori presenti in questi sistemi è ancora molto difficile e, in alcuni casi, arriva ai confini della potenza di calcolo attualmente disponibile in un normale calcolatore. Le applicazioni di questi studi, oltre che nel campo naturale dei sistemi meteorologici, permetterebbe probabilmente di migliorare la nostra conoscenza di sistemi complessi come il cervello.
I legami tra fenomeni deterministici e i fenomeni attualmente descritti esclusivamente in termini di processi casuali andrebbe forse approfondito; i sistemi dinamici nonlineari di grandi dimensioni possono infatti avere comportamenti difficilmente distinguibili da quelli che ci si aspetta da una variabile casuale.
Un'altra interessante area di sviluppo consiste nell'analisi delle biforcazioni e anche in questo caso l'uso dei calcolatori è indispensabile, se non per i casi banali. Dallo studio delle biforcazioni di un sistema dinamico è possibile prevedere cambiamenti inaspettati nella dinamica del sistema. Questi cambiamenti possono talvolta essere rilevanti e repentini (le cosiddette "catastrofi") e possono avere, se il sistema è il modello di un fenomeno fisico, conseguenze anche gravi e pericolose. Attualmente lo studio delle biforcazioni si limita, per i notevoli problemi di calcolo numerico che pone, a sistemi relativamente piccoli.
Tra le applicazioni ancora da sviluppare, le più attuali sono quelle relative all'analisi dei sistemi ecologici, del comportamento di popolazioni e dei mercati finanziari. È stata recentemente anche proposta la possibilità di utilizzare tecniche basate sulla dinamica nonlineare per la ricerca di informazioni in grandi database di informazioni non omogenee.

Ringraziamenti

Tutte le immagini sono state prodotte dall'autore con software open-source o freeware eccetto la figura 8, per la quale si ringrazia la Nasa/Goddard Space Flight Center. Si ringraziano i (numerosi) autori dei programmi (open-source) *Octave, Gnuplot e Gimp*. Un particolare ringraziamento a Nicolas Desprez autore del programma (freeware) *Chaoscope* (http://www.chaoscope.org) utilizzato per generare alcune delle figure incluse nel testo.

Bibliografia

[1] Annovazzi-Lodi V., Benedetti M., Merlo S., Norgia M., Provinzano B.: *Optical Chaos Masking of Video Signals*. IEEE Photonics Tech. Letters, 2005, p. 19-9.

[2] Bilotta E., Gervasi S., Pantano P.: *Reading Complexity in Chua's Circuit by Music*. Int. J. of Bifurcation and Chaos, 2005, p. 15-2.

[3] Eckmann J.P., Ruelle D.: Ergodic Theory of Chaos and Strange attractors. *Rev. Modern Phys*., Vol. 57, n. 3, 1985.

[4] Garfinkel A., Spano M.L., Ditto W.L., Weiss J.N.: Controlling Cardiac Chaos. *Science*, Vol. 257, p. 1230-1235.

[5] Goldberg A.L., West B.J.: Application of Nonlinear Dynamics to Clinical Cardiology. *Ann. N.Y. Acad. Sci*. Vol. 504, 1987, p. 195-212.

[6] Goldberg A.L., West B.J.: *Fractals in Physiology and Medicine*. Yale J. Biol.

[7] Kennedy M.P.: *Basic Concepts of Nonlinear Dynamics and Chaos*. In IEEE ISCAS 1994 Circuit and Systems Tutorials, ed. C.Toumazou, 1994.

[8] Kennedy M.P.: Three steps to Chaos – Part I : Evolution. *IEEE Transactions on Circuits and Systems – I Fundamental Theory*, Vol. 40, n. 10, 1993.

[9] Kennedy M.P.: Three steps to Chaos – Part II : A Chua Circuit primer. *IEEE Transactions on Circuits and Systems – I Fundamental Theory*, Vol. 40, n. 10, 1993.

[10] Lorenz Edward N.: Deterministic Nonperiodic Flow. *Journal of the Atmosferic Sciences*, Vol. 20, 1963.

[11] Lorenz Edward N.: The Mechanics of Vacillation. *Journal of the Atmosferic Sciences*, Vol. 20, 1963.

[12] Lorenz Edward N.: *The Problem of Deducing the Climate from the Governing Equations*. Tellus, Vol. 16, 1964.

[13] Poincaré Henri: *Geometria e Caso*. Universale Boringhieri, Torino, 1995.

[14] Strogatz S.H.: *Nonlinear Dynamics and Chaos*. Addison Wesley, N.Y. 1994.

APPENDICE - *SISTEMI DINAMICI NONLINEARI*

Qui di seguito vengono considerati alcuni aspetti più tecnici relativi ai sistemi dinamici nonlineari. Si tenga presente che non si pretende di essere rigorosi in senso matematico, e che alcune questioni senz'altro importanti, ma irrilevanti per una trattazione qualitativa come la presente, sono stati omessi.

1. *Unicità e continuità delle soluzioni*

Un sistema dinamico a tempo continuo può essere definito dalla equazione differenziale ordinaria:

$$\dot{\boldsymbol{x}}(t) = F[\boldsymbol{x}(t), t] \qquad (1)$$

dove $\boldsymbol{x}(t)$ è lo stato del sistema, che supponiamo essere definito su uno spazio a n dimensioni e la derivata di $\boldsymbol{x}(t)$ rispetto al tempo è indicata con $\dot{\boldsymbol{x}}(t)$. La funzione $F(\cdot\,,\cdot)$ descrive un *campo vettoriale*, in quanto definisce in ogni punto dello spazio di stato un vettore di "velocità".
Se supponiamo che $F(\cdot\,,\cdot)$ sia continua quasi ovunque e sia definita la condizione iniziale $\boldsymbol{x}_0 \equiv \boldsymbol{x}(t_0)$, *esiste*, è *unica* ed è *continua* una funzione del tempo $\phi(t, \boldsymbol{x}_0, t_0)$ che soddisfa le due condizioni:

$$\phi(t_0, \boldsymbol{x}_0, t_0) = \boldsymbol{x}_0$$

e

$$\dot{\phi}(t, \boldsymbol{x}_0, t_0) = F[\phi(t, \boldsymbol{x}_0, t_0), t]$$

La funzione $\phi(t, \boldsymbol{x}_0, t_0)$ è detta *soluzione* o *traiettoria* del sistema definito dalla (1).
Per semplicità supponiamo il campo vettoriale non esplicitamente dipendente dal tempo, possiamo quindi arbitrariamente fissare $t_0 = 0$ e definire formalmente le traiettorie con la notazione semplificata $\phi(t, \boldsymbol{x}_0, 0) \equiv \phi_t$ così da poter scrivere $\boldsymbol{x}(t) \equiv \phi_t(\boldsymbol{x}_0)$. L'introduzione di questo operatore è molto comoda, si ha infatti banalmente $\phi_{t1+t2} \equiv \phi_{t1}\,\phi_{t2}$, ma, soprattutto, data l'ipersfera $S_r(\boldsymbol{x}_0)$ di raggio r centrata su $\boldsymbol{x}_0$, possiamo scrivere $\phi_t[S_r(\boldsymbol{x}_0)]$ per indicare il fascio di traiettorie generate usando i punti della sfera come condizioni iniziali.

2. *Insiemi limite e attrattori*

Le traiettorie di un sistema dinamico stabile finiscono, eventualmente dopo un transitorio, in un insieme limite che rappresenta un sottoinsieme delle soluzioni di regime (stabili) del sistema. Ovviamente si possono avere più insiemi limite, ogni insieme limite rappresenta in questo caso un diverso *attrattore*, e i diversi insiemi di punti che portano ad ogni singolo attrattore sono chiamati *bacini di attrazio-*

ne. I bacini d'attrazione sono insiemi di punti dello spazio di stato, talvolta dotati a loro volta di geometrie molto complesse.
Definiamo meglio questi concetti e classifichiamo la struttura geometrica degli attrattori.
L'insieme limite Ω^+ definito dal comportamento del sistema per $t \to +\infty$, rappresenta le *soluzioni di regime* di un sistema e un punto $\boldsymbol{x}_\Omega$ è un *punto limite* di $\boldsymbol{x}_0$ se esiste una sequenza di tempi $t_k \to +\infty$ per la quale si abbia:

$$\lim_{k\to\infty} \phi_{t_k}(\boldsymbol{x}_0) = \boldsymbol{x}_\Omega$$

Tutti i punti che soddisfano questa condizione definiscono l'insieme limite Ω^+ della condizione iniziale $\boldsymbol{x}_0$ e che indichiamo con $\Omega^+(\boldsymbol{x}_0)$. Un insieme limite Ω^+ è *attrattivo* se esiste un intorno $S_\varepsilon(\Omega^+)$ tale per cui si abbia $\Omega^+(\boldsymbol{x}_0) = \Omega^+$, $\forall\, \boldsymbol{x}_0 \in S_\varepsilon(\Omega^+)$, e costituisce un *attrattore* se contiene almeno una traiettoria che si avvicina arbitrariamente a tutti i suoi punti in qualche istante di tempo.
Tutto quanto detto fino ad ora può essere ripetuto invertendo la freccia del tempo e definendo quindi gli insiemi limite Ω^- e trattando, di fatto, le situazioni non stabili.

3. *Classificazione degli insiemi limite – 1. i punti di equilibrio – regime costante*

I punti di equilibrio rappresentano il comportamento di regime più semplice e intuitivo. Sono tutti i punti dello spazio di stato dove si annulla il campo vettoriale definito dalla (1), cioè le soluzioni di $F(\boldsymbol{x}) = 0$ o, equivalentemente, i punti per cui si ha $\phi_t(\boldsymbol{x}_0) = \boldsymbol{x}_0$, $\forall_t$.
I sistemi nonlineari possono ovviamente avere più di un punto di equilibrio, si pensi per esempio ai singoli bit delle memorie dei calcolatori: questi circuiti sono generalemente dotati di tre punti di equilibrio, due stabili ed uno instabile. I due stati stabili rappresentano i valori binari 0 e 1, la transizione del bit di memoria tra i due stati avviene perturbando il circuito in modo da portarne il funzionamento da un bacino di attrazione all'altro.

4. *Classificazione degli insiemi limite – 2. le traiettorie periodiche – regime periodico*

Gli stati per cui si ha $\phi_T(\boldsymbol{x}_0) = \boldsymbol{x}_0$ per qualche valore $T > 0$ sono detti stati periodici. Anche i punti di equilibrio soddisfano questa condizione, e il regime costante può essere considerato infatti un caso particolare di quello periodico. Se consideriamo gli stati periodici in senso proprio (quelli che non soddisfano le condizioni di regime costante), possiamo definire *ciclo limite* una traiettoria chiusa e isolata del sistema. Tutti gli oscillatori, uno tra i circuiti più diffusi in qualsiasi sistema elettronico, sono caratterizzati proprio da un ciclo limite stabile.
Ovviamente se uno stato è periodico di periodo T lo è anche di periodo $2T$, $3T$... Il piu' piccolo valore di T per cui è soddisfatta la condizione di periodicità è chiamato *periodo* del ciclo limite. Geometricamente, nello spazio di stato, un ciclo limite corrisponde ad una linea chiusa.

5. *Classificazione degli insiemi limite – 3. il regime quasiperiodico*

In alcuni casi le soluzioni di un sistema dinamico possono avere comportamenti più complicati di quelli visti sopra, se per esempio si ha un sistema dinamico di ordine tre o superiore, è possibile che le traiettorie siano esprimibili come combinazione di due funzioni periodiche di periodo incommensurabile. In questo caso le traiettorie non saranno propriamente periodiche, ma descriveranno nello spazio di stato una superficie toroidale.

6. *Classificazione degli insiemi limite – 4. il regime caotico e ipercaotico*

Se un sistema dinamico autonomo è dotato di un insieme limite attrattivo, ma non sono soddisfatte le condizioni viste sopra, allora si è in un caso di regime caotico o ipercaotico. Classificare questi tipi di insiemi limite non è banale. Da un punto di vista sperimentale il comportamento caotico si manifesta in modo apparentemente simile ad un processo casuale, resta confinato all'interno di un attrattore, e non mostra nessuna periodicità o quasiperiodicità... però è deterministico.
Una risposta quantitativa al problema della classificazione dei comportamenti di regime di un sistema dinamico è data dal calcolo degli esponenti di Lyapunov di una traiettoria (o di un insieme di traiettorie). Gli esponenti di Lyapunov sono in numero equivalente alle dimensioni dello spazio di stato del sistema e rappresentano una misura "statistica" della sensitività del sistema alle condizioni iniziali.
Consideriamo una piccola sfera $S_\varepsilon(\boldsymbol{x})$ di condizioni iniziali, usando la notazione introdotta sopra, dopo un tempo t questa sfera di condizioni iniziali sarà stata trasformata dal campo vettoriale del sistema in un ellissoide $\phi_t[S_\varepsilon(\boldsymbol{x})]$. In generale l'ellissoide avrà degli assi principali di lunghezza differente, e, se la sfera iniziale ha diametro 2ε, ci si può ragionevolmente aspettare che tra gli assi principali dell'ellissoide ce ne siano alcuni di lunghezza maggiore di 2ε, e alcuni di lunghezza inferiore. Se per esempio siamo nel caso di una traiettoria che converge verso un punto di equilibrio stabile troveremo che l'ellissoide corrisponde ad una *contrazione* della sfera iniziale e tutti i suoi assi saranno di lunghezza in-

feriore a 2ε. Questa operazione viene ripetuta più volte lungo la traiettoria, normalizzando ogni volta gli assi dell'ellissoide e procedendo per piccoli passi temporali. Se la sfera iniziale ha, ad ogni passo, diametro 2ε, calcoliamo per ogni passo p i numeri:

$$\eta_{jp} = \log \frac{\alpha_{jp}}{2\varepsilon}$$

dove con α_{jp} abbiamo indicato il j-esimo asse principale dell'ellissoide al passo p. Definiamo esponenti di Lyapunov λ_j i numeri che otteniamo come valore medio, dopo un tempo sufficientemente lungo, dei valori η_{jp} e rappresentano il valore medio esponenziale della separazione delle traiettorie lungo il flusso definito dal campo vettoriale. Il metodo qui esposto per il calcolo degli esponenti di Lyapunov non è particolarmente efficiente, il calcolo viene abitualmente eseguito utilizzando il sistema variazionale associato al sistema dinamico. Sfruttando le proprietà di ergodicità dei flussi definiti dai sistemi dinamici è poi possibile convergere ai valori esatti molto più rapidamente. Per una trattazione più approfondita si rimanda a [3, 8, 14].
Nella tabella 1 sono riassunte le varie caratteristiche dei comportamenti possibili in un sistema dinamico autonomo, in base ai valori degli esponenti di Lyapunov ed alla dimensione geometrica degli insiemi limite corrispondenti.
La dimensione degli attrattori strani è non intera, si tratta infatti di oggetti frattali, i casi più conosciuti e studiati, i sistemi caotici di dimensione tre, hanno attrattori di dimensione compresa tra due e tre, come negli esempi mostrati nella figura 5 e nella figura 6.

TABELLA 1 - CLASSIFICAZIONE DEI POSSIBILI COMPORTAMENTI DI UN SISTEMA DINAMICO

Regime	Insieme limite	Esp. di Lyapunov	Dimensione
Costante	punto di equilibrio	$0 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$	0
Periodico	linea chiusa	$\lambda_1 = 0,$ $0 \geq \lambda_2 \geq \cdots \geq \lambda_n$	1
Quasiperiodico	k-toro	$\lambda_1 = \lambda_2 = \cdots = \lambda_k = 0$ $0 \geq \lambda_{k+1} \geq \cdots \geq \lambda_n$	k
Caotico	frattale	$\lambda_1 > 0,\ \lambda_2 = 0$ $\sum_{j=1}^{n} \lambda_j < 0$	non intera
Ipercaotico	frattale	$\lambda_1, \lambda_2, \ldots \lambda_k > 0,\ \lambda_{k+1} = 0$ $\sum_{j=k+2}^{n} \lambda_j < 0$	non intera

GIANCARLO STORTI GAJANI è Professore Associato di Elettrotecnica presso il Dipartimento di Elettronica e Informazione del Politecnico di Milano. Attualmente i suoi campi di interesse riguardano la teoria dei circuiti, lo studio delle proprietà dei circuiti dinamici nonlineari e la simulazione dei circuiti elettronici. Su questi argomenti è autore o coautore di circa 40 lavori pubblicati su rivista o presentati a congresso.
E-mail: storti@elet.polimi.it