UNIVERSITÀ DEGLI STUDI DI TORINO

Facoltà di Economia

Corso di Laurea in Economia e Commercio

Tesi di Laurea

Simulazione delle organizzazioni tramite jES: il 118

RELATORE:

Prof. Pietro Terna

CORRELATORE:

Prof. Sergio Margarita

CANDIDATO:

Carlo Desotgiu

ANNO ACCADEMICO 2003-2004

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Le scienze hanno l'obiettivo di conoscere, capire e spiegare gli avvenimenti circostanti, cercando di rendere semplici fenomeni che all'apparenza sembrano complessi.

I sistemi semplici sono facili da comprendere e aiutano l'uomo nel creare modelli per rappresentare la realtà, ma in molti casi non sono sufficienti a fornire spiegazioni adeguate. Si pensi ad esempio ad un'organizzazione come il 118: è evidente che non sarebbe possibile rappresentare il suo funzionamento con un sistema lineare; date le condizioni iniziali, infatti, non sarebbe possibile prevedere come la gestione delle ambulanze verrebbe condotta, perché gli imprevisti dovuti a traffico, urgenze, code, non sono a priori prevedibili. Il funzionamento di una qualsiasi organizzazione non è rappresentabile da un sistema lineare; questo vale per l'economia e per le altre discipline che si occupano dell'uomo.

La realtà è caratterizzata soprattutto da sistemi complessi, che hanno caratteristiche opposte a quelle dei sistemi semplici. In un sistema complesso molte cause producono un dato effetto, ma gli effetti delle cause non sono indipendenti tra loro e non possono quindi essere isolati; le cause non si possono sommare tra di loro per determinare un effetto, in quanto non lineari.

I sistemi complessi sono in genere composti da moltissimi elementi, i quali interagiscono solo con un ristretto numero di altri elementi di cui è composto l'intero sistema; da queste numerose interazioni locali emergono proprietà globali dell'intero sistema che non sono prevedibili o deducibili a priori anche se conosciamo alla perfezione come sono fatti i singoli elementi e in che modo interagiscono tra di loro.

Obiettivo di questo lavoro è lo studio del funzionamento della centrale operativa del 118 tramite la simulazione ad agenti, che consente di analizzare sistemi complessi nel senso definito precedentemente; questo tipo di simulazione risulta di particolare utilità nello studio dei fenomeni sociali.

Nella definizione data da Parisi (2001), la simulazione ad agenti è un particolare tipo di simulazione che cerca di riprodurre fenomeni collettivi facendo interagire tra di loro un certo numero di agenti, i quali hanno determinate regole, ossia reagiscono in modo determinato agli stimoli che ricevono. Il comportamento del singolo agente è in grado di influenzare i comportamenti di altri agenti e da questa interazione emerge la complessità.

Capitolo 2

La complessità

2.1 L'uomo è un risparmiatore cognitivo

Gli esseri umani da sempre cercano di rendere semplice la realtà che li circonda. Il cervello umano tende a cancellare tutte le informazioni inutili registrate durante il giorno, per avere la capacità di memorizzare, elaborare e conservare le informazioni che invece saranno utili all'uomo per la sua sopravvivenza; due importanti psicologi (Pratkanis, Aronson, 1992) hanno usato la definizione di *risparmiatore cognitivo*; l'uomo cerca cioè di conservare la propria energia cognitiva. Le scienze hanno da sempre cercato di fare la stessa cosa: conoscere, capire, elaborare e spiegare fenomeni rendendo semplici fenomeni che all'apparenza sembrano complessi. Tutto ciò è riuscito abbastanza bene nelle scienze naturali, mentre i risultati delle scienze sociali sono stati decisamente meno positivi.

2.2 Scienze naturali, scienze sociali

Come mai una così grande differenza di risultati tra le scienze dell'uomo e quelle della natura? Innanzitutto bisogna dire che la scienza ha come obiet-

tivi l'osservazione, la descrizione e l'analisi dei fatti empirici e poi la loro spiegazione. Per fare ciò vengono elaborate teorie che spiegano determinati fenomeni e che permettono (o dovrebbero permettere) di prevederli e controllarli.

Le scienze dell'uomo crescono dall'ottocento in poi, ma non sono mai arrivate ad avvicinarsi ai risultati ottenuti dalle scienze naturali. In queste ultime, infatti, si verifica un confronto costante tra teorie e fatti osservati, le teorie sono espresse in termini quantitativi e c'è quasi sempre la possibilità si effettuare gli esperimenti in laboratorio. Nel laboratorio si possono osservare i fenomeni in condizioni controllate, escludendo così i fattori e le variabili irrilevanti, e si possono manipolare le condizioni che fanno verificare certi fenomeni osservando le conseguenze di queste manipolazioni, ottenendo un elevato numero di informazioni sulla realtà; ma soprattutto in laboratorio vengono messe alla prova le predizioni empiriche, cioè le ipotesi tratte da una teoria.

Nelle scienze naturali le teorie corrono immediatamente a cercare una conferma o una smentita nei fatti, e i fatti sono subito confrontati con una teoria che ne determini una spiegazione e la comprensione. Nelle scienze dell'uomo ci sono sia le teorie sia i fatti empirici, ma c'è scarso dialogo e interazione tra essi (Parisi, 2001). Le teorie formulate sono spesso anche molto elaborate, ambiziose, suggestive, ma non riescono a trovare conferma nei dati e quindi da esse non si possono derivare previsioni empiriche esatte.

In sostanza quello che manca nelle scienze dell'uomo è il *circolo virtuoso* tra teorie e fatti che caratterizza la scienza in generale e le scienza della natura in particolare (Parisi, 2001).

2.3 L'uomo e la complessità

L'uomo, oltre a cercare di semplificare i fatti che lo circondano, tende a pensare che la realtà sia essa stessa caratterizzata da sistemi semplici. In un sistema semplice una singola causa produce un singolo effetto, per cui, per ottenere un determinato effetto, è sufficiente fare in modo che la causa si verifichi, mentre per evitarlo è sufficiente bloccare la causa. L'effetto può anche essere prodotto da più cause, ma in genere queste non sono molte e, soprattutto, il ruolo che ciascuna causa ha nel produrre l'effetto è separabile da quello delle altre. In altre parole l'effetto è determinato dalle somma delle diverse cause, perché queste non interagiscono tra di loro e possiamo quindi isolare e prevedere l'effetto di una delle cause senza preoccuparci delle altre. Tutto ciò si esprime dicendo che i sistemi semplici sono lineari. I sistemi semplici hanno delle caratteristiche peculiari (Parisi, 2001):

- Gli stati successivi di un sistema semplice sono prevedibili se si conoscono gli stati precedenti; si può quindi prevedere il futuro del sistema;
- Se un sistema semplice si trasforma nel tempo, il modo in cui si trasforma è prevedibile;
- Se un sistema semplice è *perturbato* da un evento esterno, l'effetto della perturbazione sul sistema è commisurato all'entità della perturbazione: una perturbazione piccola produrrà un effetto piccolo e viceversa;
- Se due sistemi semplici partono da condizioni iniziali diverse, il loro sviluppo nel tempo sarà diverso e questa diversità sarà tanto più grande quanto più sono diverse le condizioni iniziali;
- Un sistema semplice può venire isolato dal contesto, cioè il fatto di operare in contesti diversi non cambia il funzionamento del sistema;

- Un sistema semplice tende a non essere coinvolto in rapporti di causazione reciproca, un elemento del sistema ne influenza un altro, ma non viceversa; se il sistema *vive* in un ambiente, l'ambiente può influenzare il sistema ma il sistema non influenza l'ambiente;

- Un sistema semplice tende a non essere parte di una gerarchia di sistemi, dove quello che è un intero sistema formato dai suoi elementi a un certo livello diventa un singolo elemento di un sistema più grande a un livello più alto della gerarchia, e ci sono influenze reciproche tra un livello e il successivo;

- Un sistema semplice è fatto di parti (elementi) il cui ruolo nel determinare il comportamento complessivo del sistema è ben individuabile;

- Un sistema semplice può essere riprodotto in copie identiche.

I sistemi semplici sono facili da comprendere e aiutano l'uomo nel creare modelli per rappresentare la realtà, ma in molti casi non sono sufficienti a fornire spiegazioni adeguate. Si pensi ad esempio ad un'organizzazione come il 118: è evidente che non sarebbe possibile rappresentare il suo funzionamento con un sistema lineare; date le condizioni iniziali, infatti, non sarebbe possibile prevedere come la gestione delle ambulanze verrebbe condotta, perché gli imprevisti dovuti a traffico, urgenze, code non sono a priori prevedibili. Il funzionamento di una qualsiasi organizzazione non è rappresentabile da un sistema lineare, ma non solo; il discorso vale anche per l'economia e per le altre discipline che si occupano dell'uomo. Il comportamento di tali sistemi è difficile da rappresentare e da prevedere perché non può essere ricavato dalla somma dei comportamenti delle singole parti che lo compongono.

A causa di tutti questi problemi stanno suscitando grande interesse gli studi sui modelli non lineari per cercare di comprendere meglio i fenomeni caratterizzati dalla complessità. Le scienze della complessità si occupano di

studiare questo tipo di fenomeni; il loro ruolo è spiegare il funzionamento di quei sistemi che non possono essere studiati utilizzando le metodologie *classiche*.

La realtà è fatta anche di sistemi complessi, anzi, è fatta soprattutto di sistemi complessi, e questi hanno caratteristiche opposte a quelle dei sistemi semplici. In un sistema complesso molte cause producono un dato effetto, ma gli effetti delle cause non sono indipendenti tra loro e non possono quindi essere isolati; le cause non si possono quindi sommare tra di loro per determinare un effetto in quanto non lineari. I sistemi complessi sono in genere composti di moltissimi elementi, i quali interagiscono solo con un ristretto numero di altri elementi di cui è composto l'intero sistema, e da queste numerose interazioni locali emergono proprietà globali dell'intero sistema che non sono prevedibili o deducibili a priori anche se conosciamo alla perfezione come sono fatti i singoli elementi e in che modo interagiscono tra di loro. Un sistema complesso presenta le seguenti caratteristiche (Parisi, 2001):

- non è possibile prevedere gli stati futuri del sistema pur conoscendo gli stati precedenti;
- il sistema si trasforma nel tempo ma la sua evoluzione è imprevedibile;
- quando il sistema è perturbato da un evento esterno, l'effetto che tale perturbazione ha sul sistema non può essere commisurato all'entità della perturbazione (lievi perturbazioni possono influenzare molto il sistema);
- il sistema è molto sensibile alle condizioni iniziali; due sistemi analoghi che partono
- da condizioni molto simili possono divergere notevolmente nel tempo;

- il sistema è sensibile al contesto in cui opera e quindi non può essere isolato dal suo ambiente;
- nel sistema ci sono rapporti di causazione reciproca: un elemento del sistema ne influenza un altro ed a sua volta è influenzato da questo;
- il sistema è inserito in una gerarchia di sistemi che si influenza a vicenda;
- non è ben identificabile il ruolo che ciascun elemento di un sistema ha nel determinare il comportamento globale del sistema;
- il sistema non può essere riprodotto in copie identiche.

I problemi maggiori nello studio dei sistemi complessi si riscontrano negli esperimenti di laboratorio; in laboratorio il ricercatore fondamentalmente manipola una causa (facendo in modo che si verifichi o meno, o graduandone quantitativamente il valore) e osserva qual è l'effetto. Ma se questo produce buoni risultati quando si studia un sistema semplice, in cui gli effetti derivano da cause singole o dalla loro somma, per un sistema complesso non ha molto senso cercare di isolare e manipolare una singola causa poiché gli effetti derivano da molte (moltissime) cause interconnesse tra di loro. Inoltre il metodo sperimentale richiede la ripetibilità dei fenomeni, la possibilità di riprodurli in forma identica. Questo si può fare per i sistemi semplici, ma non per i sistemi complessi che tendono ad essere molto sensibili alle condizioni iniziali, per cui differenze anche minime nelle condizioni iniziali determinano sviluppi anche molto diversi, e quindi la non riproducibilità di un fenomeno in forma identica. E' però in atto un cambiamento; si sta cominciando ad accettare che la realtà è fatta anche di sistemi complessi; il tempo meteorologico, il movimento all'interno di un fluido, le interazioni tra i miliardi di cellule che determinano il funzionamento complessivo del corpo, tra le diverse componenti di un ecosistema, e quelle tra attori economici che costituiscono

un mercato sono solo alcuni dei moltissimi esempi di sistemi complessi, e la scienza sta cominciando a cercare metodi di studio alternativi a quelli usati finora per poterli comprendere.

2.4 Una definizione di complessità

Si è fin qui parlato di realtà e sistemi complessi, ma non si è ancora spiegato in quale accezione la parola "complessità" sarà utilizzata in questo testo. Spesso, infatti, la parola "complesso" è intesa come sinonimo di complicato, cioè di non facile comprensione. Per Stein (1989),

> ...complexity is almost a theological concept; many people talk about it, but nobody knows what "it" really is. For example, organization and systems are often called complex, not because they are seen as dynamic and adaptive, but because they defy easy notion as to how they are organized or function.

Per chiarire la differenza che intercorre tra il termine complesso e complicato può essere utile descrivere due sistemi indubbiamente "difficili" da comprendere come il motore di un'automobile ed il formicaio.

Il motore è composto da molti meccanismi, spesso anche sofisticati, ma smontandolo si riesce a comprendere come ciascuna parte influenza il sistema, e può quindi essere studiato attraverso una sua scomposizione nelle componenti di base.

Si consideri ora un formicaio: anch'esso è certamente un sistema complicato; composto da migliaia di formiche, possiede caratteristiche affascinanti come la termoregolazione, che permette al formicaio di mantenere la temperatura costante, o con minime variazioni, sia in estate che in inverno. Ma non è possibile scomporre il formicaio e studiare come le singole formiche influenzino il fenomeno; la temperatura costante del formicaio non rientra,

infatti, tra gli obiettivi della formica. Il formicaio è un sistema complesso poiché da esso emergono fenomeni che non sono spiegabili con la semplice analisi delle caratteristiche delle sue componenti.

Se, dunque, per studiare i sistemi lineari (il motore di un'automobile), si procede alla loro scomposizione ed allo studio analitico di ciascuna delle sue parti, questo non può avvenire per lo studio dei sistemi non-lineari (il formicaio).

L'economia, come tutte le scienze dell'uomo, è caratterizzata principalmente da sistemi complessi, il cui comportamento dipende dall'interazione delle parti più che dal comportamento delle parti stesse; l'impresa, come i mercati e le organizzazioni in generale sono solo uno dei tanti casi di complessità per cui, in questo lavoro, si propone un approfondimento sui temi della complessità.

E' importante segnalare che, data la sua natura interdisciplinare, non esiste un consenso generale sulla definizione di sistema complesso e di complessità e, attualmente, si possono individuare (Kilpatrick, 2001) due approcci all'interno delle scienze della complessità. La prima studia quello che viene definito "margine del caos" (edge of chaos) e deriva principalmente dalle scienze matematiche e fisiche, mentre la seconda è associata al concetto dei "sistemi adattivi complessi" (complex adaptive system, CAS).

Con il termine caos si indica generalmente la natura di situazioni complesse che sono proprie di qualsiasi settore scientifico, dalla turbolenza dei fluidi alle fluttuazioni delle popolazioni. In termini matematici la teoria del caos ha principalmente origine dalla dinamica lineare, dallo studio dei fenomeni retti da equazioni differenziali non lineari dove sistemi anche molto semplici possono manifestare una notevole complessità di comportamento.

Anche un sistema caotico, tuttavia, può, ad un certo punto, presentare delle regolarità, risultando così apparentemente non più caotico, come se si

adattasse all'ambiente che lo ospita. Leggiamo in Waldrop (1992):

> ... at a kind of abstract phase transition called "the edge of Chaos," you also find complexity: a class of behaviours in which the components of the system never quite lock into place, yet never quite dissolve into turbulence, either.
>
> These are systems that are both stable enough to store information, and yet evanescent enough to transmit it. These are the systems that can be organized to perform complex computations, to react to the world, to be spontaneous, adaptive, and alive.

Le teorie associate alla biologia hanno anch'esse un ruolo importante nello sviluppo della teoria della complessità, ma seguendo ragionamenti diversi. In questo caso gli "agenti", la cui dinamica di gruppo è differente dal comportamento dei singoli, reagiscono e si adattano ai cambiamenti esterni; in parole povere: si evolvono. Sistemi come questi vengono definiti "sistemi adattivi complessi" (CAS).

Tali sistemi non necessariamente presentano comportamenti caotici: un'organizzazione come il 118, una multinazionale, un mercato borsistico, un distretto industriale, possono essere visti come dei sistemi adattivi complessi. Ogni sistema adattivo complesso può essere costituito da altri sistemi analoghi al suo interno e la sua evoluzione è legata ad essi; le parti che lo compongono possono agire in modo imprevedibile e gli effetti delle loro azioni dipendono dalle azioni delle altre parti. Un mercato borsistico, ad esempio, è composto da venditori, compratori, società quotate ed organizzazioni regolatrici del mercato. Ogni attore del sistema opera secondo le sue strategie per perseguire i propri scopi, questi possono essere condivisi con altri attori o essere individuali, ma il formarsi di bolle speculative emerge dall'interazione tra gli attori. Un sistema adattivo complesso presenta caratteristiche per le quali è praticamente imprevedibile la sua evoluzione, tuttavia, in alcuni

casi, è possibile identificare comportamenti ciclici e situazioni ricorrenti che permettono una certa prevedibilità a corto termine e a lungo termine.

Per quanto concerne la ricerca scientifica in campo economico, in Terna (2002) si trova la visione economica di Santa Fe (in riferimento al Santa Fe Institute, dedicato agli studi sulla complessità), espressa da Arthur, Durlauf e Lane(1997). In questo approccio si suggeriscono sei caratteristiche che, sebbene presentino molte difficoltà ad essere trattate con i tradizionali strumenti matematici, i modelli economici dovrebbero presentare:

- interazione dispersa tra gli agenti che compongono il sistema;
- nessuna capacità di controllo del modello;
- organizzazioni gerarchiche che si intersecano;
- adattamento continuo degli agenti che sono in grado di imparare ed evolvere;
- innovazioni continue (nuovi mercati, tecnologie, . . .);
- dinamica (intesa come sequenza di modificazioni del sistema nel tempo) priva di un equilibrio globale, ma con molti punti di equilibrio instabili.

Sistemi che presentano queste caratteristiche sono definiti dagli autori "adaptive nonlinear networks"; Arthur sottolinea che si tratta di una riproposizione del lavoro di Hayek il quale, a sua volta, può essere fatto risalire ai principi contenuti nei lavori di Smith sulla *mano invisibile*, in particolare nella convinzione che

> ...l'economia è il risultato dell'azione umana, ma non di un progetto degli uomini

(Terna, 2002).

2.5 L'economia e la complessità

L'economia tradizionale presuppone che i gusti e le preferenze dei consumatori rimangano fissi. Questa semplificazione si scontra con ciò che avviene nella realtà dove non solo i gusti delle persone cambiano, ma sovente sono influenzati dai gusti e dalle opinioni delle altre persone. Un esempio intuitivo di quanto detto viene dall'industria cinematografica, dove la presenza di attori famosi, grandi investimenti pubblicitari e scenografie molto costose non garantiscono il successo del film al botteghino. Questo avviene perché un aspetto importante nelle scelte individuali dei consumatori è dato dal passaparola e cioè dalle opinioni di amici e conoscenti che hanno già visto il film; questo meccanismo porta alcuni film minori ad avere un grandissimo successo di pubblico ed alla debacle di film molto costosi.

La razionalità olimpica di cui l'agente economico dispone secondo l'economia tradizionale non è ovviamente presente nella realtà, ma gli studi sull'economia si scontrano con problemi che non riguardano solo l'informazione completa e la perfetta capacità di calcolo, ma anche con la complessità che deriva dall'interazione tra gli individui e tra le persone e l'ambiente circostante.

In un sistema complesso i comportamenti degli individui e le conseguenze che ne derivano non sono prevedibili nemmeno conoscendo perfettamente tutti gli elementi di partenza. Per l'economia vale lo stesso discorso; Ormerod (2003) sottolinea che:

> ...le economie e le società non sono macchine. Somigliano di più a organismi viventi. Gli individui non agiscono isolati gli uni dagli altri, ma si influenzano a vicenda attraverso modalità complesse. [...] La società è simile a una creatura vivente, e come una creatura vivente è capace di adattarsi e apprendere. Il comportamento complessivo del sistema non può essere dedotto sommando

> in modo meccanico le sue parti; così come un organismo vivente non è semplicemente la somma delle singole cellule di cui è costituito, il sistema economico e sociale è più della somma degli individui che lo compongono.

In questa maniera si spiega perché nella realtà non ci sia una connessione regolare tra l'ampiezza di un evento e le sue conseguenze; a volte eventi di grandi dimensioni provocano conseguenze modeste, mentre altre volte piccoli eventi possono causare conseguenze di grande rilievo e del tutto imprevedibili. Un esempio ricorrente nei testi che trattano la teoria del caos è quello del battito d'ali di una farfalla che può provocare un tornado all'altro capo del globo. Tale imprevedibilità a livello di singolo individuo o avvenimento non per forza si estende all'intero sistema; secondo Ormerod (2003),

> ...da questo rapporto spesso sfocato e indistinto tra gli avvenimenti e le loro ripercussioni si deve trarre un'importante lezione: prevedere l'immediato futuro con un qualsivoglia grado di precisione è nel migliore dei casi difficile, talvolta addirittura impossibile. Tuttavia nel lungo termine, esiste una notevole regolarità di comportamento. Le interazioni tra gli individui, spesso imprevedibili, portano a una qualche forma di autoregolazione nel comportamento complessivo del sistema. Anche se non siamo in grado di dire dove si troverà esattamente il sistema in un dato istante, spesso siamo in grado di delimitare l'area entro cui si muoverà.

2.6 La complessità e le formiche

Sul tema della complessità è interessante parlare del parallelo che si può tracciare tra gli uomini e le formiche. A molti questo paragone potrà sembrare

azzardato o quanto meno non ovvio, ma in realtà spiega bene la complessità che caratterizza la società in cui viviamo.

Le formiche sono, a differenza degli uomini, organismi semplici ed è probabile che potremmo conoscere e formalizzare le equazioni che descrivono il comportamento delle singole formiche, ma la loro interazione fa sì che non si possa conoscere in anticipo il comportamento di un gruppo di esse. La complessità non è nella formica: è nel formicaio.

Un esperimento condotto da alcuni entomologi, riportato nel libro di Ormerod (2003) serve a capire meglio quanto detto. I ricercatori misero alla stessa distanza da un formicaio due mucchietti identici di cibo, e li reintegrarono costantemente in modo che restassero sempre uguali. In altre parole, ogni volta che una formica portava via una briciola, al mucchietto intaccato veniva aggiunta un'altra briciola. L'esperimento consisteva nel verificare come si sarebbero comportate le formiche; quale mucchietto avrebbero scelto? Sarebbero andate tutte su di uno o si sarebbero divise equamente?

A chi non conosce le formiche verrebbe probabilmente in mente che il rapporto sarebbe stato all'incirca del 50:50, ma i biologi avevano elaborato una versione più sofisticata di questa ipotesi, basata su una particolare caratteristica del comportamento delle formiche. Se una formica ha avuto successo nella ricerca di cibo, in futuro tenderà a tornare verso lo stesso mucchietto. Ma quando una formica che ha trovato del cibo fa ritorno al formicaio, stimola fisicamente un'altra formica a seguirla verso il luogo del rifornimento attraverso una secrezione chimica. Pertanto, una formica che esce per la prima volta dal formicaio sarà influenzata dal comportamento delle altre formiche. L'esistenza delle scie implica che le scelte casuali delle formiche uscite per prime dal formicaio potrebbero esercitare un'influenza decisiva sul comportamento dell'intera colonia e, secondo questo ragionamento, i biologi prevedevano che il valore finale del rapporto sarebbe dipeso dalle scelte com-

piute dalle formiche durante le prime fasi del processo di ricerca del cibo. Tale valore, inoltre, si sarebbe dovuto stabilizzare nel tempo.

In realtà l'esperimento diede risultati del tutto diversi. Il numero di formiche che sceglieva un determinato mucchietto variava in modo apparentemente aleatorio. Il rapporto medio era di 50:50, ma questa situazione non si verificava quasi mai nella realtà, e le fluttuazioni erano continue. La situazione sembrava stabilizzarsi quando una larga maggioranza di formiche sceglieva un mucchietto, ma alla fine tale maggioranza si sgretolava e le formiche prendevano a privilegiare l'altro sito. Il conflitto tra le ipotesi di partenza ed i risultati spinsero i ricercatori ad apportare modifiche all'esperimento, ma il risultato finale non cambiò. Usando le parole di Ormerod (2003) si può dire che

> ...il comportamento delle formiche, la loro influenza diretta sulle scelte delle altre e le conseguenze sull'intero formicaio dell'interazione forniscono una descrizione generale, o modello, di un'ampia gamma di fenomeni economici e sociali. I principi che regolano il comportamento delle formiche valgono infatti anche per gli esseri umani.

L'esempio delle formiche è utile per comprendere come l'esito delle scelte in un sistema complesso non è a priori prevedibile, ma non è certo l'unico che si può fare; gli stessi meccanismi di fondo si presentano anche in campi del tutto diversi. Prima si è parlato del mercato cinematografico, ma le stesse considerazioni valgono per i mercati azionari, la moda, la televisione ecc...

Estremamente interessante è un ulteriore esempio presente nel libro di Ormerod che illustra come il processo di scelta porti a volte a far prevalere un prodotto tecnologicamente inferiore tra due in competizione. Il caso in questione è quello dell'antagonismo fra i sistemi di videoregistrazione Betamax e Vhs. Gli apparecchi Betamax erano più facili da usare e possedevano

alcune caratteristiche che ancora oggi non sono disponibili sul modello Vhs, ma come tutti ben sanno il modello Vhs ha persino estromesso dal mercato l'avversario.

Come è potuto accadere che i consumatori abbiano scelto il prodotto inferiore? Qualsiasi modello economico tradizionale avrebbe predetto il contrario, in quanto ogni consumatore dovrebbe accedere a tutte le informazioni, elaborarle e scegliere di conseguenza. Il problema è proprio nelle informazioni. In realtà, infatti, al momento dell'ingresso di una nuova tecnologia sul mercato, la grande maggioranza dei consumatori ha poche informazioni su di essa, e deve prendere le proprie decisioni in un contesto di incertezza; a questo punto è ragionevole pensare che il comportamento dei primi acquirenti influenzi quello degli altri in almeno due modi: fornendo una prima indicazione, sia pure sbagliata, sulla convenienza ad acquistare un prodotto piuttosto che un altro, e dando via al meccanismo del passaparola. A questo punto si aggiungono altre leggi di mercato, per cui, per esempio, la superiore quota di mercato raggiunta dal Vhs spinge i commercianti a tenere nastri per Vhs anziché per Betamax, e ciò incentiva i nuovi acquirenti a scegliere apparecchi Vhs. Si arriva così all'estromissione del modello Betamax dal mercato.

2.7 L'economia ortodossa e la complessità

Considerando quanto detto finora, è lecito chiedersi perché gran parte degli economisti del passato e di oggi sviluppino teorie ed elaborino modelli così poco aderenti alla realtà. E' ovvio che gli studiosi che gettarono le basi della moderna teoria economica fossero ben consci dei limiti di realismo dei loro modelli, ma c'è da dire che a quel tempo gli strumenti matematici e le capacità di calcolo attuali non esistevano e la dimostrazione dell'equilibrio generale, pur in presenza di vincoli stringenti, fu un grande passo in avanti.

Un esempio su come gli economisti dell'epoca prestassero attenzione a come rendere più realistici i modelli economici è dato dai pensieri e dalle opere di un famoso economista dell'epoca: Alfred Marshall. Ecco le sue riflessioni su questo argomento:

> Avverto sempre più forte la sensazione che con i teoremi matematici difficilmente si possa fare buona economia, e sempre più spesso mi baso sulle seguenti regole:
>
> 1. usa le formule matematiche come un espediente stenografico più che come un motore di indagine
> 2. servitene finchè sei arrivato in fondo
> 3. traduci in parole
> 4. chiarisci con esempi importanti della vita reale
> 5. brucia le formule
> 6. se non ti riesce 4, brucia 3
>
> Quest'ultima è una cosa che faccio spesso.

La difficoltà delle teorie economiche tradizionali di spiegare la realtà è dovuta a varie cause, ma la prima fra queste è che quasi sempre tali teorie si basano su relazioni causa-effetto lineari che non includono la possibilità del cambiamento di preferenze o idee in base all'interazione tra gli individui. Questo ed altri problemi affliggono una teoria ultimamente in voga nel mondo economico: la teoria del *ciclo economico reale*. Questa cerca di spiegare secondo le logiche dell'economia ortodossa la presenza nelle moderne economie del ciclo economico, che possiede caratteristiche di complessità: si ha infatti una costante variabilità del tasso annuo di crescita del reddito nazionale di ogni Paese, e inoltre i cicli economici non sembrano avere regolarità, né per quanto riguarda la durata, né per quanto riguarda l'ampiezza. La teoria

del ciclo economico reale piace agli studiosi convenzionali in quanto spiega l'esistenza del ciclo seguendo la tradizionale linea ortodossa. Essa ha però anche eminenti oppositori; il noto economista Joseph Stiglitz, iniziò il suo discorso alle Marshall Lectures dell'Università di Cambridge in questo modo:

> La teoria del ciclo reale...

disse prendendo il primo foglio dei suoi appunti e gettandolo a terra,

> ...questo è praticamente tutto quello che c'è da dire sull'argomento.

Altrettanto forte nei toni è la dichiarazione di Paul Krugman:

> La teoria del ciclo reale sta diventando simile a un movimento politico estremista che a furia di epurazioni finisce con il ridursi a uno sparuto gruppo di militanti.

Anche in alcune riviste economiche importanti, come l'American Economic Review, sono stati pubblicati articoli molto critici nei confronti di questa teoria, ma qui non importa tanto discutere delle opinioni sulla validità della teoria, quanto esaminare come trattando il problema in un'altra maniera si possono ottenere risultati migliori. Le caratteristiche di base del modello del ciclo reale sono quelle classiche: gli agenti possiedono una conoscenza completa del funzionamento dell'economia, hanno accesso a tutte le informazioni, i loro gusti sono fissi e affrontano un trade-off tra lavoro e tempo libero; l'ipotesi innovativa della teoria è che esiste una sequenza infinita di shock casuali all'esterno del sistema. Fra le varie motivazioni di tali shock, la più frequente è quella che essi siano causati dall'innovazione tecnologica. La teoria ci dice che quando avviene un shock positivo nell'economia, gli agenti sanno che c'è la possibilità di guadagnare e quindi decidono di lavorare maggiormente; quando invece ci sono degli shock negativi la scelta si sposta sul

tempo libero e la popolazione lavora meno; nel corso del tempo, perciò, la gente aumenta o diminuisce il numero delle ore lavorative. Tali adeguamenti costituiscono un elemento fondamentale del ciclo economico.

I risultati di questa impostazione sono però piuttosto modesti e il poco realismo delle ipotesi di base ha attirato feroci critiche alla teoria; in merito, l'opinione di Wickens, sembra essere la più equilibrata:

> Per molti il vantaggio dell'economia Keynesiana sull'economia ortodossa consiste nel presunto maggiore realismo dei suoi postulati. Tutti sappiamo che i mercati non sono completi, che la gente non è razionale, che prende decisioni miopi e non dispone di informazioni esaurienti, che un agente è diverso dall'altro e che un modello costruito su fondamenta microeconomiche non è aggregabile al livello delle variabili macroeconomiche convenzionali. Il fatto è che creare simili modelli è un obiettivo sostanzialmente irrealizzabile.

Ormerod (2003) tenta invece di costruire proprio un modello del genere, servendosi dei principi prima spiegati attraverso l'esempio delle formiche, e grazie ad esse, espone quali sono le caratteristiche innovative della sua teoria. Un'importante caratteristica del modello è il fatto di essere composto da attori individuali, come nel caso delle formiche; le oscillazioni e i cambiamenti del comportamento aggregato dipendono dalle interazioni dei singoli agenti. In questo modello del ciclo economico gli attori non conoscono l'avvenire e cercano di farsi un'idea di quanto può accadere nel periodo immediatamente successivo a quello in cui si trovano. Ciascuno degli agenti è leggermente diverso dagli altri, ma ognuno attribuisce molta importanza a quello che pensano gli altri e permette che le proprie decisioni vengano influenzate dall'opinione generale; gli agenti agiscono quindi in maniera non razionale e seguono delle regole empiriche per prendere le proprie decisioni in un mondo

caratterizzato dall'incertezza e dalla complessità. Uno dei risultati ottenuti è che i cicli economici sono endogeni e non si verificano per l'esistenza di shock esterni; ovviamente nella realtà tali shock esistono, ma il ciclo economico non per forza è causato esclusivamente da essi. In questo modello il ciclo scaturisce dall'interazione degli agenti economici presi in considerazione: le imprese. Queste determinano il livello di produzione per il periodo successivo a quello in cui si trovano in base a due elementi:

- l'inerzia e all'entità degli incrementi o dei tagli alla produzione effettuati nel periodo corrente;
- il livello generale di fiducia nel futuro e nei suoi cambiamenti.

Il modello del ciclo reale di Ormerod ottiene risultati migliori di quelli elaborati dall'economia ortodossa, ma qui non interessa descrivere minuziosamente le caratteristiche del modello, la sua formalizzazione o i risultati ottenuti; per questo si può consultare il libro di Ormerod. E' invece importante comprendere come molti dei problemi che affliggono le scienze sociali possono essere trattati in maniera differente da come si è fatto finora, cercando di dare maggiore realismo alle teorie e utilizzando strumenti, come le simulazioni, che già forniscono un valido aiuto alle scienze, ma con il loro ulteriore sviluppo potranno farlo sempre di più.

2.8 Hayek, l'ordine spontaneo e gli studi sulla complessità

In un contesto economico, quando si trattano temi riguardanti la complessità, è naturale parlare dell'opera dell'economista, psicologo e politologo Friedrich von Hayek.

La teoria dell'ordine spontaneo presenta diverse analogie con la teoria del caos e con quelle riguardanti la complessità, e Hayek sembrerebbe un precursore di queste nuove discipline.

Come si è già detto, la visione neoclassica presenta diverse difficoltà nell'elaborare teorie che spieghino fenomeni come il funzionamento delle organizzazioni, il ciclo economico, i mercati borsistici..., ed è praticamente impossibile effettuare delle analisi sulle scelte intertemporali degli agenti; la teoria dell'ordine spontaneo fu sviluppata per cercare una soluzione a queste problematiche. Hayek introdusse a tal fine il concetto di ordine che, a differenza dell'equilibrio, può coesistere con un certo grado di disequilibrio; l'ordine secondo Hayek è distinto in:

- Ordine spontaneo, che si forma per evoluzione ed è in grado di perpetuarsi e di autoriprodursi grazie al meccanismo endogeno che ne regola il funzionamento. Le strutture relazionali presenti in un ordine spontaneo non sono direttamente e facilmente comprensibili e il comportamento del sistema non è riconducibile alla volontà degli individui.

- Organizzazione, che è un ordine costruito artificialmente da una o più persone. Ne consegue che un'organizzazione è un tipo di ordine relativamente semplice con semplici strutture relazionali. In un'organizzazione ciascun individuo ha un suo ruolo e dei compiti da svolgere mentre in un ordine spontaneo gli individui possono perseguire i propri fini particolari. Un'organizzazione può essere usata per scopi semplici e limitati, mentre l'ordine spontaneo arriva là dove l'organizzazione non può arrivare.

Nel suo lavoro, Hayek accosta i processi di mercato al metodo scientifico. Le procedure scientifiche sono utili perché consentono di esplorare la realtà e di giungere con ciò stesso alla conoscenza di fatti nuovi e, a volte, imprevedibili. Allo stesso modo, il mercato consente di scoprire e seleziona chi è in

grado di svolgere al meglio un certo compito. Attraverso il meccanismo dei prezzi relativi, che include il loro continuo cambiamento, gli imprenditori si sforzano di percepire quali opportunità di investimento sono disponibili. Né il metodo scientifico né il mercato sono in grado di scoprire in anticipo quali fatti saranno scoperti dai ricercatori o dagli agenti economici.

Seguendo questa linea di ragionamento, Hayek sostiene che anche il monopolio o l'oligopolio possono a volte essere visti come il risultato della concorrenza più che una deviazione da essa. Tranne nei casi in cui essi sono il frutto di privilegi tesi ad impedire l'ingresso nella data industria di concorrenti, sia il monopolio che l'oligopolio sono perfettamente legittimi ed efficienti. Essi, infatti, derivano da abilità specifiche o dall'uso di fattori particolarmente adatti alla produzione di beni, oppure dall'aver colto prima di altri le opportunità offerte dal mercato. In questi casi, tuttavia, i vantaggi acquisiti risultano in genere temporanei. Un esempio banale, ma molto usato in economia, può aiutare a capire il senso di quanto detto: si pensi al caso di una merce che viene venduta ad un prezzo maggiore in una città rispetto ad un'altra; tralasciando il costo del trasporto, un commerciante potrebbe ottenere un facile guadagno comprando la merce dove costa di meno per rivenderla dove costa di più. In questo modo egli otterrebbe un profitto che sorge da una "opportunità di arbitraggio". Così facendo, tuttavia, egli farebbe aumentare la domanda nella città dove il prezzo è minore e farebbe aumentare l'offerta dove invece è maggiore facendo così variare i prezzi nelle due città; inoltre, altri commercianti lo imiterebbero e, alla lunga, scomparirebbe la differenza di prezzo.

Come si è detto, Hayek sostiene l'esistenza di una stretta analogia tra il metodo della scienza e i processi di mercato. Tuttavia, allo stesso tempo egli sottolinea un'importantissima differenza; i fatti scoperti dalla scienza sono fatti riproducibili. Essi, infatti, rappresentano *uniformità empiriche*. Al con-

trario, come mostra l'esempio del commerciante, i fatti scoperti nel processo del mercato sono fatti temporanei, perché dipendono da specifiche circostanze relative ad un certo tempo e ad un determinato spazio. Per Hayek, uno dei più gravi errori dei sostenitori delle economie pianificate consiste nell'aver confuso i due generi di fatti. I fatti e le conoscenze scientifiche possono essere concentrate in un'unica mente o in un'unica istituzione, mentre i fatti e le conoscenze del mercato sono disperse tra innumerevoli individui. Lo Stato non potrà mai concentrare in sé tutta la conoscenza rilevante e, di conseguenza, le economie pianificate sono destinate a fallire perché più inefficienti dei processi spontanei del mercato.

Per Hayek, il vantaggio della competizione risiede precisamente nella *diffusione delle conoscenze* che la teoria della concorrenza perfetta presuppone come già acquisite. Detto in altri termini, lo studio sugli stati di equilibrio cela l'importante ruolo che la concorrenza riveste come attività o processo che consente la scoperta di fatti nuovi da parte degli individui.

Hayek sottolinea anche che spesso le conoscenze sono tacite; gli individui non sanno di possederle e ciò che spinge i singoli a scoprirle è il meccanismo della concorrenza. E' grazie alle variazioni dei prezzi che gli imprenditori scoprono i differenti modi di ridurre i costi di produzione, modi che non erano loro noti prima che la concorrenza li spingesse a ricercarli. Il mercato concorrenziale è quindi una *procedura di scoperta*, un *processo dinamico* che favorisce la diffusione di informazioni. In un ordine concorrenziale di mercato un certo grado di disequilibrio è indispensabile per il suo funzionamento; è il disequilibrio che provoca il cambiamento dei prezzi che segnala agli agenti come modificare i loro piani di azione e che attiva l'incentivo alla scoperta e alla diffusione delle informazioni.

Per quanto detto, sembrerebbe esistere un collegamento piuttosto forte tra l'immensa opera di Hayek e gli studi sulla complessità. Esistono però

anche delle differenze non secondarie tra le due teorie. Kilpatrick (2001) sottolinea come si differenzia il pensiero tra i più autorevoli studiosi delle scienze della complessità (Arthur, Durlauff, Lane, 1997) ed il pensiero di Hayek:

> Heyek's theory of spontaneous order meshes meritoriously with complexity theorist, and would do well to study his works. However, his writings and those of complexity theorist appear to be qualitatively different. Believers of spontaneous order find a benevolent force creating a more efficient system than humans could devise by planning. Believers of complex adaptive behaviour, or complexity theory, find that forces may at time be malevolent and that the collective actions of humans may be necessary to return to optimum efficiency.

Secondo i teorici della complessità, quindi, non è definibile a priori quale sia la soluzione da preferirsi: i pianificatori potrebbero fare peggio del mercato, ma potrebbero anche non avere effetti o avere risultati migliori, mentre per Hayek il mercato ottiene sempre i migliori risultati possibili, e l'intervento di ipotetici pianificatori centrali lascerebbe, nel migliore dei casi, la situazione invariata.

2.9 La razionalità in economia

In qualsiasi scienza, il concetto di razionalità svolge un ruolo fondamentale. Non è certamente casuale che la scienza venga considerata l'attività umana per eccellenza razionale. Quando uno scienziato sostiene che una certa teoria è ben confermata - oppure che una teoria è preferibile ad un'altra teoria o che un fenomeno è ben spiegato dalle leggi scientifiche esistenti, e così via - fa implicitamente riferimento a criteri di razionalità scientifica. Parte del

compito della filosofia della scienza consiste precisamente nel chiarire tali criteri impliciti di razionalità.

L'economia non fa eccezione. Sin quasi dalla sua nascita, economisti e filosofi si sono impegnati a chiarire il tipo di razionalità scientifica che caratterizza l'economia; in termini approssimativi, la domanda principale che si pone questo genere di riflessione è così riassumibile: "con quali criteri è possibile valutare la correttezza di una teoria?". Si tratta, evidentemente, di un problema che si presenta in modo analogo in qualsiasi scienza, sia pure con le specificità proprie di ciascuna indagine.

Per altri versi, tuttavia, l'economia è una scienza peculiare. La razionalità a cui gli economisti fanno riferimento non è semplicemente la razionalità scientifica. In quanto scienza sociale, l'economia è interessata a studiare le azioni degli esseri umani, ed è proprio dell'economia giungere a tali spiegazioni attribuendo agli agenti un comportamento strettamente, o almeno in gran parte, razionale. Perciò, in economia il concetto di razionalità rappresenta anche *l'oggetto* della ricerca scientifica. Detto in altri termini, non avrebbe molto senso chiedersi in fisica quale razionalità esibisce un elettrone ma, al contrario, è scientificamente interessante chiedersi quale tipo di razionalità informa il comportamento di un imprenditore o di un consumatore. Ciò contraddistingue l'economia anche rispetto ad altre scienze umane. In psicologia, in sociologia o in antropologia il concetto di razionalità svolge un ruolo assai più modesto. Per chiarezza espositiva, è dunque utile distinguere, da una parte, la razionalità scientifica, e, dall'altra, la razionalità pratica, ovverosia la razionalità che è attribuita agli agenti, il cui comportamento è oggetto di indagine scientifica.

E' da sottolineare che l'economia non comunica all'esterno il rilievo che ha una delle questioni interne: quella su razionalità e ottimizzazione, nonché sul paradosso dell'agente rappresentativo, reso tanto complesso quanto neces-

sario per giustificare la microeconomia dei fenomeni macro (Terna, 1996). Il non comunicare tale discussione non significa però che non ci sia uno scontro, anche molto forte in alcuni casi, tra chi sostiene la necessità e la plausibilità dei modelli fondati sulla razionalità degli agenti economici e chi invece critica tele visione e cerca teorie alternative. Sempre da Terna (1996), leggiamo che

> ...definire il comportamento degli agenti economici in termini di razionalità e ottimizzazione consente di introdurre nei modelli ipotesi e formalismi tanto complessi quanto è necessario per giustificare la complessità del reale; ciò grazie alla costruzione di agenti (rappresentativi degli agenti reali, considerati uniformi) capaci di conoscenze illimitate e dotati di completa capacità di calcolo, doti entrambe necessarie per compiere scelte ottimizzanti; la complessità sta così negli agenti e non fuori di essi.

Il più famoso e sistematico razionalista è senza dubbio Ludwig von Mises. Per Mises tutta la scienza economica si basa sulla verità autoevidente che il concetto di azione umana include in sé il mirare consapevolmente ad uno scopo; non possiamo neanche pensare ad una azione umana (contrapposta alla mera reazione a stimoli) senza pensare con ciò stesso ad uno scopo intenzionalmente perseguito. Il criterio di controllo delle teorie economiche dovrebbe essere interamente deduttivo (Mises, 1976):

> Chi vuole attaccare un teorema [economico] deve, passo dopo passo, risalire indietro fino a quando raggiunge un punto in cui viene smascherato un errore logico [...]. Ma se questo processo a ritroso di deduzioni finisce con la categoria dell'azione senza aver scoperto un legame vizioso nella catena dei ragionamenti, il teorema è pienamente confermato.

Tuttavia, pochi economisti oggi aderiscono al razionalismo di Mises, in quanto essi probabilmente ritengono che la previsione (con tutte le sofisticate

indagini econometriche che ne conseguono) sia un valore troppo importante per essere trascurato.

Importante, nell'economia ortodossa, è la figura dell' "uomo economico". Sovente si afferma che l'uomo economico si caratterizza per un comportamento egoista e avido, in quanto la teoria economica presuppone che gli uomini siano sospinti solo dal proprio interesse personale. Per poter rispondere a questa affermazione si deve esaminare, in via preliminare, il significato e l'uso in economia del concetto di "razionalità".

In primo luogo, un comportamento razionale richiede che l'agente abbia un *insieme razionale di preferenze*. Un insieme razionale di preferenze rispetta fondamentalmente due assiomi. Il primo, detto di "completezza", dice che date due qualsiasi opzioni, A e B, ogni agente è in grado di dire se preferisce A, se preferisce B o se le due sono indifferenti. Il secondo assioma è quello della transitività, il quale dice che per qualsiasi opzione A, B e C, se un agente preferisce A a B e preferisce B a C allora l'agente preferisce A a C.

Si dirà dunque che un agente si comporta razionalmente se l'insieme delle preferenze è razionale e se non esiste un'opzione a lui disponibile che sia preferita a quella da lui effettivamente scelta. In campo economico, gli individui che "massimizzano la loro utilità" scelgono razionalmente, nel senso appena definito. Un individuo che agisce razionalmente, perciò, è semplicemente un individuo che sceglie coerentemente con le proprie preferenze. Come leggiamo in Terna (1996):

> L'aspetto negativo dell'ipotesi di razionalità olimpica, e della conseguente trattazione generalizzata dei problemi con lo strumentario della massimizzazione vincolata, sta nell'abuso del metodo. Certo non nel ritenere che i soggetti economici vogliano incrementare una qualche misura di risultato, implicita o esplicita, e comunque definita in termini di arco temporale (da immediato a

> totale). L'eccesso è quello di rifugiarsi nell'astrattezza di modelli che non hanno riferimenti realistici, ma che danno l'impressione o l'illusione di spiegare, mentre non sono che descrizioni non plausibili di una realtà che probabilmente è simultaneamente: più semplice, nei soggetti; più complessa, nelle loro interazioni.

E' importante comprendere che la teoria della razionalità non presuppone un comportamento egoistico in quanto esso non risiede nella soddisfazione delle preferenze, ma nella natura delle preferenze da soddisfare. Un altruista massimizza la propria utilità al pari dell'egoista e, allo stesso modo dal punto di vista della teoria della razionalità, un santo massimizza la propria utilità al pari di un sadico o di un misantropo. In termini sintetici, il principio di razionalità si basa esclusivamente sulla coerenza tra il comportamento ed un insieme di preferenze date.

E' evidente che l'uomo economico rappresenta una caricatura che non si trova mai nella realtà, tuttavia è considerato una "finzione" utile per comprendere i fenomeni di mercato. Come già scrisse Mill (1844):

> L'economia politica considera il genere umano solo in quanto dedito all'acquisizione e al consumo [...]. Non già che qualche economista sia mai stato così folle da supporre che gli uomini siano fatti in questo modo. Si tratta semplicemente del modo in cui la scienza deve necessariamente procedere.

La figura dell'uomo economico è giustificata perché si tratterebbe di una buona idealizzazione, capace di semplificare la teoria senza eccessive perdite nelle sue capacità esplicative e predittive. In alcuni ambiti, ciò appare ragionevole (si pensi alle decisioni che riguardano il consumo di beni privati). Tuttavia, in altri ambiti, come ad esempio nell'esame del funzionamento dei contratti, le caratteristiche dell'uomo economico non sono utili a spiegare la

realtà. I contratti, infatti, sono spesso incompleti, così come lo sono le informazioni dei contraenti. La fiducia tra i contraenti è perciò un valore morale economicamente importante, perché completa tacitamente la definizione di un contratto e sorregge il suo rispetto. Senza norme di condotta morali, le transazioni economiche avrebbero costi assai più elevati.

Una scuola di pensiero che si situa a metà tra chi utilizza la razionalità olimpica e che invece la respinge è quella della "razionalità limitata", che prende ispirazione dal lavoro dello psicologo e premio Nobel per l'economia Herbert Simon. Secondo Simon,

> La teoria classica della razionalità onnisciente è singolarmente semplice e affascinante [...]. Tutta la potenza predittiva proviene dall'aver caratterizzato la forma dell'ambiente in cui ha luogo il comportamento. L'ambiente, combinato con le assunzioni della razionalità perfetta, determina completamente il comportamento.

Per Simon vi sono almeno tre ordini di problemi nell'idea di una razionalità "onnisciente". Innanzitutto, molte decisioni avvengono in condizioni di incertezza sulle conseguenze delle varie alternative disponibili. Inoltre, sovente vi sono informazioni incomplete nello stesso insieme delle alternative disponibili. Infine, la massimizzazione di una grandezza richiede capacità di calcolo eccessive. In particolare, è l'ultimo aspetto ad assumere un'importanza decisiva nell'opera di Simon. Per questi motivi, Simon insiste che la ricerca deve spostarsi dall'ambiente esterno alla mente dei decisori. Con ciò si apre la strada allo studio dei processi cognitivi e alla psicologia. La razionalità della decisione non viene più valutata dal risultato della massimizzazione, ma dalla procedura seguita nel prendere le decisioni. La *razionalità procedurale* è dunque un'inevitabile conseguenza dei limiti della ragione umana. Più in concreto, Simon propone un modello di razionalità pratica basata sul concetto di *satisficing* (inteso come risultato soddisfacente). Ad esempio,

nel gioco degli scacchi, su cui Simon si è molto soffermato, date le difficoltà computazionali è fuori luogo immaginare che lo scacchista abbia di fronte un problema di ottimizzazione (ovverosia, la ricerca della mossa migliore). Piuttosto, egli si affida a regole euristiche. Esattamente come gli imprenditori, lo scacchista è un *rule-follower*, non un *maximiser*. Le regole lo aiutano a selezionare il numero di mosse da esaminare. La ricerca finisce quando egli, seguendo alcuni criteri di valutazione, trova una mossa soddisfacente (di nuovo non ottimale).

Una visione ancora diversa riguardo alle teorie economiche proviene da Mill (1844):

> Sovente la teoria economica viene contraddetta dall'esperienza. Tuttavia, se si ragiona correttamente, la teoria economica dovrebbe avere la stessa certezza epistemica delle premesse. Qual è dunque la ragione del frequente insuccesso delle previsioni economiche?.

La risposta si trova in una famosa tesi di Mill, secondo la quale le leggi economiche sono leggi di *tendenza*, i cui effetti possono essere contrastati da fattori di origine psicologica o sociologica. Dunque le leggi economiche sono vere sotto l'assunzione che non operino fattori contrastanti. Quando una previsione fallisce, ciò non significa che la teoria sia falsa. Piuttosto il fallimento della previsione indica che l'economista non ha preso in considerazione i fattori contrastanti che operano nelle circostanze date.

Tale teoria è stata però criticata in quanto, a volte, è necessario che i fattori contrastanti siano esplicitamente presi in considerazione dalla formulazione teorica, concedendo con ciò una modifica della stessa teoria. Il primo a sollevare questa obiezione è stato probabilmente Keynes, il quale osserva che

> ...le stesse modificazioni pratiche di cui parla Mill richiedono un trattamento scientifico e dovrebbero, perciò, avere un loro posto nella scienza.

Nella sua analisi, Friedman (1953) si sofferma sulla plausibilità delle assunzioni di base, sostenendo che quanto più una teoria è irrealistica, tanto più può essere utile. Si tratta del principio *as if*, che viene così spiegato da Friedman:

> In generale, più significativa è una teoria più irrealistiche sono le sua assunzioni. La ragione è semplice. Un'ipotesi è importante se spiega molto con poco, cioè se astrae gli elementi comuni e cruciali dalla massa di circostanze complesse e dettagliate che circondano i fenomeni [...]. Per essere importante, perciò, una teoria deve essere descrittivamente falsa nelle sue assunzioni.

Friedman, con queste parole, intende criticare una precisa posizione metodologica in economia. Data la già nota difficoltà di controllare le conseguenze predittive della teoria economica, molti economisti ritengono che debbano essere controllate non solo le conseguenze, ma anche le assunzioni della teoria. Ad esempio, tramite questionari si dovrebbe controllare se gli imprenditori massimizzano realmente il guadagno atteso. Per Friedman, al contrario, è irrilevante controllare le assunzioni della teoria. All'economista non interessa sapere come gli imprenditori giungano a prendere le loro decisioni. Ciò che importa è assumere che gli imprenditori agiscano *come se* massimizzassero il profitto e controllare la teoria confrontandola con la classe dei fenomeni che egli intende spiegare, ovverosia i complessivi fenomeni di mercato (prezzi, quantità, ecc...).

Secondo Barrotta (2003), la posizione di Friedman, presa alla lettera, non è sostenibile. Infatti, se si potesse affermare la verità della proposizione "gli

imprenditori cercano di massimizzare il profitto atteso" sarebbe possibile anche affermare la verità di tutte le sue conseguenze logiche, le quali riguardano anche i fenomeni di mercato. Dunque non è corretto affermare che la verità delle assunzioni sia irrilevante. Lo stesso si deve dire per l'affermazione che più le assunzioni sono irrealistiche più la teoria è empiricamente significativa. Con ciò, probabilmente Friedman intende sostenere che ogni buona teoria inevitabilmente trascura fattori causalmente rilevanti; tuttavia la necessaria economia di pensiero non va confusa con l'affidabilità predittiva di una teoria (Barrotta, 2003).

Da un punto di vista metodologico, la posizione di Simon è diametralmente opposta a quella di Friedman. A parere di Friedman è irrilevante controllare se realmente gli imprenditori cercano di massimizzare il profitto; in questo senso, i processi decisionali dell'impresa vengono posti al di fuori dei fenomeni di stretta competenza dell'economia, ma secondo Simon, il principio di massimizzazione non è affatto un buon strumento predittivo, ed è difficile sostenere il contrario se si pensa che spesso i modelli degli economisti non sbagliano solo l'entità della variazione di una grandezza, ma persino il segno.

Hahn (1994), in quello che definisce un suo *percorso attraverso la teoria economica*, critica duramente la visione di Friedman:

> Questi macroeconomisti [...] usano la matematica che si trova nei testi ma non sono affatto rigorosi nell'analisi e nella definizione dei casi a cui i loro esercizi sarebbero applicabili. Tuttavia sembra che l'aria scientifica del ragionamento matematico li abbia erroneamente indotti a credere di stare dicendo qualcosa di scientifico.
>
> Sulla base di queste riflessioni e preoccupato per quello che considero un cattivo uso del ragionamento matematico, sono giunto alla conclusione che la colpa non è della matematica. La colpa

> innanzitutto va attribuita a Milton Friedman e in secondo luogo al desiderio romantico di passare da "scienziati". La responsabilità è soprattutto della dottrina di Friedman del "come se" e di tutte le chiacchiere sulle virtù della "semplicità", "bellezza" ed "eleganza". Se l'economia fosse una scienza con un corpo dottrinale confermato da esperimenti rigorosi, forse si potrebbe dire qualcosa in favore dell'atteggiamento di Friedman. Ma non lo è e, per quanto ne so, nessuna teoria economica è mai stata falsificata in modo definitivo da esperimenti, figuriamoci poi dall'inferenza statistica. Invocare il fatto che ci sono teorie apparentemente paradossali come la meccanica quantistica accettate perché "funzionano", sarà giustificato quando la teoria degli economisti permetterà di fare previsioni corrette fino all'ottava cifra decimale, come fa la teoria quantistica. Fino ad allora, la diretta plausibilità delle nostre ipotesi resta un test che una teoria applicata al mondo "reale" deve superare.

L'analisi di Hahn sembra la più equilibrata in un mondo dove molte delle teorie vengono falsificate dalla realtà senza che gli autori cerchino di correggerle o migliorarle. La sua visione sull'economia cerca di riportare alla realtà studiosi fin troppo convinti della capacità delle proprie teorie di produrre previsioni corrette:

> L'idea è che il modo migliore di pensare all'economia è di considerarla come una specie di grammatica: un modo cioè di parlare coerentemente di eventi sociali complessi. Attualmente l'economia non permette molto più che previsioni qualitative. Tutti i discorsi sulle politiche "ottimali" pur se fatti sul serio non possono essere presi sul serio, anche se spesso costituiscono un buon punto di partenza per la discussione. Considero la teoria economi-

ca soprattutto un tentativo di "capire" piuttosto che di predire e prescrivere. Gli economisti possono portare un contributo molto utile al dibattito sulle scelte di politica economica ecc., ma la loro disciplina non fornisce risposte univoche, o tanto meno decisive e certe. Io sono sempre stato contento di questo ruolo modesto. Molti dei miei colleghi non lo sono, e molti si definiscono "scienziati", usando un termine che, preso nel suo significato comune, sembra piuttosto presuntuoso e prematuro. La nostra ambigua reputazione può apparire meritata date queste pretese eccessive.

Prosegue dicendo che

...certamente non sono il solo a credere che questi economisti, a forza di semplificare, abbiano distrutto la teoria economica in modo tale che molti, forse la maggior parte, dei problemi su cui la nostra disciplina ha indagato non possono nemmeno essere presi in considerazione. [...] Naturalmente ho anch'io le mie colpe. In gran parte del mio lavoro ho accettato la semplificazione della concorrenza perfetta e l'assenza di rendimenti crescenti. [...] Tuttavia, anche se non ho resistito alla comodità di assumere l'"ipotesi di concorrenza perfetta, penso di non avere mai ritenuto che i risultati raggiunti fossero effettivamente applicabili all'economia. Gran parte del lavoro è semplicemente nato da discussioni tra teorici.

Sono convinto però che se non ci si dimentica mai che siamo molte miglia lontani dal parlare di un'economia reale, l'ipotesi di concorrenza perfetta sia stata utile nel rispondere alla domanda puramente teorica: è possibile che un sistema economico in cui tutte le decisioni sono prese da agenti isolati sia un sistema ordinato senza che siano disponibili altri segnali informativi oltre

> ai prezzi? [...] Tuttavia ho continuato ad avvertire il bisogno di andare oltre questa semplificazione.

Hahn conclude così la sua analisi:

> Penso che la maggior parte della teoria elaborata nel dopoguerra sia decisamente valida e necessaria, ma l'ho sempre considerata una specie di "ouverture". La mia delusione è che io, e altri, abbiamo trovato tante difficoltà nel cominciare a scrivere l'opera vera e propria. Sono ancora più deluso del fatto che tanto pochi si accorgano che dobbiamo ancora cominciare.

2.10 Moving Away from the Holy Trinity

A conclusione di questa breve introduzione alle problematiche sulla complessità mi sembra giusto riassumere la questione riguardante i problemi della teoria economica con un paragrafo dedicato alle prospettive future di tali studi e il titolo del paragrafo (tratto da Colander (2003)) suggerisce la via alternativa alla visione ortodossa. La "sacra trinità" in questione è formata da razionalità, non sazietà ed equilibrio, e sono il fondamento dell'approccio analitico deduttivo. La moderna economia è in una fase di lento cambiamento e, a poco a poco, sta cominciando a cercare vie alternative a quelle classiche per spiegare i fenomeni economici. Ad esempio da Colander (2003) leggiamo che:

> Behavioral economics, which involves a challenge to the rationality and greed assumptions, is currently having the biggest impact on economics. But that, in my view, is simply a precursor of a larger change in method and analysis that will follow. That larger change involves the third pillar of economics-equilibrium.

> Accepting a behavioural foundation of economics requires one to give up equilibrium because the interactions become too complex to analytically solve for equilibrium. To overcome this problem economists are now developing agent-based models, in which researchers grow a model of the economy. They will create virtual economies, in which virtual agents are endowed with behavioural characteristics that will become more and more similar to real world agents.

Ma i modelli agent-based non sono gli unici sviluppi della moderna economia:

> The changes that are occurring can be seen in a variety of theoretical work, such as work in behavioural economics, evolutionary game theory, agent based modelling, experimental economics, and new institutional economics. Indeed, as I have argued elsewhere (Colander, Holt and Rosser, forthcoming) much of the work that is considered cutting edge theoretical work falls into the category of moving away from the holy trinity. [...] Specifically, I see the changes leading from a vision that sees economics as the study of infinitely bright agents in information rich environments to a vision of economics as the study of reasonably bright individuals in information poor environments.

Parte degli studi stanno cambiando la precedente visione di un'economia vista come un *"sistema semplice" molto complesso* in un *"sistema complesso" molto complesso*. I sistemi lineari (semplici) possono essere rappresentati in maniera analitica da un insieme di equazioni, mentre per un sistema non lineare (complesso) non è così.

Sistemi semplici e complessi differiscono nei loro micro-fondamenti: i sistemi semplici possono essere studiati partendo esclusivamente dai loro micro-

fondamenti, mentre i sistemi complessi possiedono proprietà "emergenti" e non possono essere compresi attraverso l'analisi dei singoli elementi che li compongono.

Cosa vuol dire avere proprietà emergenti? In Merlo (2002) leggiamo una definizione di Tinti (senza anno)

- è associato al funzionamento di un sistema complesso che evolve nel tempo;
- presenta contemporaneamente le seguenti proprietà:
  1. è una novità, un fenomeno che è descrivibile soltanto mediante un linguaggio qualitativamente diverso da quello utilizzato per descrivere il sistema e le sue componenti;
  2. ha origine dal basso all'alto, la sua formazione è dovuta esclusivamente alle interazioni locali tra le componenti del sistema;
  3. è imprevedibile, poiché le regole che descrivono il sistema negli stati locali presentano caratteristiche di non-linearità;
  4. non è scomponibile, è indipendete dall'esistenza e dalle proprietà delle singole componenti del sistema.

Dopo quanto detto è possibile prevedere un completo allontanamento degli studi economici dalle ipotesi dell'economia ortodossa? Colander (2003) risponde così:

> I predict that it will not, at least in the structure that we know it. The reason is that as economics moves away from its holy trinity assumptions, more and more cross specialization will occur. New hybrid fields will develop: psychoeconomics, neuroeconomics, socioeconomics, bioeconomics, and a variety of others.

The training, and tools of each will differ, pulling the profession apart. Without the holy trinity of assumptions holding it together, the profession will ultimately loose it coherence as a single field. It will exist, but as loose associations of different approaches, such as what one finds in the field of psychology today.

Bibliografia

[1] Arthur W.B., Durlauf S.N., Lane D.A., *The Economy as an Evolving Complex System II*, Addison-Wesley, 1997.

[2] Barrotta P., *Filosofia dell'economia*, Linee di ricerca, SWIF, 2003.

[3] Colander D., *The Complexity Revolution and the Future of Economics*, Middlebury College Economics discussion paper no. 03-19, 2003.

[4] Friedman A., *The methodology of positive Economics*, in Essays in Positive Economics, 1953.

[5] Hahn F., *Una retrospettiva intellettuale*, Moneta e Credito, no. 188, 1994.

[6] Kilpatrick H. E. Jr, *Complexity, spontaneous order, and Friedrich Hayek: are spontaneous order and complexity essentially the same thing?*, Complexity Vol. 6 N. 3, John Willey Sons, 2001.

[7] Mill J.S., *Sulla definizione di politica economica*, in Alcuni problemi insoluti dell'economia politica, a cura di S. Parrinello, Isedi, 1976.

[8] Merlo F., *L'impresa rifatta nel computer, applicazione del modello jES al caso BasicNet*, tesi di laurea, Facoltà di Economia, Torino, 2002.

[9] MISES, L., *The Ultimate Foundation of Economics*, Sheed Andrews and McMeel, 1976.

[10] ORMEROD, P., *L'economia della farfalla*, Instar Libri, 2003.

[11] PARISI D., *Simulazioni. La realtà ifatta nel computer*, Il mulino, 2001.

[12] SIMON H. A., *Causalità, razionalità, organizzazione*, Il Mulino, 1985.

[13] STEIN D.L., *Lectures in the sciences of complexity*, Addison-Wesley, Reading, MA, pp 1-13, 1997.

[14] TERNA P., *Economia e simulazione: una rivoluzione nel metodo*, Sistemi Intelligenti no. 3, 1996.

[15] TERNA P., *Hayek e il connessionismo: modelli con agenti che apprendono*, in G. Clerico e S. Rizzello (a cura di), Il pensiero di Friedrich von Hayek, Utet, 2000.

[16] WALDROP, MITCHELL, *Complexity: the emergence of science at the edge of order and chaos*, Simon and Schuster, 1992.

Capitolo 3

L'informatica nelle scienze e le simulazioni

3.1 La scienza dell'informazione

Con la parola "informatica" si indica il complesso di discipline e tecniche che concernono e permettono la trattazione automatica di tutte quelle informazioni che sono alla base delle nostre conoscenze e delle loro comunicazioni.

Il termine "informatica" è nato in Francia, nell'aprile del 1966, per un bisogno di sintesi, in quanto prima, per esprimere la stessa cosa, era necessario usare più parole e precisamente elaborazione automatica delle informazioni.

Il calcolatore elettronico è stato lo strumento di rottura che ha aperto nuove prospettive (come il cannocchiale lo fu per l'astronomia); è diventato il mezzo fondamentale per il trattamento delle informazioni, ma è bene distinguere lo sviluppo dell'informatica da quello dell'elettronica; il campo dei calcolatori non è l'elettronica e neppure la matematica, come il termine calcolatore potrebbe suggerire. Il campo dell'informatica riguarda la raccolta, l'analisi e l'elaborazione delle informazioni di ogni tipo, numeriche e non numeriche.

La scienza di tali processi è dunque la scelta dell'informazione o, appunto, informatica. L'Accademia di Francia ha formulato la seguente definizione di informatica:

> L'informatica è la scienza del trattamento razionale, particolarmente con macchine automatiche, delle informazioni considerate come supporto delle conoscenze umane e delle comunicazioni nei settori tecnici, economici e sociali.

E' da rilevare che la definizione non contiene il termine calcolatore ma parla, più in generale, di macchine automatiche, aggiungendo che l'utilizzazione di queste macchine non è indispensabile e che pertanto possono esistere anche altri modi di trattamento dell'informazione.

L'informatica, quindi, non è la scienza dei calcolatori ma quella dell'elaborazione delle informazioni.

In effetti l'informatica, intesa come elaborazione delle informazioni, preesiste al trattamento automatico. Le informazioni sono sempre state elaborate in qualche modo: a mano o con l'ausilio delle macchine e la tecnica di tale elaborazione trae le sue premesse da due fondamentali esigenze dell'uomo: quella di conoscere e quella di comunicare.

Oggetto dell'informatica è quindi l'informazione in sé, cioè come elemento di conoscenza e strumento di comunicazione tra gli uomini. L'informazione è un elemento di conoscenza ma, in quanto mezzo di comunicazione, deve essere idonea a esprimere concetti, scelte che determinano e influenzano l'azione dell'uomo. E come tale deve essere utile. L'informatica ha infatti per oggetto le informazioni che si prestano ad uno sfruttamento utilitario. Esse vengono elaborate per poter poi essere utilizzate e il loro utilizzo riguarda ogni campo del sapere.

Il calcolatore viene utilizzato specialmente per la sua proprietà di eseguì-

re calcoli molto rapidamente, ma i calcolatori non sono solo elaboratori di numeri e cifre, possono fare molto di più.

Si può dire che un nome più appropriato del calcolatore sarebbe elaboratore di simboli perché esso è in grado di elaborare in una grande varietà di modi, entità che sono la rappresentazione simbolica di qualunque grandezza. Questi simboli contengono informazioni in riferimento alle entità che rappresentano.

L'idea della elaborazione per simboli è fondamentale per poter capire la scienza delle informazioni. I simboli possono rappresentare le lettere dell'alfabeto, zone di una scacchiera, note musicali e pertanto con il calcolatore si possono programmare dimostrazioni di teoremi, partite a scacchi, traduzioni da una lingua all'altra, composizioni musicali ecc...

Con l'eccezione della matematica, non c'è altra scienza di uso così vasto come quella dell'informatica. Essa ha dato luogo ad una grande serie di nuove professioni e specializzazioni. La sua conoscenza è essenziale al fisico, all'ingegnere, al medico, al dirigente d'azienda, al sociologo, all'archeologo.

Se questa nostra epoca ha già visto le varie fasi di sviluppo della società industriale, vive ora quelle della società computerizzata, intendendo con questo termine la modificazione profonda introdotta in tutti gli aspetti della vita dall'uso dei calcolatori elettronici.

3.2 Sviluppo storico della elaborazione dei dati

Da quando l'uomo ha incominciato a contare sono stati escogitati strumenti capaci di eseguire le operazioni aritmetiche. Tali strumenti si indicano oggi come calcolatori numerici o digitali.

Il pallottoliere può essere considerato il più elementare calcolatore numerico e costituì l'unico strumento di calcolo delle prime civiltà, compresa quella romana, e rimase tale fino al grande rinnovamento scientifico dei secoli XVI

e XVII. Quei tempi furono caratterizzati da un intenso lavoro matematico, ingegneristico ed in generale scientifico, che richiedeva calcoli e introduceva nuovi procedimenti per operare.

Furono B.Pascal (nel 1642) e G.Leibniz (nel 1671) a costruire le prime macchine da calcolo. Nel 1671 il filosofo e matematico Leibniz scriveva:

> Non è cosa degna di uomini eccellenti perdere ore come schiavi per far calcoli che potrebbero essere tranquillamente affidati a qualcun altro se fossero usate dalle macchine.

La macchina costruita da Pascal, detta Pascalina, poteva eseguire l'addizione e la sottrazione mentre quella di Leibniz eseguiva anche la moltiplicazione e la divisione. Queste macchine contenevano, in embrione, gli stessi concetti su cui sono fondate le moderne calcolatrici; esse applicavano infatti un principio fondamentale nel calcolo matematico: il riporto automatico.

Ma il calcolo automatico nel suo significato moderno, non è questo. Esso consiste nell'affidare a una macchina l'esecuzione, in blocco, di tutte le operazioni che risolvono un problema, cioè l'esecuzione di quello che si chiama un processo di calcolo. Il primo tentativo di costruire strumenti di calcolo automatico fu intrapreso nel 1792 dall'inglese Charles Babbage. Egli progettò una macchina capace di eseguire qualsiasi operazione aritmetica e capace di organizzare le operazioni necessarie per risolvere un problema aritmetico, ma non poté mai realizzarla a causa delle difficoltà tecniche che presentava la sua costruzione. Fu nel 1880 che si presentò pressante l'esigenza di costruire dei sistemi per l'elaborazione dei dati mediante macchine; in quell'anno negli Stati Uniti si svolse il decimo censimento decennale della popolazione che contava allora cinquanta milioni di abitanti. Poiché i dati raccolti venivano riportati manualmente su schede e poi classificati e selezionati più volte, furono necessari sette anni per ottenere i risultati. Lo statistico americano Hollerith, riprendendo il concetto della scheda perforata di Babbage, pensò di

registrare i dati di uno stesso individuo o nucleo familiare su un cartoncino di carta. I dati venivano riportati su ogni scheda mediante perforazioni e la lettura di tali schede avveniva meccanicamente per mezzo di macchine elettromeccaniche che interpretavano ed elaboravano i dati letti.

Con la rapida evoluzione dell'elettronica fu risolto il problema del calcolo automatico; il primo calcolatore elettronico veramente universale fu progettato e costruito nel 1946 da Eckert e Mauchly presso l'Università della Pennsylvania e prese il nome di ENIAC. Esso è considerato il capostipite dei calcolatori di prima generazione. Negli anni sessanta inizia l'era della seconda generazione ed in questo periodo si ebbe, grazie a J. Von Neumann, l'idea di memorizzare all'interno del calcolatore non solo i dati ma anche il programma trattando le istruzioni come dati. E così l'idea della macchina cambia: da computer, o macchina per calcolare, diventa ordinateur in francese, ed elaboratore in italiano e l'accento si sposta dalle sue capacità aritmetiche a quelle logiche.

Con l'evolvere della tecnologia gli elaboratori elettronici si sono ulteriormente evoluti ed oggigiorno siamo ormai nell'era della quinta generazione. Attualmente si parla di ingegneria del software, di banche dati e soprattutto di intelligenza artificiale. (AI: Artificial Intelligence).

3.3 L'informatica orientata ai problemi aziendali

Prendendo spunto da Bussolin (1975) sull'utilizzo dell'informatica in azienda, si può notare come trent'anni fa fosse difficile applicare le conoscenze informatiche alle attività aziendali:

> Molte imprese, hanno ritenuto sufficiente la formazione di uno specialista: il programmatore. Sul mercato non era facile trovare

> degli esperti in analisi e progettazione di sistemi informativi orientati al computer e tanto meno esistevano delle scuole capaci di formare questo personale. L'unica via era, ed è ancora in parte, la formazione interna all'azienda attraverso l'esperienza professionale, formazione per altro incompleta in quanto mancava, e purtroppo manca ancora oggi, una divulgazione delle conoscenze acquisite, degli esperimenti condotti e delle applicazioni realizzate. L'aver ritenuto sufficiente, sul piano operativo, la preparazione di specialisti programmatori, ha voluto dire per molte imprese sprecare molte risorse (tempo, investimenti in uomini e in hardware).

Sempre da Bussolin (1975), si apprende che uno dei principali strumenti utilizzati all'epoca in campo aziendale era il linguaggio di programmazione COBOL, inteso quale strumento atto a tradurre i problemi amministrativi in programmi da far eseguire all'elaboratore elettronico.

> Questo linguaggio è stato appositamente costruito per risolvere i problemi aziendali. [...] Attualmente non si può dire che già tutte le aziende adottino il COBOL. Quando però un'azienda sente anche solo l'esigenza di riscrivere un programma per eseguirlo su nuove macchine, il ricorso al COBOL è quasi sempre la via più rapida e seguita.

La tecnologia si è però sviluppata molto velocemente e oggi anche la più modesta azienda, organizzazione o singolo individuo può permettersi di disporre di un proprio sistema informativo e di una "piccola" base di dati. E' però noto che una delle più diffuse applicazioni delle basi di dati riguarda certamente il mondo aziendale; la gestione informatica di dati come gli *articoli*

prodotti, gli *ordini* dei diversi *clienti* e le relative *fatture* emesse permette di evitare lentezze e ridurre i rischi di un lavoro manuale, ma soprattutto consentono di accedere alle informazioni rilevanti in tempi molto ridotti.

Un esempio pratico di quanto detto riguarda il sistema di prenotazione delle compagnie aeree, servizio che è stato uno dei primi ad essere stato informatizzato e a coinvolgere l'uso delle reti telematiche. Gli utenti, in modo imprevedibile e da diverse parti del mondo, chiedono di prenotare, spostare o rinunciare a uno o più voli. Il sistema deve essere in grado di accettare, respingere o mettere in lista di attesa le prenotazioni. Questa realtà è interessante perché il servizio che offre all'utente, specie nelle ore immediatamente precedenti un imbarco, deve essere fornito in tempo reale e in modo assolutamente affidabile. In particolare occorre gestire correttamente le richieste quasi simultanee di due prenotazioni quando rimane, ad esempio, un unico posto su uno stesso volo. Se tutto ciò è dato oggi per scontato, non era, fino a vent'anni fa così, ed ha richiesto numerosi studi e ricerche sui modi migliori per arrivare alla gestione di questo come di altri servizi che utilizzano le reti telematiche.

In Simon (1977) si può leggere come gli sviluppi futuri dell'informatica non fossero facilmente prevedibili e portassero gli studiosi a chiedersi come si sarebbe evoluto il mondo grazie ad essi:

> ...nel prossimo futuro - forse nel giro di una generazione - disporremo della capacità tecnica per sostituire l'uomo con le macchine nello svolgimento di qualsiasi funzione organizzativa. Entro lo stesso periodo disporremo anche di una vasta teoria, verificata empiricamente, relativa ai processi cognitivi e all'interazione tra questi ultimi e le emozioni, gli atteggiamenti e i valori umani.
>
> Tuttavia, dicendo che disporremo di queste capacità tecniche non si indica minimamente in che modo le utilizzeremo.

Anche dal punto di vista del mercato del lavoro le discussioni sulle nuove tecnologie erano accese:

> ...indipendentemente dagli effetti passeggeri dell'automazione, le risorse umane della società saranno integralmente e sostanzialmente impiegate. Parlando però di piena occupazione non bisogna necessariamente intendere una settimana di quaranta ore e questo perché la distribuzione della capacità produttiva tra la produzione di ulteriori beni e servizi e l'incremento del tempo libero può forse continuare a cambiare, come è avvenuto in passato. Per piena occupazione bisogna invece intendere che la stragrande maggioranza degli individui adulti avrà la possibilità di lavorare e che, attraverso i salari e gli altri dispositivi di distribuzione del reddito, il prodotto dell'economia sarà ampiamente ripartito tra le famiglie.

3.4 Le simulazioni

Nella lingua italiana la parola simulazione ha diversi significati ed è importante che si capisca fin da subito in che accezione è qui utilizzata.

Nell'edizione del 1988, il dizionario Garzanti fornisce tre possibili significati:

1. Il simulare, l'essere simulato, finzione; riproduzione ai fini sperimentali delle condizioni in cui si verifica un fenomeno

2. Situazione che si verifica quando nel compiere un negozio giuridico vi è divergenza preordinata tra la volontà e la dichiarazione che se ne fa, in base a un accordo tra le parti o fra queste e un terzo

3. Analisi di un fenomeno, di un processo o di un sistema effettuata attraverso la costruzione di un modello matematico che lo simuli, sviluppato in genere mediante un elaboratore elettronico.

E' interessante notare come nessuna delle definizioni fornite corrisponda a quella che qui interessa. Si potrebbe pensare che su un dizionario più recente ci sia un'analisi più approfondita, ma invece nel De Agostini del 1999 delle tre definizioni manca quella riguardante il modello matematico. Parisi (2001) coglie bene il problema definendo le simulazioni come un nuovo strumento per esprimere le teorie scientifiche che si è aggiunto agli strumenti tradizionali con cui la scienza cerca di conoscere e capire la realtà. Le simulazioni vanno diffondendosi in tutte le discipline scientifiche ma sono ancora una novità e perciò la loro natura non è ancora ben compresa.

La terza definizione del dizionario, pur essendo quella che più si avvicina al significato che a noi qui interessa, per noi è incompleta. Il modello che simula il fenomeno o il processo in questione non per forza dev'essere un modello matematico.

Per esprimere le teorie scientifiche è possibile utilizzare il linguaggio, per cui la teoria viene formulata verbalmente come normalmente avviene in psicologia o filosofia; una seconda possibilità è data dall'utilizzo di schemi grafici, ed infine il metodo più comune è proprio quello matematico, caratterizzato da numeri ed espressioni numeriche, che viene impiegato soprattutto nelle scienze naturali.

Il comune denominatore di questi metodi è che tutti esprimono le teorie scientifiche tramite simboli, e qui sta la prima grande innovazione delle simulazioni tramite computer: queste ultime infatti non esprimono le teorie usando i simboli, ma usando un programma di computer. In particolare, il programma *è* la teoria. Questo perché le simulazioni non spiegano la realtà: la riproducono, e per farlo c'è bisogno che il programma sia costruito sulla

base dei concetti e delle cregole che formano una teoria e, cosa ancora più importante, tali concetti devono essere espressi in maniera chiara e completa, senza la possibilità di ambiguità o piccole inesattezze che le teorie formulate in altra maniera possono avere.

E' per questo che le teorie espresse in forma tradizionale, cioè con le parole, le equazioni, gli schemi grafici, si limitano a spiegare la realtà. Le teorie espresse come simulazioni la riproducono. Le simulazioni ci fanno capire la realtà *ricreando la realtà nel computer.*

Sempre secondo Parisi (2001), se il programma al computer riproduce fenomeni che avvengono nella realtà, allora vuol dire che le idee che lo studioso ha usato per costruire il programma riescono a cogliere le cause, i meccanismi e i processi che stanno dietro ai fenomeni e li spiegano.

Simon (1988) sottolinea che la simulazione, quale tecnica per capire e prevedere il comportamento dei sistemi, è più antica, ovviamente, dei calcolatori numerici. Il bacino sperimentale e il tunnel aerodinamico sono sistemi validi per studiare il comportamento dei grandi sistemi che riproducono il piccolo ed è noto che la legge di Ohm fu suggerita al suo scopritore dalla sua analogia con fenomeni idraulici elementari. Tuttavia, grazie al suo carattere astratto e alla sua generalità di manipolatore di simboli, il calcolatore numerico ha notevolmente esteso la gamma di sistemi di cui è possibile imitare il comportamento.

Una questione interessante è se la simulazione può risultare utile quando in partenza sappiamo poco delle leggi naturali che governano il comportamento del sistema interno. Anche a questa domanda si deve rispondere affermativamente (Simon, 1988). Anzitutto si deve sottolineare che solo raramente gli studiosi si preoccupano di spiegare o prevedere i fenomeni in tutti i loro particolari; in genere si interessano solo di alcune proprietà tratte dalla complessa realtà. Per esempio, nessuno pensa che un satellite lanciato

"simuli" la luna o un altro pianeta. Esso obbedisce semplicemente ad alcune leggi fisiche, che interessano solo la massa inerte e gravitazionale, astratta da tutte le atre sue proprietà. Più si è disposti ad astrarre dei particolari di una serie di fenomeni, più facile è la simulazione di tali fenomeni. Inoltre, non è necessario conoscere, o indovinare, tutta la struttura interna del sistema ma solo quella parte di esso che ha un'importanza critica per l'astrazione. E' una fortuna che sia così, altrimenti non sarebbe stata realizzabile quella strategia discendente (top-down) che, nel corso degli ultimi tre secoli, ha consentito di costruire le scienze naturali.

Non solo, la realtà virtuale mostrerà anche le potenzialità del mondo reale che non sono ancora state espresse. Agendo sulla simulazione si possono scoprire gli effetti degli interventi praticati e le soluzioni a problemi sui quali non ci si era soffermati in precedenza.

Se analizziamo il comportamento delle organizzazioni possiamo notare che nessuno è in grado di risolvere problemi altamente complessi in modo totalmente razionale; le soluzioni possibili, quindi, si riducono per due principali motivi:

- perché si scelgono le soluzioni che sembrano più giuste;
- perché quella è la prassi e si è sempre fatto così.

La prima motivazione è corretta ma da migliorare, mentre la seconda è sbagliata e quindi da superare: la simulazione permette di andare verso queste direzioni. Attraverso una simulazione, infatti, si dispone di un sistema computabile su cui provare i cambiamenti e le scelte necessarie per migliorare i processi complessi, come se si lavorasse sulla realtà, tenendo conto in primo luogo della conoscenza effettivamente disponibile nel contesto studiato. Questo può essere fatto perché, come sostiene Parisi (2001), le simulazioni sono laboratori sperimentali virtuali. Nel laboratorio sperimentale reale, cioè

nella stanza fisica con le apparecchiature, le provette, ecc.., lo studioso osserva i fenomeni in condizioni controllate e manipola le condizioni che implicano il verificarsi dei fenomeni osservando le conseguenze delle sue manipolazioni. In questo modo può ottenere un maggior numero di informazioni sulla realtà rispetto ad una semplice osservazione dei fenomeni nel loro contesto naturale.

Così come avviene in un laboratorio reale, i fenomeni *virtuali* che emergono dalla simulazione, cioè i risultati della simulazione, si verificano in condizioni controllate dallo studioso. Inoltre, come nel laboratorio reale, il ricercatore può manipolare le condizioni che determinano o influenzano il verificarsi di questi fenomeni virtuali, può intervenire sui fattori e le variabili incorporate nella simulazione, può modificare il valore quantitativo dei parametri, e osservare ogni volta quali sono i risultati di questi suoi interventi. Tutto ciò permette non solo di sfruttare le possibilità offerte da un laboratorio reale, ma in più la libertà con cui lo studioso può controllare e manipolare le condizioni e le variabili in una simulazione è molto maggiore di quella con cui può fare queste cose nel laboratorio reale.

3.5 I vantaggi delle simulazioni

In cosa le simulazioni portano dei vantaggi rispetto agli altri metodi di indagine scientifica? Innanzitutto c'è da dire che già nella fase di costruzione del modello ci sono degli aspetti positivi: il ricercatore, infatti, deve elaborare con chiarezza la teoria che dovrà testare. La ragione è che altrimenti il programma, scritto con un linguaggio informatico, risulterà incompleto o incoerente; presentando questi problemi il programma non potrà neppure essere avviato nel computer. Se una teoria interpretativa si presenta come una simulazione, i suoi concetti non possono non essere chiari, espliciti e univoci. Il significato dei concetti è tradotto interamente nel codice della simulazione.

Nelle scienze naturali le teorie sono sottoposte alla verifica empirica; le

predizioni tratte dalla teoria sono cioè confrontate con i fatti direttamente osservati. Se questo confronto mostra che vi è corrispondenza, la teoria si ritiene verificata, mentre se mostra una discordanza la teoria viene considerata falsificata dalla realtà empirica. Quando una teoria è espressa come simulazione e questa gira nel computer, i risultati della simulazione sono le predizioni derivate dalla teoria (Parisi, 2001). Se una simulazione dà certi risultati, questi sono predizioni la cui discendenza dalla teoria è certa. Ciò significa che la teoria del ricercatore è esatta, poiché il modello da lui creato ripropone ciò che avviene nella realtà. Inoltre, con l'uso delle simulazioni come laboratori sperimentali virtuali, capaci di generare una enorme massa di risultati nelle diverse condizioni manipolate dall'utente, le simulazioni rendono molto più ampia, differenziata e completa la possibilità di verificare le teorie dato che la teoria-simulazione mostra tutto il range di soluzioni di cui è capace. Questo irrobustisce il collegamento tra teorie e realtà empirica che è così importante per la scienza.

A tutto ciò si deve aggiungere che con le simulazioni si possono studiare fenomeni che per ragioni puramente fisiche non possono essere studiati in un laboratorio reale. I fenomeni fisicamente troppo grandi, o che durano troppo a lungo, infatti, non si possono riprodurre in un laboratorio. Alcuni esempi sono i fenomeni che riguardano intere regioni della Terra, o che durano anni, secoli o millenni. Nessuno di questi ostacoli, legati alle caratteristiche della realtà fisica, vale per le simulazioni. Ancora, attraverso le simulazioni possono essere studiati fenomeni avvenuti in passato ma oggi non più presenti e che quindi non possono essere osservati allo stesso modo dei fenomeni presenti.

Un ulteriore vantaggio è che una simulazione riproduce, semplificandolo, il mondo reale (Parisi, 2001); ma se una simulazione riproduce ciò che avviene nella realtà, allora si può chiedere a una simulazione di riprodurre non solo il mondo che esiste, ma anche mondi che non esistono ma che potrebbero esi-

stere. Partendo da alcuni dei principi che governano il mondo reale, possono essere studiate anche situazioni che non si sono mai verificate, ma di cui si vuole conoscere le possibili conseguenze.

Si deve infine sottolineare il fatto che il metodo sperimentale richiede la ripetibilità dei fenomeni, cioè la possibilità di riprodurli in forma identica. Ritornando al concetto di complessità (ampiamente trattato nel capitolo 1), questo si può fare per i sistemi semplici, ma non per i sistemi complessi che tendono ad essere molto sensibili alle condizioni iniziali, per cui differenze minime nelle condizioni iniziali determinano sviluppi anche molto diversi, e quindi la non riproducibilità di un fenomeno in forma identica. Senza il computer, e senza le simulazioni, probabilmente lo studio dei sistemi complessi non sarebbe possibile.

3.6 Le critiche "infondate" delle simulazioni

Le simulazioni ricevono spesso critiche; alcune di esse sono fondate, mentre altre sono basate su convinzioni errate o incomplete. E' quindi necessaria una panoramica sia delle une che delle altre. Le critiche infondate sono riassumibili nelle quattro seguenti:

1. le simulazioni sono troppo semplificate rispetto alla realtà;
2. le simulazioni non dicono nulla di nuovo;
3. le simulazioni vogliono riprodurre la realtà, ma come è possibile riprodurla se, come capita spesso, ancora non la si conosce bene?;
4. le simulazioni sono opache, sono poco trasparenti.

Si procederà ora nell'analisi dettagliata di tali critiche.

3.6.1 Troppo semplificate rispetto alla realtà

Questa critica si basa su una mancata comprensione di quello che è una simulazione (Parisi, 2001). Una simulazione è innanzitutto una teoria espressa sotto forma di programma di computer. Tutte le teorie della scienza semplificano la realtà per fare capire i fenomeni osservati. Anzi le teorie sono utili proprio perché semplificano; in questo modo si cerca di cogliere l'essenziale della varietà dei fenomeni osservabili, e capire consiste proprio nel cogliere l'essenziale. Non ha senso quindi criticare le simulazioni in quanto sono semplificazioni della realtà dato che tale critica andrebbe rivolta a tutte le teorie della scienza. Il problema è invece quello di stabilire se una simulazione fa le semplificazioni giuste, cioè se quello che è incluso nelle simulazione è quello che spiega i fenomeni che interessano, mentre quello che ne resta fuori è irrilevante.

Per un altro aspetto questa critica ha però una sua giustificazione. Le simulazioni sono prima di tutto teorie, ma non sono soltanto teorie. Sono anche strumenti che pretendono di riprodurre i fenomeni della realtà, non soltanto di spiegarli come hanno fatto fino ad oggi le teorie (Parisi, 2001). Di conseguenza si può capire che si possano criticare le simulazioni quando riproducono la realtà in modo troppo semplificato. Quello che è interessante è che con l'ulteriore evoluzione tecnologica, le simulazioni hanno la possibilità di diventare più complesse, più ricche e più realistiche, mentre le teorie espresse nei modi tradizionali resteranno necessariamente semplici.

3.6.2 Non dicono nulla di nuovo

Un'altra critica che viene rivolta alle simulazioni è che non dicono nulla di nuovo. Per simulare un fenomeno lo si deve conoscere, ma se già lo si conosce, a che serve simularlo?

Innanzitutto si deve sottolineare che una simulazione non ripropone sola-

mente ciò che avviene nella realtà, ma possono emergere fenomeni (simulati) nuovi, diversi da quelli sui quali ci si era basati per costruire la simulazione. Addirittura alle volte i risultati sono sorprendenti, poiché incompatibili con l'interpretazione della realtà data fino a quel momento; in ogni caso si ha sempre interesse a osservare i risultati di una simulazione, soprattutto per la possibilità di manipolare condizioni e variabili per osservarne gli effetti.

Tutto ciò porta ad un problema critico: come può la simulazione dire qualcosa che non si sa già? Normalmente la risposta a questa domanda è che non può. In realtà, esiste un parallelismo tra due affermazioni che capita di sentire frequentemente a proposito dei calcolatori e della simulazione:

- una simulazione non può valere al di là delle ipotesi su cui si fonda;
- un calcolatore può fare solo quello per cui è stato programmato.

Non pare possibile discutere nessuna delle due affermazioni, in quanto sembrano tutte e due vere. Malgrado la correttezza di entrambe, la simulazione può riuscire a dire delle cose che in precedenza non si sapevano. Simon (1988) individua due modi grazie ai quali la simulazione può insegnare qualcosa di nuovo: uno ovvio, l'altro forse un po' meno. Il punto ovvio è che anche quando si disponga di premesse corrette, può essere difficilissimo scoprire quello che implicano. Secondo Simon (1988),

> Ogni ragionamento corretto è un enorme sistema di tautologie, ma solo Dio può approfittare di questo fatto. Qualsiasi altra persona deve faticosamente e fallibilmente cavar fuori le conseguenze dagli assunti di base.

Ma soprattutto le simulazioni ci informano di qualcosa che prima non sapevamo, e cioè ci informano sull'effettivo contenuto empirico delle nostre teorie. Se una simulazione riproduce i fenomeni empirici della realtà median-

te quelli simulati, questo significa che la teoria che spiega questi fenomeni della realtà, così come tale teoria è espressa e incorporata nella simulazione, è corretta (Parisi, 2001). Quindi, a una simulazione non dobbiamo chiedere necessariamente di produrre fenomeni empirici nuovi, diversi da quelli che abbiamo usato per costruire la simulazione. Una simulazione è importante anche se riproduce soltanto i fenomeni empirici già noti. Il fatto che li riproduca significa che la teoria incorporata nella simulazione è una teoria corretta.

3.6.3 Non possono riprodurre la realtà

Le simulazioni servono a riprodurre la realtà, ma come possono a riprodurla se, come capita spesso, ancora non la si conosce bene? La risposta a questa obiezione è che se le cose stessero così, se si dovesse aspettare di conoscere completamente qualcosa per simularlo, le simulazioni non servirebbero più.

Prima di tutto le simulazioni sono teorie, e le teorie servono per andare al di là dei fatti osservati e cominciare a capire perché e in che modo succedono determinati fenomeni e che cosa li causa. Questo è il compito delle teorie nella scienza, ed è anche il compito delle simulazioni.

In secondo luogo, le simulazioni possono suggerire l'esistenza di altri fenomeni empirici della realtà, diversi da quelli che già si conoscono, e quindi possono guidare la ricerca empirica reale per scoprire se questi fenomeni nuovi suggeriti dalle simulazioni esistono effettivamente.

3.6.4 Le simulazioni sono "opache"

Un'ultima critica rivolta alle simulazioni è che se anche si riuscisse a simulare un fenomeno nel computer in tutti i suoi dettagli, questo non significherebbe affatto che quel fenomeno è stato compreso meglio. Le simulazioni sono "opache", poco trasparenti, e non è detto che si riesca a capire a cosa siano dovuti i risultati che emergono.

In realtà una simulazione appare come qualcosa di opaco solo a chi la osserva passivamente da fuori. Il ricercatore può intervenire e modificare ogni aspetto della simulazione per vedere che effetti risultano da queste sue manipolazioni osservando non solo i fenomeni ma anche quello che sta dietro ai fenomeni; questo può aiutarlo a capire come la simulazione funziona e cosa determina i risultati che si osservano.

3.7 Le critiche "fondate" delle simulazioni

Dopo quanto detto finora, non si deve però credere che le simulazioni siano immuni da problemi. In realtà il difetto delle simulazioni è sovente nel loro uso più che nello strumento in se stesso; tutto questo inizio di capoverso deve essrerivisto per l'italiano]. Spesso, infatti, chi usa le simulazioni come metodo di ricerca tende a dare un eccessivo peso nello stabilire se la teoria spiega effettivamente certi fatti empirici. Sarebbe invece meglio prestare maggiore attenzione alle predizioni derivate da una teoria.

Se la teoria è espressa come una simulazione, i risultati della simulazione sono le predizioni derivate da essa; perciò diventano chiare tutte le predizioni empiriche di una teoria, non solo quelle desiderate. Spesso ci si accontenta di costruire la simulazione, di osservarne e analizzarne i risultati, ma non ci si preoccupa molto di stabilire qual è la corrispondenza tra i risultati della simulazione e la realtà empirica. Questo dovrebbe essere invece l'aspetto che merita più attenzione, in quanto implica la correttezza o meno della teoria.

Per quanto riguarda il problema delle semplificazioni effettuate dalle simulazioni, è importante, come si è già accennato, che le semplificazioni siano quelle giuste, e che cioè non includano nella simulazione gli aspetti irrilevanti dei fenomeni empirici che si vogliono studiare, ma solo quelli critici. Un limite di molte simulazioni attuali è che a questa distinzione viene dedicata un'attenzione insufficiente.

Questi sono alcuni dei rischi ai quali il ricercatore-simulatore può andare incontro, ma che possono essere evitati utilizzando un rigoroso metodo scientifico nella preparazione degli esperimenti condotti con le simulazioni. Il rischio diventa che il pensiero (auto-ironico) di Pryor (2000) diventi realtà. Pryor racconta di un autore sconosciuto che, nel 2028, dopo la caduta di un asteroide sulla terra, ritrova un libro sulla complessità degli anni 90. Dopo una attenta lettura, l'autore catalogherà il ritrovato come un tipico libro su un argomento in cui, i quegli anni, quasi tutti i lavori non contengono applicazioni empiriche reali, ma tuttalpiù qualche interessante aneddoto.

3.8 La simulazione in ambito economico

Esistono almeno due campi in cui la simulazione al calcolatore risulta necessaria, o comunque molto utile:

1. nei modelli dotati di capacità di apprendimento, e cioè in cui gli agenti sono capaci di modificarsi e adattarsi in base all'interazione con l'ambiente circostante.

2. negli studi sull'emergenza di fenomeni particolari, quali ad esempio la cooperazione, simulando la crescita di società. Fenomeni emergenti vengono osservati in sistemi dinamici, i quali si modificano per rispondere ai cambiamenti sopravvenuti nell'ambiente, contribuendo, così, a modificare l'ambiente stesso.

In questi tipi di studi, l'analisi meramente descrittiva di un modello non potrebbe sostituire la simulazione al calcolatore, in quanto gli strumenti alternativi non sono in grado di spiegare fenomeni complessi. Le simulazioni sono quindi un valido ed interessante strumento per lo studio dell'economia: offrono infatti uno strumento che permette di studiare insieme i singoli individui e i fenomeni collettivi che derivano dalla loro interazione.

Nella definizione data da Parisi (2001), la simulazione ad agenti è un particolare tipo di simulazione che cerca di riprodurre fenomeni collettivi facendo interagire tra di loro un certo numero di agenti, i quali hanno determinate regole, ossia reagiscono in modo determinato agli stimoli che ricevono. Questo tipo di simulazione risulta di particolare utilità nello studio dei fenomeni sociali. Il comportamento del singolo agente è in grado di influenzare i comportamenti di altri agenti e dall'interazione degli agenti emerge la complessità. Riproducendo nella simulazione determinati tipi di agenti, a cui si danno sembianze il più possibile fedeli a quelle degli agenti che si intendono studiare, è possibile affrontare lo studio di numerosi sistemi reali, composti da numerosi individui interagenti con il vantaggio, sottolineato da Terna (1996), di poter verificare in qualsiasi momento le caratteristiche degli agenti e i loro comportamenti.

Uno dei più importanti effetti dello studio dei modelli economici complessi è stato il cambiamento nel metodo e nella direzione della ricerca. Nei nuovi modelli vengono descritte soltanto le interrelazioni a livello locale tra i singoli agenti, mentre i comportamenti aggregati o le strutture emergono dall'auto-organizzazione, invece che essere semplicemente imposti o accettatati. Ciò che emerge in un modello, non è quasi mai la semplice somma di quello che avviene a livello individuale; ciò è in contrasto con il metodo dei modelli ad agenti rappresentativi, nei quali la somma degli individui equivale all'aggregato. Ancora Terna (2002c) spiega che la costruzione di modelli di simulazione fondati su agenti consiste nello studio dell'azione dei soggetti economici in termini cognitivi e nella comprensione delle conseguenze delle caratteristiche degli agenti che si scoprono nel tempo, tenendo conto di azioni, reazioni e interazioni tra i singoli agenti.

In un qualsiasi modello di simulazione ad agenti è possibile effettuare la distinzione tra tipologie di agenti e tipologie di ambienti in cui questi operano.

Vi possono essere agenti con mente, che sono agenti che possiedono diversi gradi di capacità di adattamento all'ambiente, oppure senza mente, quindi privi di capacità di adattamento. L'ambiente in cui operano gli agenti può essere classificato come strutturato in base a delle regole oppure del tutto neutrale.

Si può osservare un esempio in Terna (2001), dove si descrive un modello di simulazione di un mercato molto semplice in cui si sperimentano i seguenti scenari:

- agenti senza mente operanti in ambiente non strutturato;
- agenti senza mente operanti in ambiente strutturato;
- agenti con mente operanti in ambiente non strutturato;
- agenti con mente operanti in ambiente strutturato.

Nel primo caso si nota che gli agenti senza mente operanti in contesti non strutturati producono risultati complessi ma non realistici. Inserendo agenti senza mente in un ambiente strutturato, dove per struttura si intende introduzione di un meccanismo di contrattazione telematica come quello di una borsa senza grida, si producono risultati molto realistici, come, in un mercato di borsa, bolle o crash. L'aspetto interessante è rappresentato dal fatto che il realismo dei risultati derivi dalla struttura del mercato e non dalla presenza di agenti con caratteristiche sofisticate.

Nel caso degli agenti con mente, in Terna (2002c) le capacità di adattamento derivano dall'adozione della tecnica dei Cross Target, fondata sulla metodologia delle reti neurali. L'idea di fondo è quella che l'agente non operi facendo ricorso a regole economiche specifiche definite a priori e abbia delle possibilità di apprendimento che diano coerenza alle sue azioni. Con questa semplice struttura, l'agente appare in grado di compiere azioni che sembrano pianificate e dirette al perseguimento di specifici obiettivi.

Agenti di questo tipo, che operano in un ambiente non strutturato, possono determinare risultati aggregati complessi e plausibili.

3.9 Intelligenza Artificiale (A.I.)

3.9.1 Le differenti visioni dell'A.I.

L'intelligenza artificiale è la branca dell'informatica che utilizza i calcolatori elettronici per studiare e riprodurre attività definibili come prodotto dell'intelligenza.

I filosofi, per millenni, si sono interrogati su cosa siano la mente e il pensiero, su cosa contraddistingua gli esseri umani rispetto al resto dell'universo conosciuto, ma le risposte a questi interrogativi sono ancora oggi poco soddisfacenti (Haugeland, 1988). Negli ultimi decenni non solo la psicologia, ma numerose altre scienze si sono occupate di filosofia della mente e, fra queste, spicca sicuramente l'intelligenza artificiale. L'obiettivo primario di questa scienza non è semplicemente quello di imitare l'intelligenza umana, bensì costruire macchine dotate di mente; forme di intelligenza originali e non frutto dell'imitazione delle capacità umane.

Searle (1990) fornisce due possibili visioni sull'intelligenza artificiale, una definita "forte" e l'altra "debole". La prima implica che un programma potrebbe essere una vera e propria mente, nello stesso senso in cui lo è la mente dell'uomo. La seconda, invece, implica che i modelli basati sul calcolatore siano semplicemente utili per studiare la mente, così come sono utili per le condizioni meteorologiche, i processi economici o i meccanismi della biologia molecolare.

Ci sono sostenitori della possibilità che un giorno un computer potrà essere intelligente nel senso classico che diamo a questa parola, mentre altri sono assolutamente scettici. Simon (1979) parte dalla seguente domanda,

> Ma, dopo tutto, si può domandare lo scettico, in che modo un elaboratore può dar prova di intuizione o di creatività? Esso può svolgere soltanto quello per cui lo si programma.

per sostenere che:

> Questa affermazione è del tutto ovvia, ma non riesce a sostenere alcuna delle implicazioni che solitamente se ne traggono.
>
> Un essere umano può pensare, apprendere e creare perché il programma di cui è dotato biologicamente, unitamente alle modifiche intervenute in esso in seguito all'interazione con il suo ambiente dopo la nascita, gli consente di pensare, di apprendere e di creare. Se un elaboratore riuscirà a fare tutte queste cose, ciò sarà possibile grazie a un programma che gli conferirà tutte queste capacità. Chiaramente non si tratterà di un programma che richieda un comportamento altamente stereotipato e ripetitivo, del tutto sganciato dagli stimoli provenienti dall'ambiente e della funzione che l'elaboratore dovrà svolgere, ma che svolga lo stesso ruolo del programma di cui è dotato l'essere umano. [...] Sarà quindi un programma in grado di analizzare, mediante qualche dispositivo, i propri risultati, di diagnosticare i propri insuccessi e di operare variazioni che ne migliorino l'efficienza futura. La possibilità di riuscire o meno a predisporre un programma per l'elaboratore che possegga tutte queste proprietà è semplicemente una questione di fatto, la cui risposta è che simili programmi sono già stati scritti.

Tra gli oppositori ad una visione così ottimistica c'è invece Searle (1990), che sostiene che:

> Un buon numero di scienziati che si occupano di intelligenza artificiale [...] credono di creare letteralmente delle menti allorquando scrivono i programmi giusto, con gli ingressi giusti e le

> uscite giuste. Credono inoltre di possedere un criterio scientifico per stabilire il successo o il fallimento dell'impresa: il cosiddetto "test di Turing"

Il test di Turing consiste in questo: se un calcolatore riesce a comportarsi in modo tale che un esperto non sia in grado di distinguere il suo comportamento da quello di un essere umano che possegga una certa capacità cognitiva (per esempio di fare le divisioni o capire una lingua), allora anche il calcolatore possiede queste capacità. Il fine è quindi quello di essere in grado di costruire programmi che abbiano le capacità di superare il test di Turing, simulando le capacità cognitive dell'uomo. Searle (1990) aggiunge che un programma in grado di superare questo test sarebbe considerato non solo un modello della mente: ma una vera e propria mente, nello stesso senso in cui lo è la mente dell'uomo.

Proprio per distinguere questa visione, per lui irrealistica, da quella invece possibile, Searle ha diviso in due impostazioni l'intelligenza artificiale: quella "forte" e quella "debole". Le divide perché, come lui sostiene,

> ...è importante rendersi conto di quanto la posizione espressa dall'intelligenza artificiale (I.A.) "forte" sia audace. Secondo l'I.A. "forte" il pensiero non è altro che la manifestazione di simboli formali, e questo è proprio quanto fa il calcolatore: manipola simboli formali. Questa posizione viene riassunta con la frase: "la mente sta al cervello come il programma sta al calcolatore.

3.9.2 Un po' di storia

I primi sistemi creati risolvevano problemi logico-matematici relativamente semplici e ben definiti, la cui soluzione da parte degli esseri umani richiede capacità di ragionamento ma poche conoscenze. Un fatto importante dello studio di questo genere di problemi fu l'elaborazione di metodi generali di

risoluzione (realizzabili, ovviamente da programmi di computer) che sarebbero poi stati utilizzati in seguito in molti altri sistemi. Gli anni settanta videro invece la realizzazione di sistemi che risolvevano problemi specifici - medici, chimici, geologici - facendo uso di vaste basi di conoscenze.

Tutti questi sistemi avevano prestazioni di alta qualità, paragonabili a quelle di esperti umani impegnati negli stessi compiti. Le speranze e l'ottimismo degli anni Ottanta, dipendeva largamente dai loro successi. Anche sul piano strettamente conoscitivo, poteva sembrare che i più raffinati tra questi sistemi incorporassero modelli di processi cognitivi, come la comprensione del linguaggio, più ricchi, articolati e convincenti di quanto la filosofia, la linguistica e la psicologia avessero prodotto in secoli di ricerca (Marconi, 2001). E tuttavia, si deve sottolineare come i sistemi esperti fossero ben lontani dalla versatilità che caratterizza l'intelligenza umana: la qualità (e, in realtà, la possibilità stessa) delle loro prestazioni era vincolata a un dominio drasticamente limitato, al di fuori del quale essi erano del tutto incompetenti. Questa mancanza di versatilità ha molti aspetti: l'incapacità di acquisire le informazioni di volta in volta pertinenti ai compiti da svolgere, la limitazione a problemi ben definiti, l'esigenza che tutte le conoscenze siano rappresentate esplicitamente, la mancanza di senso comune.

Un esempio tratto da Marconi (2001) aiuta a capire queste mancanze: si supponga di essere seduti alla propria scrivania sotto la finestra, e che sulla scrivania ci sia (tra l'altro) un vaso con una rosa. Un colpo di vento spalanca la finestra e rovescia il vaso: la rosa cade sul tavolo, il piano della scrivania si bagna. Un disastro, ma un disastro parziale: la scrivania infatti si è bagnata, ma la finestra presumibilmente no, e certamente neanche i quadri appesi alle pareti, il soffitto, il lampadario, la porta e la maggior parte degli oggetti della stanza.

Tutto ciò è ovvio per chiunque di noi (è questione di senso comune), ma

non lo è per un sistema artificiale che avesse una rappresentazione della stanza prima del fattaccio e debba modificarla in conseguenza dell'accaduto. Il sistema ha bisogno di essere informato esplicitamente su cosa è cambiato e che cosa no; alternativamente, può essere in grado di calcolarlo, ma se segue questa strada (il cui carico computazionale è comunque eccessivo) tenderà a generare una quantità enorme di informazioni inutili: a che gli servirà mai sapere che la maniglia della porta non si è bagnata? Gli esseri umani sanno identificare le informazioni di volta in volta utili, e sono in grado di acquisirle per via inferenziale o attraverso il contatto cognitivo con l'ambiente; inoltre, essi sembrano disporre di una grande quantità di conoscenze (il cosiddetto senso comune) che forse non sono rappresentate esplicitamente da nessuna parte, ma che sembrano avere un ruolo essenziale nello svolgimento di compiti intelligenti, per lo meno nel senso che un sistema artificiale che non ne disponga *non* è in grado di svolgere quei compiti.

3.9.3 I calcolatori diverranno più intelligenti degli uomini?

Per analizzare il parallelo tra intelligenza artificiale e umana si può prendere in esame il famoso film Matrix. In questo film si immagina che gli uomini siano stati sconfitti, in una guerra disastrosa, da una popolazione di macchine superintelligenti, che ora li tiene in vita per estrarne l'energia che consente alle macchine stesse di funzionare. La vita degli uomini è una sorta di sussistenza puramente vegetativa, che sostiene però un'attività mentale in tutto corrispondente ad una vita "normale": nella loro mente, gli esseri umani si muovono, mangiano, lavorano e interagiscono tra di loro nei modi consueti. Ma nulla di tutto ciò avviene realmente: l'illusione di vita è opera di un programma infinitamente complesso, Matrix, creato e gestito dalle macchine e utilizzato inconsapevolmente dagli esseri umani, che sono

connessi al programma.

L'esperimento mentale dei cervelli ingannati dal computer deriva da tutta una serie di idee che riguardano il rapporto tra il computer e la mente. Di queste, quella che qui interessa trattare è quella della stretta parentela tra ciò che chiamiamo "mente" e un programma di computer. Le macchine schiaviste di cui si parla nel film sono intelligenti nel senso più pieno del termine: sono creative, tant'è vero che hanno creato e gestiscono Matrix, un programma strepitosamente complesso; hanno scopi e intenzioni (vogliono sopravvivere, e proprio perciò allevano gli esseri umani e ne sfruttano l'energia). Sono quindi macchine molto diverse dal nostro computer o dalla nostra automobile; le macchine di Matrix hanno queste notevoli prerogative perché sono, o comunque includono, computer intelligenti, e i computer, a loro volta, sono intelligenti grazie al loro software, cioè all'insieme dei programmi alla base delle loro elaborazioni. L'idea di fondo, dunque, è che l'insieme di capacità cognitive che chiamiamo *intelligenza* può essere realizzato da un programma di computer; un programma per molti aspetti diverso da quelli che girano sui computer che molti di noi hanno sul tavolo, ma essenzialmente analogo a essi. E' l'idea, oggi ben nota, di *intelligenza artificiale* nella versione che Searle (1990) ha chiamato forte. Simon e Newell (1989) sostengono che

> ...uno dei contributi fondamentali al sapere della scienza dei calcolatori è stata la spiegazione, ad un livello abbastanza semplice, di cosa siano i simboli. [...] I simboli sono alla radice dell'azione intelligente, che è naturalmente l'argomento principale dell'intelligenza artificiale. Questa d'altronde è una questione fondamentale per tutta la scienza dei calcolatori, poiché tutte le informazioni elaborate dal calcolatore lo sono in vista di uno scopo, e noi misuriamo l'intelligenza di un sistema dalla sua abilità nel conseguire degli obiettivi nonostante le variazioni, le difficoltà e

le complessità poste dall'area del compito.

Da dove viene l'idea di intelligenza artificiale? Come è possibile che questa idea risalga addirittura agli anni Cinquanta del XX secolo, quando le prestazioni dei calcolatori più potenti erano enormemente inferiori a quelle del più modesto PC di oggi?

Certamente l'idea di intelligenza artificiale non viene dalla constatazione che siano state costruite macchine che sono, di fatto, intelligenti: finora, non sono state costruite (o almeno, è questa l'opinione di gran lunga prevalente). L'ipotesi della possibilità dell'intelligenza artificiale deriva da considerazioni che riguardano in primo luogo l'intelligenza *naturale*, cioè le nostre capacità cognitive. Le principali tra queste constatazioni sono due (Marconi, 2001). La prima consiste nel trattare i nostri processi cognitivi come *elaborazioni di informazioni*, ed è quella che Marconi (2001) chiama "tesi della natura computazionale della cognizione". La seconda riguarda ogni processo di elaborazione di informazioni (dunque, in forza della prima tesi, anche i nostri processi cognitivi), e sostiene l'*indipendenza dell'elaborazione*, nei suoi aspetti essenziali, *dal supporto materiale che la realizza.*

Cosa si intende per "processo cognitivo"? Gli esempi che si fanno di solito sono attività come un calcolo aritmetico, un ragionamento logico, la visione di una scena, la comprensione di una frese ecc. Non è affatto detto che sia una buona idea raggruppare queste attività in un'unica categoria, ma coloro (e sono molti) che lo ritengono, lo fanno di solito perché le concepiscono tutte secondo uno stesso modello, e cioè come qualcosa di simile a un *calcolo*, in cui si opera su certi dati secondo certe regole. I sostenitori della tesi della natura computazionale della cognizione pensano che *tutti* i processi cognitivi siano, nell'essenziale, conformi a questo modello. In sostanza, essi pensano che tutti i processi cognitivi siano calcoli (o, come anche si dice, computazioni); calcoli i cui dati non sono necessariamente numeri o altri enti aritmetici, ma *infor-*

mazioni di qualsiasi tipo, e le cui regole non sono necessariamente consce e anzi, di solito non lo sono. Niente esclude, tra l'altro, che quelle particolari computazioni che sono i processi cognitivi umani siano intrinsecamente vincolate a quello che è di fatto il loro supporto materiale, cioè il cervello umano; che cioè i processi cognitivi siano *intrinsecamente* modi di operare nel cervello. E' importante notare che chi sostenesse questa tesi non vorrebbe dire soltanto che, di fatto, a tutt'oggi il solo *hardware* capace di ragionare, o di comprendere una lingua, è il cervello umano; vorrebbe dire qualcosa di più, e cioè che è nell'essenza di computazioni come il ragionamento o la comprensione del linguaggio di essere eseguibili solo da un supporto materiale che abbia le proprietà materiali del cervello. Se così fosse, l'intelligenza artificiale sarebbe ovviamente impossibile, almeno se per "intelligenza artificiale" si intende l'intelligenza di macchine materialmente molto diverse dal cervello umano.

La tesi del carattere astratto del delle computazioni lascia aperta la possibilità che non sia così: secondo Marconi (2001),

> ... se è vero che un tipo di computazioni (e, per la prima tesi, i processi cognitivi umani sono computazioni) è identificabile non solo al livello dell'implementazione, ma anche al livello dell'algoritmo, allora è concepibile un punto di vista sui processi cognitivi che prescinde dalla particolare implementazione realizzata dal cervello umano.

In altre parole, se si identifica, ad esempio, la funzione di comprensione di una lingua come l'italiano con un determinato algoritmo, niente vieta che quell'algoritmo sia realizzabile da veri hardware, come quello di un computer, oltre che dal cervello umano. Se così fosse, sarebbe del tutto appropriato dire di quel computer che comprende l'italiano.

Naturalmente, la tesi del carattere astratto delle computazioni non garan-

tisce che sia possibile costruire, o reperire, tipi di hardware, diversi dal cervello umano, che siano in grado di comprendere una lingua, o di vedere; essa si limita a svincolare concettualmente le computazioni dalle loro realizzazioni fisiche (Marconi, 2001).

L'intelligenza artificiale, di per sé, è un'impresa tecnologica: si tratta di costruire macchine (e, soprattutto, programmi) che realizzino determinati algoritmi, quelli con cui abbiamo identificato le varie attività intelligenti che si vogliono realizzare. Tuttavia, è chiaro che essa presuppone che quegli algoritmi (di comprensione, di visione, di ragionamento ecc.) siano stati identificati, o, in altre parole, che si sia data una risposta a domande come: "Che cos'è la comprensione del linguaggio?", "Che cos'è la visione?", "Che cos'è il ragionamento?". Rispondere a queste domande è il compito della scienza cognitiva, la quale si occupa dei processi cognitivi al livello degli algoritmi.

In Haugeland (1988) leggiamo che Hobbes, nella prima espressione della visione computazionale del pensiero disse: "Ragionare non è nient'altro che calcolare". Tre secoli più tardi, con lo sviluppo dei computer elettronici, questa è divenuta la più importante ipotesi teorica della psicologia (e di alcune discipline affini), nonché la base della ricerca sull'intelligenza artificiale.

La scienza cognitiva comprende tutte queste diverse imprese scientifiche, a riconoscimento del loro fondamento concettuale comune. Questa espressione, dunque, non si applica a qualsiasi teoria scientifica delle capacità cognitive, ma solamente a quelle che hanno in comune un certo punto di vista - che è talvolta chiamato l'approccio della "elaborazione delle informazioni" o della "manipolazione di simboli". Effettivamente, l'ispirazione guida della scienza cognitiva è che, ad un appropriato livello di astrazione, una teoria sull'intelligenza *naturale* avrebbe la stessa fondamentale forma delle teorie che spiegano sofisticati sistemi di computer. E' questa l'idea che rende l'intelligenza *artificiale* non solo possibile, ma anche una forma centrale e pura di ricerca

psicologica.

Bibliografia

[1] Brusamolin A., Mantovani V., *Laboratorio di informatica*, Cedam, 1992

[2] Bussolin G., *Informatica in azienda*, Boringhieri, 1975

[3] Dennett D. C., *Sistemi intenzionali*, in Haugeland J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[4] Dreyfus H. L., *Dai micro-mondi alla rappresentazione della conoscenza: l'intelligenza artificiale ad un""impasse"*, in Haugeland J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[5] Marr D., *Lintelligenza artificiale: un'opinione personale*, in Haugeland J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[6] McDermott D., *L'intelligenza artificiale soddisfa la stupidità naturale*, in Haugeland J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[7] Minsky M., *Un sistema per la rappresentazione della conoscenza*, in Haugeland J.,in Haugeland J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[8] NEWELL A., SIMON H.A., *La scienza del computer come indagine empirica: simboli e ricerca*, in HAUGELAND J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[9] PRYOR F. L., *Looking Backwards: Complexity Theory in 2028*, in D. COLANDER, *The complexity vision and the teaching of economics*, Edward Elgar, 2000.

[10] PYLYSHYN Z., *La complessità e lo studio dell'intelligenza umana e di quella artificiale*, in HAUGELAND J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[11] SEARLE J. R., *Menti, cervelli e programmi*, in HAUGELAND J., *Progettare la mente, filosofia, psicologia, intelligenza artificiale*, il Mulino, 1989.

[12] SIMON H. A., *Informatica, direzione aziendale e organizzazione del lavoro*, Franco Angeli, 1977.

[13] SIMON H. A., *Le scienze dell'artificiale*. Universale Paparbacks, il Mulino, 1988.

[14] TERNA, P., *Reti neurali artificiali e modelli con agenti adattivi*, XXV riunione scientifica annuale della società italiana degli economisti, 1994.

[15] TERNA, P., *Economia e simulazione: una rivoluzione nel metodo*, Sistemi Intelligenti, pp. 496-502, 1996.

[16] TERNA, P., *CT Scheme and ERA Scheme*, *http* : $//web.econ.unito.it/terna/ct-era/ct_era.html$, 2000.

[17] TERNA, P., *Hayek e il connessionismo: modelli con agenti che apprendono*, in G. CLERICO E S. RIZZELLO, *Il pensiero di Friedrich von Hayek*, Utet, 2000c.

[18] TERNA, P., *Cognitive Agents Behaving in a Simple Stock Market Structure*, in F. LUNA AND A. PERRONE, *Agent-Based Methods in Economics and Finance: Simulations in Swarm*, Kluwer Academic, pp.188-227., 2001.

[19] TERNA, P., *Economic Simulation in Swarm: Agent-Based Modelling and Object Oriented Programming*, in BENEDIKT STEFANSSON AND FRANCESCO LUNA, *A Rewiew and some Comments about Agent Based Modeling*, The Electronic Journal of Evolutionary Modeling and Economic Dynamics, n°1013, http://www.e-iemed.org/1013/index.php, 2002a.

[20] TERNA, P., *La simulazione come strumento di indagine per l'economia*, Workshop su Scienze Cognitive ed Economia organizzato dalla Associazione Italiana di Scienze Cognitive, 2002c.

Capitolo 4

Programmazione ad oggetti: Swarm e jES

4.1 Object Oriented Programming

La maggior parte dei linguaggi di programmazione sviluppati nel corso degli anni, è stata di tipo sequenziale. In questo modo, è possibile descrivere un problema, o creare un modello, con una logica di tipo top-down, in cui cioè si segue una coerenza piuttosto rigida e lineare. In questi linguaggi, è sicuramente possibile creare subroutines, cioè parti di codice da eseguire più volte ed in maniera quasi autonoma, tuttavia per molti scopi della simulazione sociale questo tipo di programmazione si rivela particolarmente ostica.

La programmazione ad oggetti rivoluziona il modo di programmare perché permette di invertire l'ordine precedentemente utilizzato per realizzare un programma. Con la struttura tradizionale si impostavano in primo luogo una serie di funzioni (chiamate algoritmi) allo scopo di risolvere un determinato problema, solo successivamente i programmatori cercavano un modo appropriato per memorizzare i dati. Per questo motivo, l'autore del Pascal originale, Niklaus Wirth, ha chiamato il suo libro sulla programmazione Al-

gorithms + Data Structures = Programs (Algoritmi + Strutture di dati = Programmi; Prentice Hall, 1975). Il fatto che Wirth abbia messo la parola algoritmi prima di strutture dati è un indicatore di come i programmatori dell'epoca lavorassero; si decideva infatti prima come manipolare i dati e poi quale struttura impostare per facilitarne le manipolazioni.

Ciò che accade con la OOP è esattamente il contrario. Con i linguaggi di programmazione ad oggetti si dà la precedenza alla struttura dei dati lasciando in secondo piano gli algoritmi che agiscono sugli stessi.

L'Object-Oriented Programming (OOP) è il paradigma di programmazione che ha sostituito le tecniche di programmazione sviluppate nei primi anni Settanta. Il vantaggio principale di questa tecnica è la capacità di incapsulare il codice relativo a ciascuna unità logica all'interno della definizione della classe, potendo esporre all'esterno soltanto un'interfaccia; ciò implica che quando l'oggetto deve essere utilizzato è sufficiente conoscere a quali messaggi può rispondere e che tipo di azioni è in grado di compiere. Il modo specifico con cui lo fa non è interessante e viene nascosto nella definizione di classe.

La OOP ha alcune proprietà essenziali:

- Astrazione: il programma che descrive un oggetto è scritto in una classe, poi nel programma principale, gli oggetti sono creati come istance della classe.

- Incapsulamento: gli oggetti nascondono i loro metodi e dati, separano l'interfaccia dell'implementazione. Si sa cosa fa un oggetto, ma come lo fa si trova da un'altra parte, e può essere modificato senza dover cambiare i programmi che usano l'oggetto. I valori delle variabili nell'oggetto sono propri all'oggetto. Devono essere scritti per passare quest'informazione all'esterno dell'oggetto.

- Ereditrietà : ogni classe di oggetti può essere derivata da altre classi

già scritte. In questi casi la classe figlia eredita di tutte le variabili e i metodi della classe madre , li può modificare e può aggiungerne altri.

- Polimorfismo: Si possono avere metodi con lo stesso nome, ma che hanno parametri diversi.

4.2 Swarm

Si è fin qui parlato di simulazioni, di programmi informatici, di modelli ad agenti, e soprattutto della loro applicazione nelle scienze e nell'economia in particolare. Tuttavia, non si dà certo per scontato che i ricercatori e gli studiosi siano esperti anche nel campo informatico oltre a quello di loro specializzazione. Spesso non hanno neanche le conoscenze di base per sviluppare un'applicazione informatica complessa come una simulazione. Per ovviare a queste difficoltà tecniche, e per permettere ai ricercatori di concentrarsi sul modello che intendono simulare, esistono diversi applicativi che aiutano a farlo; fra questi, presenta soluzioni di grande interesse per la comunità scientifica la biblioteca di funzioni chiamata Swarm [1]. Essa rappresenta lo strumento di riferimento nello sviluppo di modelli informatici basati su agenti.

Aderire al progetto Swarm, significa percorrere una strada difficile, ma sicuramente rigorosa e flessibile, nello sviluppo di modelli informatici. In Askenazi ed altri (1996) si legge come agli albori dello sviluppo di una disciplina scientifica, i primi ricercatori sono generalmente costretti a sviluppare i propri strumenti sperimentali: molando le lenti, costruendo le proprie particolari sonde di misurazione, talvolta costruendo gli stessi calcolatori. Essi devono diventare ingegneri, meccanici, elettronici, oltre ad essere semplicemente scienziati. Quando un campo di ricerca diventa maturo, la collaborazione tra scienziati e ingegneri conduce a sviluppare strumenti standard ed

[1] Si può visitare il sito di Swarm all'indirizzo http://www.swarm.org

affidabili, permettendo ai primi di concentrarsi sulla ricerca piuttosto che sulla costruzione degli strumenti. L'uso di strumenti scientifici diffusi permette non solo di aumentare la quantità di tempo speso dai ricercatori nelle ricerche, ma anche e soprattutto l'ottenimento di risultati ripetibili e comparabili.

Sfortunatamente, la modellazione al computer (Parisi, 2001)

> ...ha trasformato spesso buoni scienziati in pessimi programmatori.

La conseguenza è che studi concettuali di alto livello sono progettati e realizzati in modo non adeguato. Per ovviare a questi problemi è nato il progetto Swarm, dal lavoro di un gruppo di ricercatori dell'Istituto degli Studi sulla Complessità di Santa Fe, al fine di produrre strumenti che facilitino il lavoro di ricerca attraverso la collaborazione tra scienziati ed ingegneri.

In Swarm non è richiesta la formulazione di specifiche ipotesi quali la presenza dello spazio, la necessità di fenomeni fisici, una particolare rappresentazione interna degli agenti o delle loro interazioni. Per poter utilizzare Swarm è però necessario conoscere un linguaggio di programmazione a basso livello, Java, avendo in cambio la possibilità di creare qualsiasi tipo di simulazione Agent Based. Con Swarm si può realizzare praticamente qualsiasi modello ad agenti relativo a fenomeni chimici, fisici, economici. Ciò è possibile poiché tutto il codice informatico relativo al comportamento degli agenti deve essere integralmente scritto dal ricercatore; Swarm fornisce soltanto funzioni generiche che possono facilitare il lavoro di scrittura del codice, come la gestione dei tempi e l'interazione grafica con l'utente.

In sostanza Swarm consiste in una biblioteca di funzioni ed il protocollo per utilizzarle. È caratterizzato da un linguaggio Object-Oriented; le librerie di Swarm sono scritte in Objective-c, un linguaggio che segue la filosofia OOP (Object Oriented Programming). Gli agenti sono costruiti all'interno di oggetti, le popolazioni di agenti sono le classi, e un particolare agente è una

instance (esemplare) della classe. Il comportamento di ogni oggetto è definito dai metodi della classe, ed ogni azione dell'agente richiama un metodo che viene eseguito dallo specifico agente.

Johnson and Lancaster (2000b) affermano che:

> Swarm is designed to help researchers build models in which low-level actors interact (often called complex systems). One research goal is to discern overall patterns that emerge from these detailed behaviors at the individual level.

La componente principale delle simulazioni ad agenti è un oggetto detto "swarm", letteralmente "sciame". Ma di cosa è composto lo sciame?

In generale uno sciame è composto da agenti, la cui tipologia dipende dalla ricerca che si vuole svolgere; potrebbero essere formiche, ambulanze, agenti di borsa, o unità produttive di un'azienda. Lo sciame è caratterizzato da un calendario degli eventi che accadono nel mondo simulato. Si potrebbe avere, ad esempio, una colonia di insetti che muovendosi nello spazio crea calore[2], e le azioni potrebbero essere i movimenti per cercare di raggiungere una certa temperatura. Lo sciame in questo caso rappresenta l'intero modello, comprendendo sia gli agenti che le loro azioni. Oltre che insiemi di agenti gli sciami possono essere agenti a loro volta; un agente è tipicamente definito da una serie di regole e di risposte agli stimoli (nell'esempio degli insetti la ricerca del calore come risposta alla temperatura del mondo esterno). I singoli agenti potrebbero essere sciami a loro volta, in questo caso il comportamento degli agenti è definito dai fenomeni emergenti dalla simulazione. L'approccio gerarchico utilizzato consente di creare una simulazione strutturata su più livelli, creando sciami e sotto-sciami.

La possibilità di costruire modelli multi livello è molto potente, poiché consente di rappresentare esplicitamente un fenomeno emergente, nel quale

[2]E' l'esempio chiamato 'heatbugs', disponibile all'indirizzo http://www.swarm.org

un gruppo di agenti (la popolazione) possono comportarsi coerentemente come un singolo individuo.

In un esperimento con Swarm si segue usualmente una determinata procedura, che è dettagliatamente descritta dal manuale Johnson and Lancaster (2000a); tale procedura può essere sintetizzata in:

1. Creare un universo virtuale costituito da spazio, tempo e oggetti, che possono essere collocati in certe posizioni spazio-temporali dell'universo. Gli oggetti hanno dei comportamenti in accordo con il loro stato interno e lo stato dell'universo;

2. creare gli strumenti per l'osservazione del modello. Questi oggetti servono a registrare e analizzare i dati prodotti dagli agenti nell'universo artificiale;

3. eseguire la simulazione, attivando contemporaneamente il modello e gli strumenti di osservazione di questo;

4. interagire con l'esperimento attraverso i dati prodotti dagli strumenti di osservazione, per eseguire dei controlli sulla simulazione;

5. in funzione dei risultati ottenuti tornare al punto precedente modificando la simulazione per verificare nuove ipotesi, sorte dall'osservazione del modello;

6. Pubblicare i risultati ottenuti, avendo cura di includere le specifiche tecniche dell'esperimento, così da renderlo replicabile e falsificabile.

Nel nostro modello, Il tool di sviluppo Swarm è stato utilizzato in Java. La scelta per realizzare jES è ricaduta su Java, e non ad esempio su Objective C, perché oggi probabilmente è il più diffuso ed evoluto; inoltre, possiede i seguenti vantaggi pratici:

- è semplice da usare, anche se i nomi dei metodi sono generalmente più lunghi;
- si scrivono meno file; in Objective-C ogni classe e costituita da 2 file, mentre in Java da uno solo;
- incoraggia lo "static typing", quindi forza a evitare alcuni errori (specialmente di runtime), che si possono commettere con Objective-C;
- Java distrugge automaticamente gli oggetti non usati;
- l'interfaccia Java offre la possibilità di sfruttare delle numerose librerie grafiche di Java (che sono più ricche di quelle offerte da Swarm);

Utilizzare Java ha anche alcuni svantaggi, fra cui il fatto che esistono meno risorse, tutorials e programmi per Swarm in Java.

4.3 Costruzione di un modello in Swarm

Secondo la metodologia degli autori di Swarm, la realizzazione di un modello di simulazione è composta da tre passi successivi (secondo la filosofia di Swarm):

1. innanzitutto si deve scrivere il codice relativo alle classi che corrispondono alle categorie di agenti previsti dal modello.
2. in seguito avviene la fase di scrittura della classe denominata ModelSwarm, la quale contiene il codice relativo alla costruzione ed alla gestione da un punto di vista aggregato degli agenti del modello. Essa corrisponde al modello vero e proprio ed è dotata della descrizione della sequenza di azioni che ciascun agente dovrà compiere in ogni ciclo temporale.

3. infine, si deve creare la classe ObserverSwarm, composta da un oggetto che si occupa di esaminare il comportamento dell'agente ModelSwarm nel suo comportamento aggregato. Ciò significa che in questa classe vengono costruiti il modello di simulazione, tutti gli oggetti di analisi grafica e la lista delle azioni di ciascun componente che osserva il comportamento di un oggetto, sia esso il modello o un singolo agente.

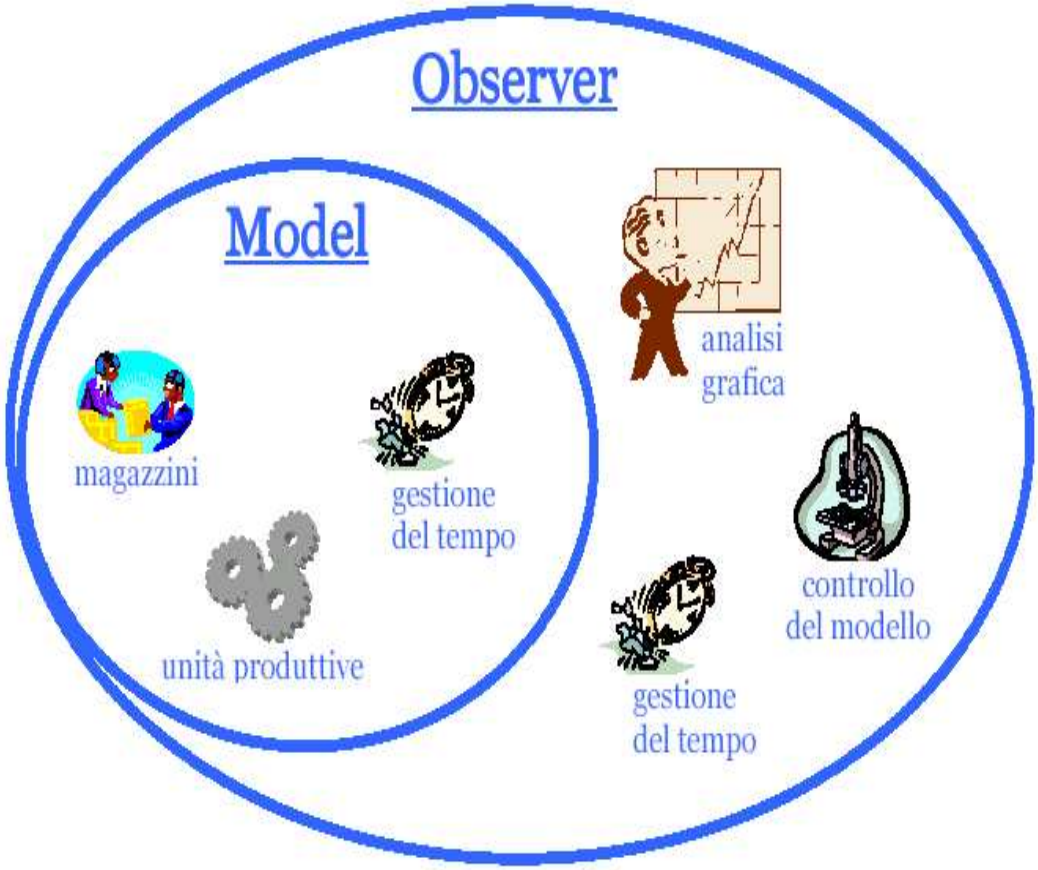


Figura 4.1: Schema grafico di un modello di simulazione in Swarm

Il model costruisce un numero di agenti pari al parametro opportuno delle categorie previste. Un oggetto detto schedule gestisce lo scorrere del tempo e il susseguirsi degli eventi. Nel suo complesso questo oggetto dà vita al modello e ne contiene tutte le componenti necessarie al funzionamento. Al fine, però,

di permettere allo sperimentatore di osservare e comprendere ciò che accade durante la simulazione, il model diventa un semplice oggetto che, attraverso la filosofia dei building block, è incorporato da un altro oggetto detto observer, il quale è in grado, attraverso le sonde, di osservare ciò avviene all'interno del modello e di elaborare i dati per l'analisi grafica o di altro tipo. Anche quest'ultimo è dotato di un oggetto schedule che gestisce il succedersi degli eventi, poiché l'osservazione dei dati può avvenire con una frequenza diversa da quella che è utilizzata dal modello. È importante notare, però, che gli oggetti che svolgono la funzione di orologi informatici all'interno del sistema Swarm, sono tutti sincronizzati e quindi gli eventi si verificano secondo una sequenza totalmente controllabile da un punto di vista aggregato del modello dallo sperimentatore.

Come si è precedentemente detto, Swarm è una biblioteca di funzioni. Le librerie di Swarm svolgono due funzioni principali: esse sono una serie di classi che i costruttori di modelli possono utilizzare direttamente, creando esemplari di oggetti derivati da esse. Per la maggior parte degli oggetti, in particolar modo quelli particolarmente tecnici, come strutture di dati, è probabile che gli utenti utilizzino questo semplice metodo. Come seconda funzione, invece, le librerie di Swarm possono essere utilizzate creando delle sottoclassi, specializzate in particolari funzioni, in base al tipo di modello che si desidera costruire.

4.4 Lo schema ERA

Secondo Gilbert e Terna (2000), passando da semplici modelli a risultati complessi, sorgono problemi, i quali suggeriscono che un ruolo cruciale per l'utilità e l'accettabilità degli esperimenti, viene giocato dalla struttura con che descrive i modelli stessi. Per questo motivo, si introduce uno schema generale, che può essere utilizzato per costruire simulazioni basate su agenti. La

struttura originale dello schema Environment-Rules-Agents (ERA) prevede di collocare gli oggetti in tre distinte sezioni (Terna, 2002).

L'ambiente (Environment) è una componente che contiene le informazioni condivisibili da tutti gli agenti del modello; all'interno del protocollo Swarm normalmente la classe è definita ModelSwarm, vale a dire il contesto all'interno del quale si definiscono gli agenti, se ne strutturano le liste, si individuano gli eventi nel tempo, si chiariscono le regole di interazione tra gli agenti grazie ai metodi (interpretabili come messaggi che gli agenti sono in grado di gestire, anche reagendo con azioni ed informazioni) definiti all'interno degli oggetti creati dal Modello.

Lo strato denominato Agent contiene la descrizione di tutti gli agenti del modello e corrisponde alla rappresentazione logica dello stesso. Gli agenti però devono possedere soltanto i metodi relativi alle diverse tipologie di operazioni che sono in grado di svolgere; le regole e la capacità adattiva che ne governano il comportamento devono essere collocate in oggetti esterni all'agente stesso. La sezione Rules contiene proprio gli oggetti in grado di applicare le regole di comportamento degli agenti ed eventualmente di modificarle. Sarà possibile inserire regole originali, ma anche regole in grado di modificarne altre: per questo motivo, l'oggetto RuleMaster, cioè il depositario delle regole, è legato nello schema ad un altro oggetto, detto RuleMaker, il cui compito è quello di modificare le regole che governano il comportamento dei singoli agenti. Attraverso questo schema la variazione dei meccanismi cognitivi o adattivi degli agenti non richiede la modifica dell'infrastruttura del modello; le regole di interazione, infatti, sono mantenute costanti.

Questo schema di progetto è estremamente efficace nella manutenzione e nella comprensibilità del codice, permettendone l'espandibilità, senza dover sprecare tempo a comprendere difficili interazioni tra agenti. Inoltre, se si volessero solo modificare alcune regole alla base del funzionamento degli agenti,

sarebbe sufficiente editare il RuleMaster corrispondente, senza modificare il codice dei singoli oggetti coinvolti nella simulazione.

4.5 Linguaggio Java in Swarm

Swarm è stato scritto originariamente utilizzando il linguaggio di programmazione Objective-C, un dialetto del C, diffuso inizialmente soltanto nell'ambito della piattaforma Unix. Inizialmente, però, l'impossibilità di eseguire Swarm nelle macchine dotate del sistema Windows costituiva un limite alla diffusione dello strumento, considerando che Swarm, come si è detto, si propone di essere uno strumento in grado di aiutare la ricerca scientifica tramite i modelli di simulazione. In seguito, grazie alla nuova possibilità di un compilatore Objective-C per il sistema operativoWindows e della creazione di una versione di Swarm nel linguaggio Java, gli ostacoli alla diffusione sono molto diminuiti.

La caratteristica più rilevante di Java è l'alta portabilità del software prodotto: esso, infatti, è indipendente dalla piattaforma in cui è eseguito; questo elemento consente in prospettiva di diffondere il modello di simulazione con grande facilità all'interno della comunità di studiosi, eliminando il fastidioso compito di ricompilare il codice nelle diverse versioni. Inoltre, è un linguaggio più diffuso poiché deriva sintatticamente dal C++. È necessario notare che la scelta tra i due linguaggi di programmazione non è definitivamente risolta a favore di Java. Objective-C, oggi, è più diffuso all'interno della comunità di utilizzatori di Swarm: i ricercatori hanno investito molto tempo nell'apprendimento del C ad oggetti e sono comprensibilmente restii a mettere in discussione le conoscenze acquisite se non in presenza di rilevanti vantaggi della nuova soluzione. Tutti i modelli già scritti dovrebbero essere inoltre tradotti e poiché il compilatore Objective-C produce codice binario specifico per la piattaforma in cui è eseguito, permette di ottenere

alte prestazioni in termini di velocità. Java, al contrario, è un linguaggio in parte interpretato, quindi più lento. In realtà, la velocità di esecuzione non è un fattore di scelta determinante, poiché il codice relativo ai modelli Swarm richiede molti cicli iterativi di calcolo, e Java è dotato di un compilatore just-in-time che sfrutta proprio i cicli ripetuti di istruzioni per aumentare le prestazioni.

Una ulteriore caratteristica interessante del linguaggio Java è la capacità di creare le applet. Un oggetto Java, può essere eseguito all'interno di un browser, con un minimo intervento da parte dell'utente. Questa caratteristica apre le porte ad un nuovo sviluppo nelle simulazioni di modelli sociali: si configura la possibilità di realizzare modelli aperti che sono eseguiti su un server, e possono interagire con umani o altri software.

4.6 jES - Java Enterprise Simulator

La decisione di simulare un'impresa deriva dal desiderio di osservare e cercare di capire meglio le imprese nelle diverse fasi della loro "vita": la nascita, la crescita, la loro caduta. L'aspetto più importante è quello legato al loro funzionamento interno, spesso largamente sconosciuto. Terna (2003) cita il pensiero di Gibbons al riguardo:

> For two hundred years, the basic economic model of a firm was a black box: labor and physical inputs went in one end; output came out the other, at minimum cost and maximum profit. Most economists paid little attention to the internal structure and functioning of firms or other organizations. During the 1980s, however, the black box began to be opened: economists (especially those in business school) began to study incentives in organizations, often concluding that rational, self-interested or-

> ganization members might well produce inefficient, informal, and institutionalized organizational behaviors.

Il progetto jES (Java Enterprise Simulation) nasce all'interno del dipartimento di scienze economiche e finanziare G. Prato dell'università di economia di Torino nel 2000. Come si può leggere in Terna (2000):

> Con il modello introdotto qui di seguito si intende far funzionare l'azienda simulata, non rappresentarla in modo animato sulla base di sequenze predeterminate di eventi; nel nostro modello gli eventi accadono in modo indipendente, generando interazioni anche imprevedibili tra atti produttivi e unità produttive, proprie della complessità. Certo su un ideale asse che vada dall'astrazione dei cosiddetti vetri di spin come strumento per studiare la complessità allo sviluppo di un videogioco sofisticato con accadimenti non definiti a priori, soprattutto nelle loro reciproche relazioni, ci collochiamo in prossimità della seconda prospettiva.

jES è un modello, realizzato attraverso un programma e che poggia sulla libreria di funzioni di Swarm, la quale permette di riprodurre le realtà delle imprese; questo può essere fatto avendo come fine:

- quello di ricostruire un'impresa o un'organizzazione esistente per simularne il comportamento e per fare studi su di essa;
- quello di costruire ex-novo un'impresa virtuale.

Non deve trarre in inganno la presunta banalità del primo fine: il fatto di ricostruire in un modello di simulazione un'impresa (o un'organizzazione) già esistente, non è inutile né banale. E' invece il primo, fondamentale, passo verso la comprensione di quella realtà. Se si riesce a riprodurre ciò che avviene realmente, significa che si sono capite le logiche di fondo di quel

fenomeno. Inoltre, solo dopo aver riprodotto ciò che avviene nella realtà è possibile sperimentare cambiamenti nei processi produttivi. In questo senso, un'importante testimonianza è quella del direttore della centrale operativa del pronto soccorso di Torino, che nei colloqui preliminari alla realizzazione del modello di simulazione del 118 ha spiegato come il suo desiderio fosse quello di sperimentare possibili cambiamenti ad un servizio delicato e funzionante, nella speranza di migliorarlo; jES può aiutare gli studiosi a comprendere meglio come vengono prese le decisioni in un sistema complesso come può essere un'azienda o un'organizzazione.

Il problema è ampiamente trattato da Simon (1997) che nell'introduzione spiega:

> Administrative Behavior has served me as a useful and reliable port of embarkation for voyages of discovery into human decision making; the relation of organization structure to decision making, the formalized decision making of operation research and management science, and in more recent years, the thinking and problem solving activities of individual human beings.

In Terna (2003) si legge che:

> ...decisions are taken asking actual people what to do: in this way we can simulate the effects of actual choices; we can also use the simulator as a training tool and, simultaneously, as a way to run economic experiments to understand how people behave and decide in organizations. This is the big Simon's (1997) question.

Il modello jES sino ad ora è stato utilizzato per ricostruire la VIR Spa, che è un'impresa che produce valvole idrauliche, e la BasicNet Spa, che è invece un'impresa non tradizionale che produce abbigliamento. Attualmente si è sviluppato, a partire dalla tesi della dottoressa Guerra, il primo modello di

simulazione di un'organizzazione: il sistema di pronto soccorso della provincia di Torino.

Il secondo aspetto del modello è improntato maggiormente su un'analisi teorica dei fenomeni riguardanti l'impresa: il suo funzionamento interno, l'innovazione tecnologica, l'organizzazione aziendale ecc. In questo caso la simulazione aiuta a creare teorie sulla natura dell'impresa come conferma la visione di Parisi (2000):

> Le simulazioni servono anche [...] per elaborare le teorie, estrapolarne e valutarne le caratteristiche e le implicazioni quando sono ancora in fase di costruzione. Più specificamente, con le simulazioni diventa possibile sviluppare e valorizzare un metodo di ricerca che viene usato, ma solo marginalmente e implicitamente, nella scienza: il metodo degli esperimenti mentali.

Quindi la simulazione permette di formulare teorie sulla natura dell'impresa in modo chiaro, esplicito e senza eccessive ed irrealistiche semplificazioni poiché il significato dei suoi concetti è tradotto interamente in forma di programma che permette al computer di eseguire la simulazione. E' importante precisare che ciò che viene riprodotto al computer è un modello della realtà, non la realtà stessa. Ovviamente ci saranno delle semplificazioni e delle mancanze, ma ciò che interessa riprodurre in una simulazione, sono i fenomeni da studiare per tentare di capire la realtà. Per questo, è importante maneggiare con cura i risultati di una simulazione. Se infatti si è interessati ad apportare cambiamenti in un determinato servizio (come nel progetto del 118), non bisogna prendere per verità tutto quello che emerge dalla simulazione; questa è un aiuto alle decisioni, ma non deve essere l'unico strumento. Se, ad esempio, un cambiamento produce cattivi risultati nella simulazione, possiamo essere ragionevolmente portati a sconsigliare quel cambiamento nella realtà, ma se un cambiamento produce buoni effetti non

si può essere sicuri che ciò avvenga anche nella realtà; la simulazione deve essere un aiuto a testare cambiamenti e nuove soluzioni. In Terna (2003) ne abbiamo la conferma:

> A first important field of application of jES is the simulation of actual enterprises, i.e., the creation of computational models of those realities, to understand their behavior, mainly in order to optimize the related decision process.

In Gilbert e Terna (2000) si legge una descrizione formale delle simulazioni come aiuto alle decisioni:

> ...there are three different symbol systems available to social scientists: the familiar verbal argumentation and mathematics, but also a third way, computer simulation. Computer simulation, or computational modeling, involves representing a model as a computer program. Computer programs can be used to model either quantitative theories or qualitative ones. They are particularly good at modelling processes and although non-linear relationships can generate some methodological problems, there is no difficulty in representing them within a computer program.

4.7 Un "mondo" in due parti

Il modello jES si basa sul concetto di simulazione ad agenti e non di simulazione di processo: gli eventi non sono minuziosamente descritti all'inizio e successivamente osservati con lo scorrere del tempo, bensì essi si svolgono in una sequenza di interazioni non controllata a priori dal programmatore che ha progettato il modello. La complessità di jES scaturisce proprio dall'interazione fra i due principali elementi che caratterizzano tale modello: le ricette e le unità.

E' importante definire da subito tre concetti che ricorreranno spesso nella descrizione del funzionamento del modello: il concetto di ordine, ricetta e unità produttiva.

- Ordine: rappresenta un oggetto che deve essere costruito, si tratta di prodotti o semilavorati, o un'attività che deve essere intrapresa, ed in questo caso si tratta di servizi o atti organizzativi. Il modo in cui gli ordini arrivano a compimento è descritto all'interno di una ricetta.

- Ricetta : descrive i passi che devono essere eseguiti in una certa sequenza per ottenere l'esecuzione di un ordine generico.

- Unità produttiva: è una struttura produttiva, ad esempio un macchinario, o una unità strutturata più complessa, presente all'interno del mondo (all'interno o all'esterno di un'impresa), in grado di eseguire alcuni passi necessari per portare a compimento un ordine.

Riassumendo si può dire che all'interno del nostro "mondo" simulato esistono degli ordini che devono essere presi in carico dall'impresa, la quale, per poterli portare a compimento, deve compiere dei passi attraverso le sue unità produttive. L'informazione sui passi da compiere è gestita in modo decentrato dagli ordini i quali detengono le informazioni relative ai passi già fatti e a quelli da compiere per concludere l'ordine stesso.

Ogni ordine richiede un determinato tempo (simulato) per essere portato a termine e la durata degli ordini è espressa attraverso le ricette. Le unità produttive possono prendere in carico un solo ordine alla volta; infatti, se vengono assegnati più ordini contemporaneamente alla stessa unità produttiva questi vengono messi in coda e verranno eseguiti solo quando l'unità si libera. Chiaramente tutte le unità produttive lavorano in modo autonomo le une dalle altre ed in parallelo.

Attraverso jES, si descrive in maniera dettagliata un "mondo" composto

da due parti: quella che descrive le azioni da compiere (the "What to Do" side, WD), in termini di ordini da eseguire, e quella che descrive le risorse per farlo (the "which is Doing What" side, DW), in termini di unità produttive. Terna (2003) spiega che:

> Our simulation model is, first of all, a description of the enterprise, as it is. Just like the various *flight simulator* programs put at our fingers the control of the simulated airplane and then execute our choices, jES executes exactly what we suggest has to take place into the simulated enterprise, on the two sides described above. The plane flights or lands; the enterprise produces or stays clogged if our WD and DW choices are inconsistent.

Il modello è pertanto creato dall'incontro dei due formalismi: le operazioni da compiere, (siano esse produzioni o processi organizzativi) e le unità che compiono le operazioni descritte. In questo modo, è possibile ricreare una situazione reale costruendo, per esempio, la realtà del 118 o ipotizzare una struttura astratta, per valutare nuove configurazioni organizzative. L'organizzazione virtuale così ottenuta può essere oggetto di indagine, esattamente come se si trattasse di un esperimento creato in laboratorio.

In precedenza si è accennato alle code d'attesa della simulazione: da ciò si deduce che la simulazione è in grado di simulare anche lo scorrere del tempo, ed è stato suggerito che questo sia un terzo formalismo (o una terza parte del nostro "mondo") che descrive "Quando fare Cosa" (the "When Doing What" side, WDW). Deciso il livello di dettaglio che si intende utilizzare, l'utente deve associare la più piccola unità di tempo reale presa in considerazione (un secondo, un minuto, dieci minuti, un ora, . . .) all'unità temporale base della simulazione: un tick. Le fasi lavorative sono rappresentate come gruppi di tick; una volta deciso che un tick rappresenta dieci secondi[3] e se, ad esempio,

[3] questo è il livello di dettaglio che si è deciso utilizzare nel modello del pronto soccorso

si vogliono simulare interventi di otto minuti, un intervento all'interno della simulazione sarà composto da 48 tick.

Il lancio degli ordini nella simulazione si può ottenere in modi diversi:

- generando casualmente delle ricette;
- lanciando in modo casuale (o con qualche algoritmo) delle ricette preparate dall'utente;
- lanciando le ricette preparate dall'utente seguendo una determinata sequenza;

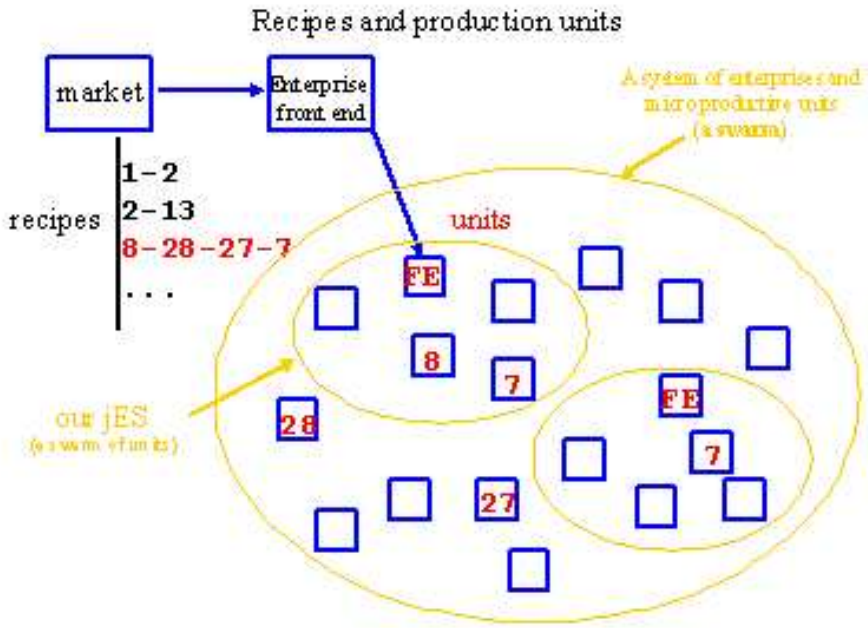


Figura 4.2: Una versione semplificata delle componenti di jES

Come si vede dalla figura 1, ad ogni ciclo della simulazione l'impresa riceve degli ordini dal mercato e questi vengono presi in consegna da un'unità interna che svolge il compito di Front End verso il mercato; tale unità si occupa esclusivamente di passare ogni ordine che le giunge all'unità che è in grado di svolgere il primo passo richiesto dalla ricetta produttiva. Ogni unità produttiva, quando prende in carico un ordine, lo accoda nella propria lista di ordini da eseguire; dopo l'esecuzione richiede all'ordine l'informazione necessaria per trasmetterlo ad una successiva unità.

Il modo in cui vengono assegnati i passi della lavorazione alle unità potrebbe creare alcuni problemi se esistono più unità che possono svolgere un tipo di lavorazione; per ovviare a questi problemi nel modello esistono tre criteri di assegnazione:

1. tra più unità disponibili a fare una certa lavorazione questa viene assegnata alla prima unità che compare nella lista;

2. si assegna la lavorazione all'unità con la coda di attesa più corta;

3. la lavorazione è assegnata in maniera casuale tra le unità che sono in grado di effettuarla.

Esattamente come nella realtà possiamo avere delle imprese che lavorano in outsourcing e che quindi non fanno svolgono le lavorazioni all'interno del loro stabilimento ma le delegano ad altre imprese; in questo caso avremo che le unità produttive non sono interne all'impresa, ma sono in un'altra impresa presente nel mondo o dislocate all'esterno nel mondo. Vediamo ora un semplice esempio introduttivo del mondo che fino ad ora è stato descritto. In figura 2 si può vedere l'impresa simulata, la ricetta che deve essere eseguita e le unità esterne all'impresa indispensabili per portare a conclusione l'ordine se l'impresa che si vuole simulare non ha al suo interno tutte le unità necessarie alla produzione, come in questo esempio.

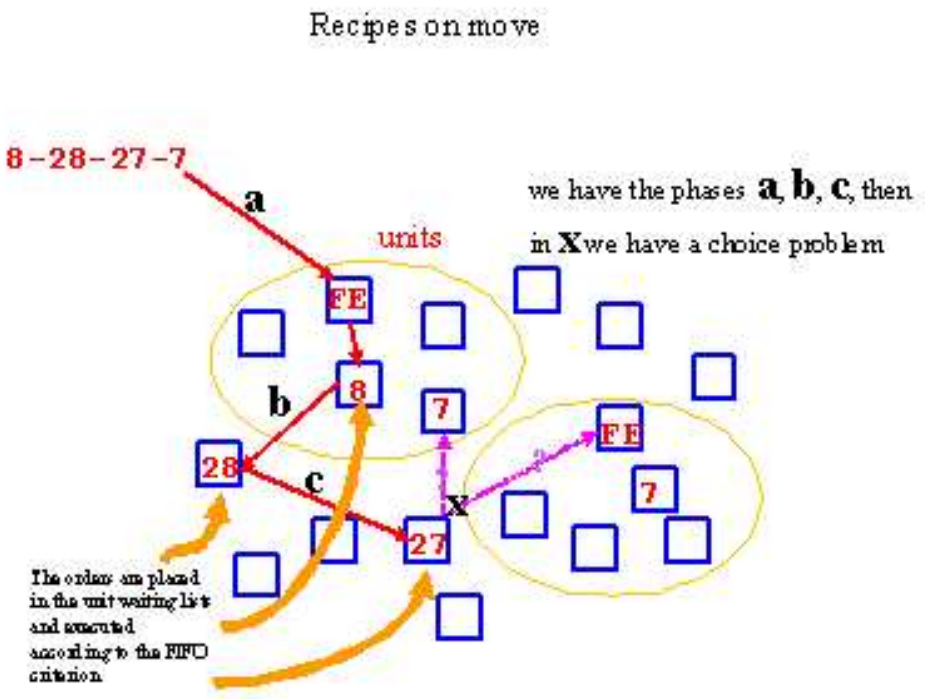


Figura 4.3: Le ricette

La ricetta che l'impresa deve portare a termine è:

8	28	27	7

Tabella 4.1: Le ricette

Il front end dell'impresa prende in carico la ricetta e la passa alla prima unità produttiva che è l'unità che può svolgere lo step 8. In questo esempio l'unità produttiva che prende in carico la ricetta è esterna all'impresa. Una volta che l'unità 8 ha eseguito il passo chiede alla ricetta le informazioni

necessarie per poter affidare la ricetta stessa ad un'altra unità produttiva e per permettere all'ordine di proseguire e di giungere a conclusione. Nel caso specifico esistono due unità che sono in grado di fare lo step 7: l'unità a cui sarà affidata la lavorazione verrà scelta in base agli unit criterions di cui si è accennato in precedenza. Si descriveranno ora nel dettaglio i due formalismi che permettono di descrivere cosa deve fare l'impresa (aspetto WD) e chi è in grado di farlo (aspetto DW), all'interno o all'esterno dell'impresa. Verrà fatto anche un breve accenno al terzo formalismo di cui si è accennato in precedenza, che riguarda il tempo (aspetto WDW).

4.8 Che cosa fare (WD)

All'interno della simulazione di impresa ci sono degli ordini da compiere; per farlo vengono descritti una sequenza di passi produttivi che rappresentano le lavorazioni necessarie per portare a termine una lavorazione: le ricette. Queste vengono rappresentate attraverso un formalismo numerico; i passi della ricetta sono espressi in modo univoco con una cifra, la quale rappresenta la fase produttiva necessaria per la produzione del bene o del servizio dell'impresa; per ogni fase di produzione viene indicato anche il tempo necessario per il compimento dello stadio produttivo. Terminata la ricetta l'ordine è stato completato ed è pronto ad uscire dall'impresa.

All'interno della ricetta è molto importante la specificazione di tempo necessaria per compiere i processi produttivi; questo può essere espresso in:

- d: giorni;
- h: ore;
- m: minuti;
- s: secondi.

Sinora sono stati descritti esclusivamente processi di lavorazione che seguono un ordine sequenziale. Ovviamente questo in un'impresa reale non è sempre vero, per cui all'interno del modello sono stati introdotti alcuni passi speciali che permettono di descrivere:

1. le lavorazioni per lotti.
2. gli atti di approvvigionamento;
3. la ramificazione delle ricette;

4.8.1 I Batch

All'interno della normale attività produttiva dell'impresa esistono lavorazioni che richiedono tempi molto piccoli per essere completate. Questi processi produttivi, se descritti all'interno della simulazione singolarmente, non risulterebbero realistici perché prenderebbero tempi unitari troppo piccoli ed ininfluenti sull'intera simulazione. Per ovviare a questo problema, risulta piuttosto naturale ragionare in termini di lotti. I batch nascono proprio per affrontare questo problema. Un sequential batch, che viene espresso con il simbolo "\" all'interno del formalismo delle ricette, deve essere associato ad una fase produttiva, dopodichè, gli ordini verranno nuovamente gestiti in modo separato.

Oltre ai sequential batch all'interno del nostro modello sono presenti quelli che vengono chiamati stand alone batch. Questo formalismo è stato introdotto all'interno delle ricette per simulare il rifornimento di materie prime necessarie alla produzione di un prodotto. E' introdotto all'interno delle ricette tramite il simbolo "/". A differenza del sequential batch l'utilizzo di questo formalismo è più limitato in quanto può essere inserito unicamente in ricette composte da un passo e da un'unità di destinazione, in genere la endUnit.

4.8.2 Procurement

All'interno dell'impresa virtuale possono essere prodotti dei semilavorati utili a una fase di lavorazione successiva o allo scambio con altre imprese. La produzione di questi prodotti sottintende la presenza di un magazzino nel quale questi vengano stoccati. Come utilizzare questi semilavorati e, di conseguenza, come recuperarli dal magazzino?

Innanzitutto, all'interno del modello, il magazzino è rappresentato come una endUnit; per poter recuperare un componente da una endUnit è necessario utilizzare un passo di tipo procurement, introdotto all'interno delle ricette dal simbolo "p". La ricetta che permette un approvvigionamento è così composta:

7	s	20	p	2	30	27	8	h	3

Tabella 4.2: Esempio di procurement

Dall'esempio si può vedere come il passo 8, per poter essere portato a termine, ha bisogno di 2 prodotti non finiti che hanno codice 30 e 27 che sono stati stoccati precedentemente nelle endUnit. In figura 3 si può vedere una rappresentazione grafica di un procurement.

4.8.3 Il Processo Or

Un'ulteriore componente di realismo all'interno di jES è dato dalla possibilità, per l'mpresa, di scegliere in che modo produrre un determinato bene o eseguire una determinata lavorazione. La caratteristica di queste possibilità di scelta è che i prodotti finali risulterebbero uguali ma i processi impiegati, e quindi i costi, potrebbero risultare molto diversi.

Un passo di tipo Or, espresso con il simbolo "||", permette di ramificare la ricetta produttiva su due o più alternative. In figura 4 si può vedere come

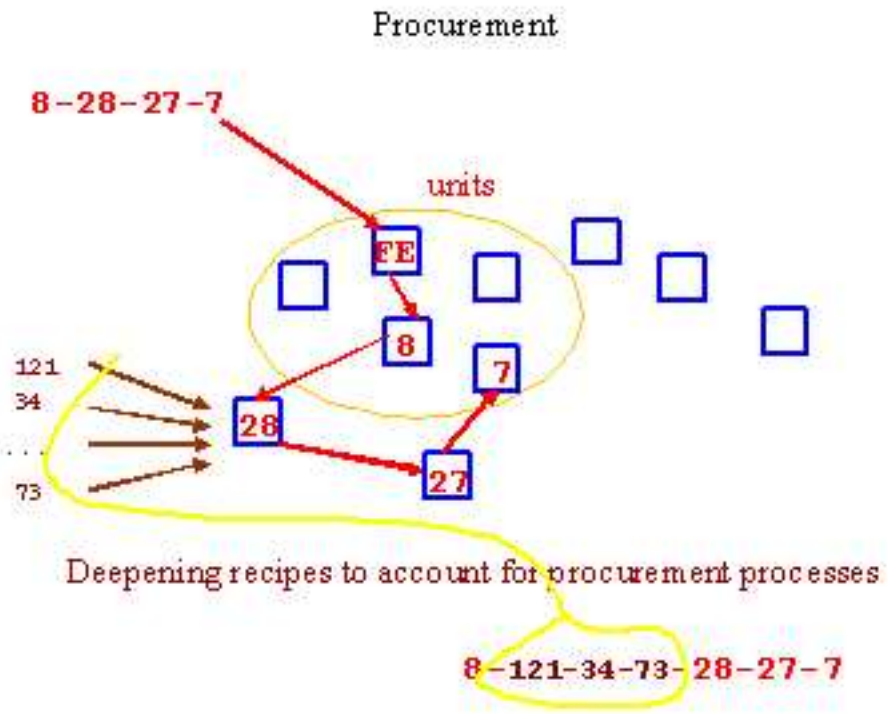


Figura 4.4: I procurements

dopo il passo 1, si può avere la sequenza con i due passi n2 e n3, oppure quella con il passo n22. Dopodiché l'esecuzione della ricetta continua con il passo n4.

La scelta sul passo da compiere avviene in funzione di alcuni criteri previsti nel modello. È infatti possibile impostare la simulazione in modo che:

- si eseguano tutti i rami possibili di produzione;
- si esegua solo il primo ramo;

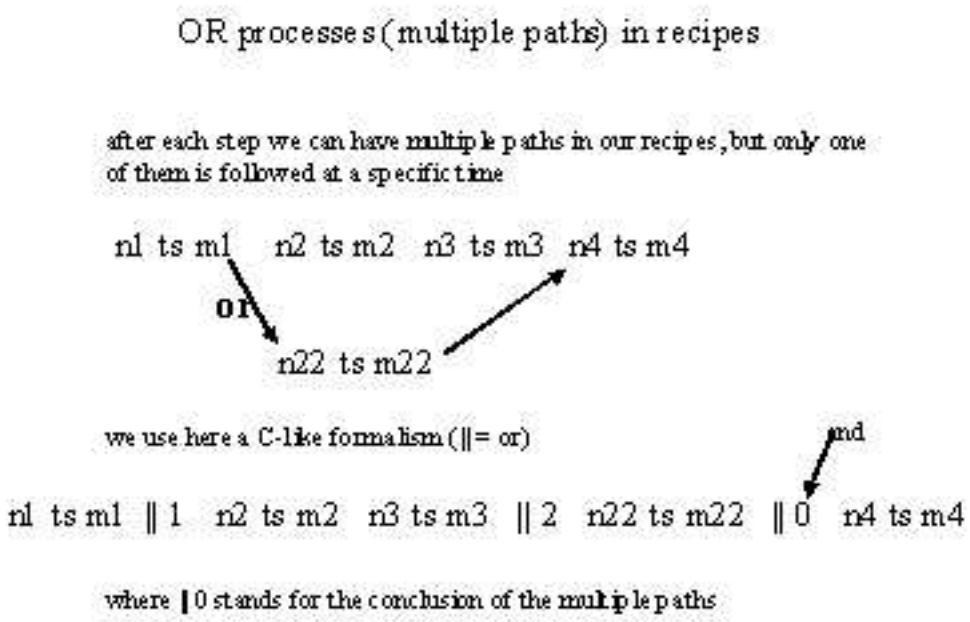


Figura 4.5: Processo OR

- si esegua solo il secondo ramo;
- il ramo da eseguire sia scelto in modo casuale;
- si esegua il ramo che presenta il passo con la minore coda di produzione;
- il ramo sia scelto con l'ausilio di un passo computazionale

4.9 Chi fa che cosa (DW)

Si esaminerà ora il formalismo che serve ad indicare "chi fa che cosa", ossia quel meccanismo che descrive le unità produttive che si occupano di produrre i beni quando sono richiamate dalle ricette. Le unità produttive, presenti all'interno del modello, sono di tre tipi:

1. unità produttive semplici;
2. unità produttive complesse;
3. endUnits.

4.9.1 Unità semplici

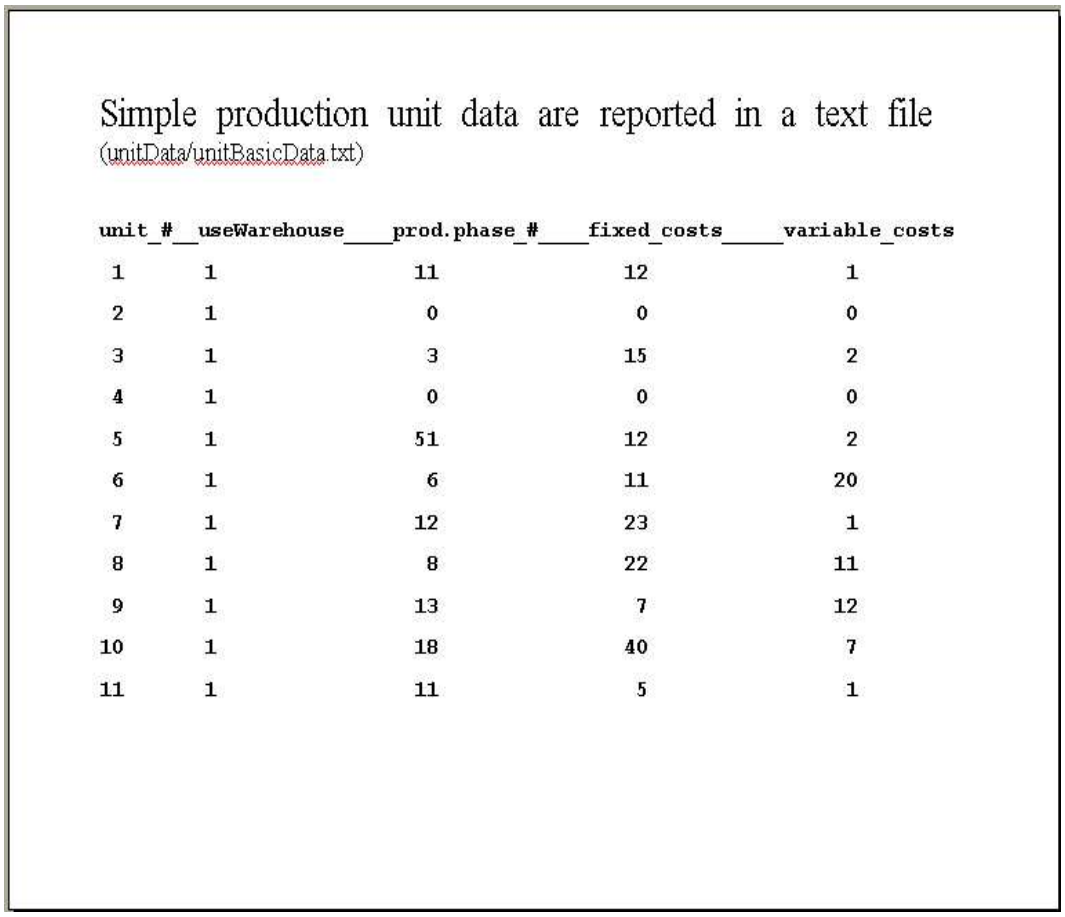
Le unità semplici sono in grado di fare solo un tipo di produzione e questa è inserita nel file che descrive le unità. Il file in questione è chiamato unitBasicData.txt, ed è contenuto all'interno della directory UnitData.

L'attività principale dell'unità produttiva consiste nell'inserire l'informazione di completamento del proprio compito e individuare a quale unità spetta il processo produttivo seguente. Le differenti unità si contraddistinguono in base ad una cifra; ad ogni unità produttiva sono associati uno o più passi che questa è in grado di svolgere.

In figura 5 si ha un esempio di come è composto il file unitBasicData.txt.

La prima riga del file è obbligatoria ed indica:

- il nome delle unità (unit);
- se le unità possono usare gli stand alone inventories (useWarehouse);
- la fase di produzione che le unità sono in grado di fare;
- i costi fissi legati alle unità;



Simple production unit data are reported in a text file
(unitData/unitBasicData.txt)

unit_#	useWarehouse	prod.phase_#	fixed_costs	variable_costs
1	1	11	12	1
2	1	0	0	0
3	1	3	15	2
4	1	0	0	0
5	1	51	12	2
6	1	6	11	20
7	1	12	23	1
8	1	8	22	11
9	1	13	7	12
10	1	18	40	7
11	1	11	5	1

Figura 4.6: unitBasicData.txt

- i costi variabili.

Le informazioni contenute all'interno di questo file sono quindi:

- il codice dell'unità: ad ogni unità presente nella simulazione è attribuito un numero univoco;
- fase di produzione: indica quale passo l'unità produttiva è in grado di svolgere all'interno della simulazione;
- utilizzo dei magazzini: indica se l'unità presenta un magazzino ad essa associato;

- costi fissi : indicano i costi fissi che l'impresa deve registrare ad ogni ciclo della simulazione;
- costi variabili: indicano i costi variabili associati all'attività produttiva.

4.9.2 Unità complesse

Le unità complesse utilizzate nella simulazione sono in grado di compiere, oltre alla fase delle unità semplici, anche altre fasi produttive, che però non vengono descritte nel file unitBasicData.txt, ma in un altro file chiamato units.xls. Per creare unità complesse è necessario inserire all'interno del file unitBasicData.txt uno zero nella posizione che indica la fase di produzione (prod.phase), e poi creare un file Excel chiamato units.xls in cui vengono indicate le fasi produttive che l'unità complessa e in grado di compiere. Ogni foglio del file units.xls corrisponde ad un'unica unità complessa e ad ogni foglio è assegnato il nome dell'unità complessa che è descritta al suo interno.

Riprendendo l'esempio riportato in figura 5, si può notare che la seconda e la quarta unità descritte nel file (le unità 2 e 4) hanno tutti i parametri posti a zero; questo indica al simulatore che deve leggere le fasi produttive che quelle unità sono in grado di compiere nel file units.xls. Il foglio Excel si presenta così:

In questo foglio è stato indicato:

- il numero di fasi che l'unità è in grado di svolgere. Ad esempio, l'unità 2 è in grado di compiere tre fasi produttive, come indicato nella casella in alto a sinistra;
- i codici di ogni fase produttiva che l'unità 2 è in grado di svolgere, indicati nella colonna successiva;
- i costi variabili e costi fissi, che si trovano di seguito ai codici delle fasi produttive;

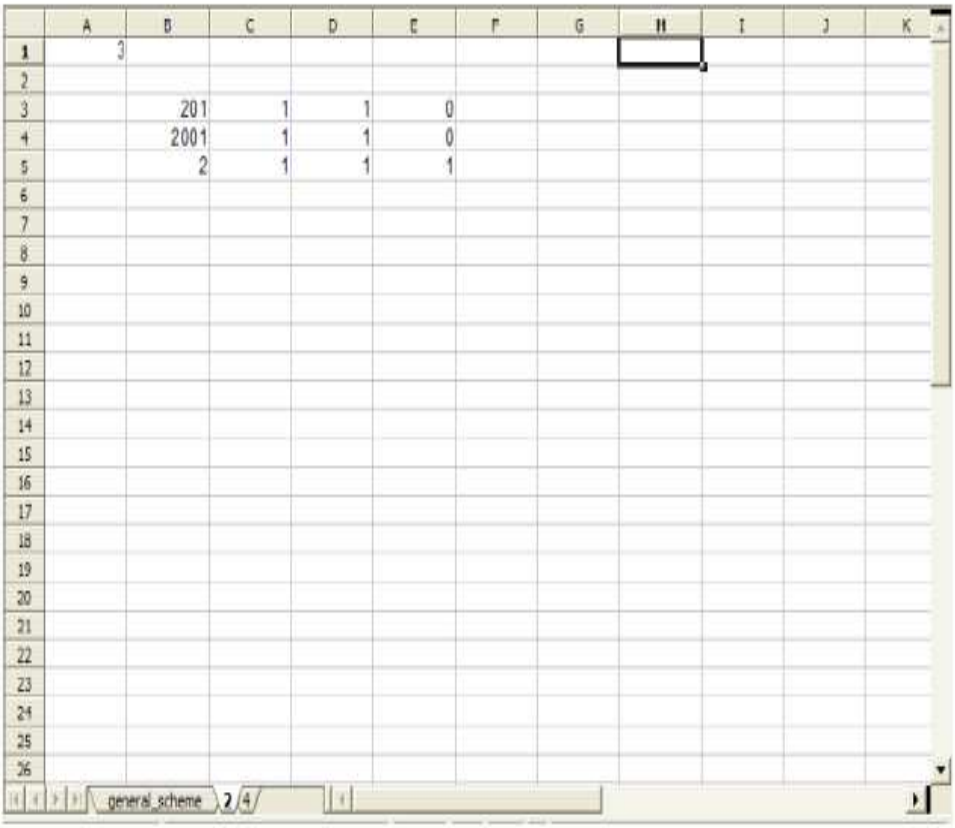


Figura 4.7: Units.xls

- l'utilizzo o meno da parte dell'unità complessa degli stand alone inventory, che viene indicato con 1 o 0 ;

4.9.3 endUnits

Come già detto in precedenza, i componenti della produzione possono essere stoccati per essere poi riutilizzati da altre unità produttive. Per far sì che le materie stoccate siano oggetto di procurement è necessario che tali componenti, al termine delle loro lavorazioni, vengano depositati in unità-magazzino dette endUnit.

Il formalismo per descrivere tali unità prevede di elencare in un file il loro

codice univoco indicando, con un segno meno, se sono sensibili o meno ai layer.

Le endUnits sono descritte in un file chiamato endUnitList.txt contenuto nella directory unitData, che ha il seguente aspetto:

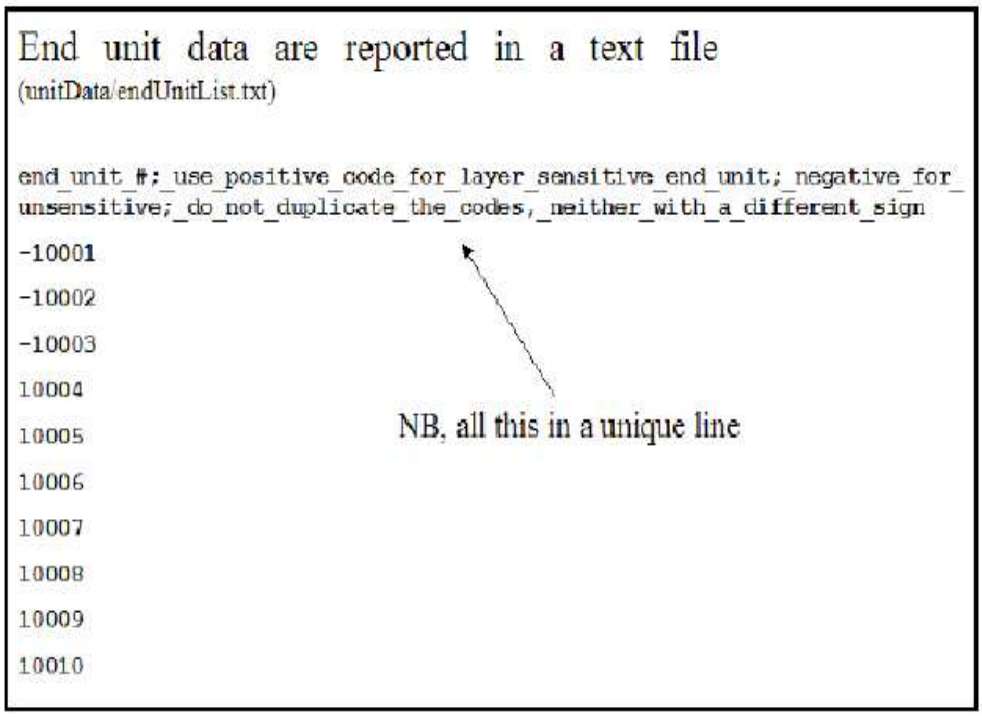


Figura 4.8: EndUnitList.txt

La prima riga del file è obbligatoria. Il codice di queste unità può essere di due tipi, un tipo di codice è sensibile al layer ed è identificato con un codice positivo, mentre un altro, che identifica le end units, è insensibile ai livelli ed è identificato con un codice negativo.

4.10 Quando fare che cosa (WDW)

Per far iniziare la simulazione, innanzitutto, si devono definire i parametri temporali. Vengono stabiliti:

1. ticksInTimeUnit, il quale descrive quanti tick, nell'orologio della simulazione, sono necessari per concludere un unità di tempo;

2. timeToFinish, che descrive il numero di tick al termine dei quali il programma conclude la simulazione.

Gli ordini all'interno della simulazione possono essere generati sia in modo casuale che tramite l'immissione di ricette che vengono lanciate in un preciso momento. Nel primi caso gli ordini sono generati, all'interno della simulazione, dall'orderGenerator, mentre nel secondo caso si sceglie una precisa sequenza temporale descritta nell'oderDistiller. È solo con l'uso di quest'ultimo che viene introdotto il terzo, importante, formalismo di jES: il When Doing What (WDW).

Questo formalismo è molto utile per creare un repertorio di ricette e farle partire secondo una scadenza prestabilita. Le ricette sono contenute in un file chiamato recipes.xls, nella directory recipeData. Le ricette devono essere scritte nel primo foglio di lavoro di Excel, che deve essere nominato Recipe. Le righe e le colonne possono essere usate liberamente, e quindi non importa se all'interno di una ricetta ci sono degli spazi vuoti, perché questi non verranno letti. Ciò che è molto importante è l'ordine con cui vengono disposti gli elementi all'interno del foglio; la sua lettura, infatti, avviene da sinistra verso destra e dall'alto verso il basso, quindi gli elementi devono essere disposti secondo questo ordine.

All'interno del file possono essere scritti dei commenti; per segnalare al simulatore la presenza di righe di commento, si deve mettere all'inizio della riga, e quindi in colonna 1, il segno #. In questo modo il simulatore non legge la riga e passa oltre.

Il primo elemento di una ricetta deve essere il suo nome ed il numero identificativo; di seguito si avrà il corpo della ricetta. Il nome può essere

uguale per ogni ricetta ma il numero identificativo di ogni ricetta deve essere univoco. Ogni ricetta deve essere conclusa con un segno di punto e virgola.

L'elenco degli ordini da lanciare è contenuto nel file orderSequences.xls; ogni riga, conclusa con il segno di ";" descrive gli eventi di un tick dell'orologio della simulazione. Per ogni riga si ha il numero che la identifica ed il numero identificativo della ricetta che deve essere lanciata in quel tick. Il numero identificativo della ricetta è seguito da un segno di * e da un numero che indica le volte che la ricetta deve essere ripetuta. Tra il numero identificativo della riga ed il numero identificativo della ricetta può esserci la lettera "l" seguita da un numero che identifica il livello. Il contenuto del file orderSequences.xls è eseguito un tick per volta e quando la lettura e l'esecuzione del file giungono alla fine la simulazione riprende a leggere gli ordini dall'inizio del file. Il file orderStartingSequences.xls contiene esattamente le stesse informazioni contenute in orderSequences.xls ma è utilizzato quando la simulazione deve fermarsi alla fine della lettura degli ordini, quindi quando le cose da fare vanno ripetute una sola volta. I due file possono essere utilizzati insieme, ma se nella simulazione abbiamo bisogno di utilizzarne uno solo, non possiamo eliminare l'altro; si deve compilare il file inutilizzato riempiendo la prima casella con uno zero seguito dal punto e virgola.

4.11 Altre funzionalità di jES

jES possiede alcune funzionalità molto particolari che possono venire utilizzate all'interno della simulazione. Queste funzionalità sono state create e sviluppate quando si presentavano problematiche tipiche della realtà che si stava studiando. Ad esempio, si può sottolineare come sia stato approfondito l'uso dei livelli con la costruzione della BasicNet Spa.

Con la simulazione del 118 invece si è potuto approfondire un aspetto diverso che riguarda quelli che sono stati chiamati "passi computazionali", i

quali erano già stati creati ma non ancora utilizzati in alcuna simulazione. Sono stati sviluppati:

1. la gestione dei magazzini;
2. la gestione della conoscenza;
3. la contabilità;
4. i layer;
5. i passi computazionali.

4.11.1 Gestione dei magazzini

Affinchè i magazzini funzionino vi è bisogno di regole che dicano quando questi devono riempirsi o svuotarsi. Tali regole sono custodite all'interno di un oggetto chiamato "RuleMaster". In questo oggetto le regole sono dettate da un algoritmo genetico scelto dall'utente. Le unità produttive, quando non lavorano, interrogano questo oggetto chiedondogli se vi sono scorte da produrre e in quale quantità. Il RuleMaster permette anche la gestione delle quantità di semilavorati che possono essere contenute nei magazzini, definendone in questo modo la dimensione. Il funzionamento del magazzino è visualizzabile nella figura 8.

4.11.2 Gestione della conoscenza

Il personale di un'azienda ha una certa conoscenza dei processi che caratterizzano l'organizzazione di questa impresa; quasi sempre tale conoscenza non è completa, ma si limita, essenzialmente, alle incombenze personali. Risulta quindi di notevole interesse studiare la circolazione delle informazioni all'interno dell'azienda.

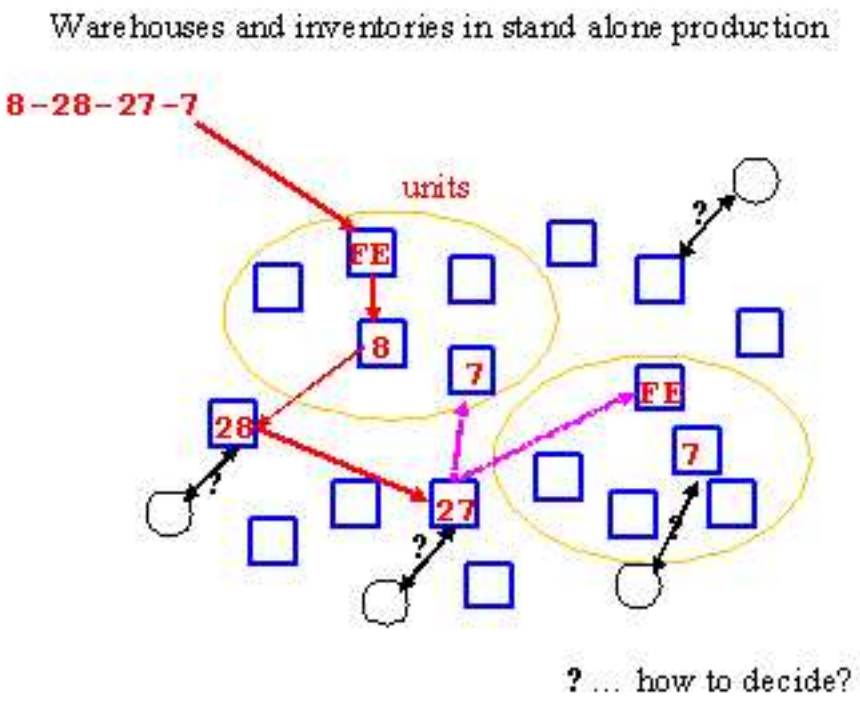


Figura 4.9: I magazzini

Il modello jES prevede una rappresentazione della conoscenza aziendale decentrata, rispettando le caratteristiche delle simulazioni ad agenti e facendo emergere la complessità tipica di questo fenomeno. Le informazioni sono racchiuse all'interno di appositi oggetti definiti News. Le unità produttive decidono ad ogni ciclo se inviare o meno informazioni ad altre unità produttive; attualmente jES permette di simulare una circolazione di informazioni molto semplificata, caratterizzata da una serie di messaggi che le diverse unità si scambiano per facilitare le decisioni relative alla produzione di scorte o alla

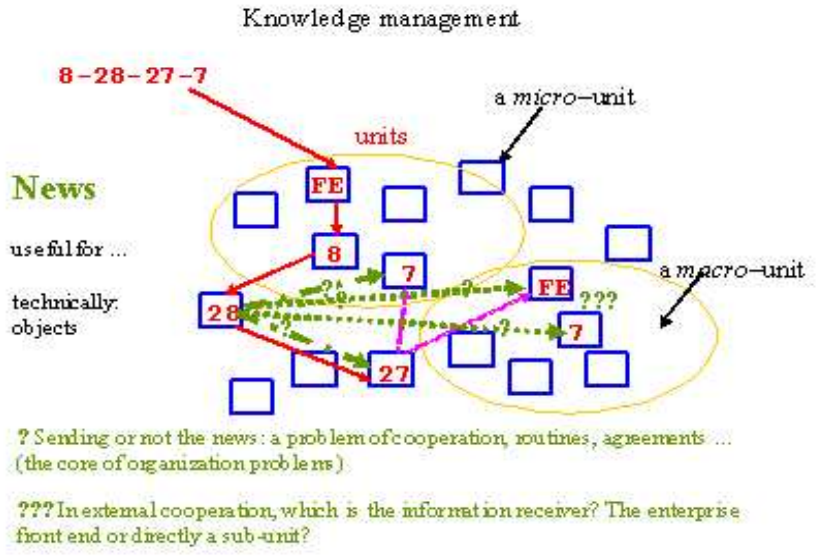


Figura 4.10: L'informazione

destinazione di risorse scarse condivise tra più unità produttive.

4.11.3 Contabilità

jES prevede all'interno del modello due punti di vista differenti per gestire la contabilità; questa può essere eseguita per unità produttive e per prodotto, ma in ogni caso viene aggiornata in tempo reale durante la simulazione. In figura 10 si comprende meglio come funziona questa caratteristica.

La contabilità per unità produttiva è caratterizzata da costi fissi e variabili che, a conclusione di ogni ciclo, l'impresa simulata registra. Nella contabilità per prodotto, in ogni fase di lavorazione il prodotto viene caricato dei costi fissi e di quelli variabili delle unità che lo hanno preso in carico. Se un ordine prevede un approvvigionamento (procurement), i costi accumulati nel prodotto approvvigionato vengono interamente scaricati sull'ordine finale.

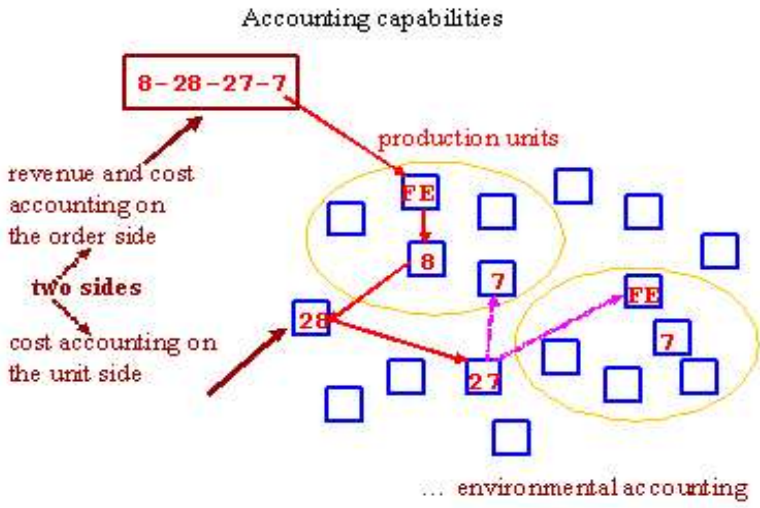


Figura 4.11: La contabilità

Date queste caratteristiche è normale che non sempre i due tipi di contabilità coincidano. In questo modo vengono alla luce i costi fissi delle unità che in fase di lavorazione non sono utilizzate.

4.11.4 Layer

Questo strumento della simulazione nasce dall'osservazione che, ad ordini uguali, possono essere associati beni che hanno caratteristiche fisiche identiche ma caratteristiche qualitative differenti. Si pensi, ad esempio, al modello di simulazione sviluppato per la ditta di abbigliamento BasicNet, nella quale era necessario sviluppare collezioni diverse poichè i prodotti, sebbene identici, dovevano essere trattati in maniera differente perché facenti parte di collezioni diverse. All'interno della simulazione, tramite i layer, si potran-

no distinguere ordini con ricette uguali ma con caratteristiche qualitative o quantitative differenti.

L'assegnazione del layer avviene in fase di lancio degli ordini da parte dell'orderDistiller. Questa diversificazione degli ordini potrà avere, in base alle decisioni dell'utente, effetto o meno sulle operazioni interne alla simulazione. E' possibile, in fase di descrizione del lato DW, distinguere (con il segno -) le unità e le matrici che risultano insensibili ai layer da quelle sensibili.

4.11.5 I passi computazionali

jES possiede capacità computazionali che possono essere associate ad ogni passo della ricetta. Per associare ad passo della ricetta un passo computazionale si deve seguire la seguente notazione:

1	s	1	c	1999	3	0	1	3	2	s	2

Tabella 4.3: Chiamata del passo computazionale 1999

Nell'esempio, dopo l'esecuzione per un tick della fase di produzione 1, la fase di produzione 2 richiama il passo computazionale numero 1999 (la notazione è: c 1999) il quale indica quante e quali matrici devono essere usate. In questo esempio si utilizzano tre matrici di memoria, i cui numeri identificativi sono 0, 1 e 3. Le procedure che devono essere eseguite dal passo computazionale sono contenute all'interno della matrici richiamate.

La creazione delle matrici avviene grazie all'ausilio di un file di testo che si chiama memoryMatrixes.txt, il quale contiene alcune informazioni:

- il numero di matrice (nome della matrice);
- le righe della matrice;
- le colonne della matrice.

Il numero che indica il nome della matrice potrebbe essere anche negativo, indicando così l'insensibilità di quella matrice ai layer che vengono cercati automaticamente. In questo semplice esempio il file delle matrici di memoria contiene le seguenti informazioni:

Numero	Riga	Colonna
0	2	3
-1	3	5
2	4	1

Tabella 4.4: MemoryMatrixes

In questo esempio la seconda matrice è insensibile ai livelli. Il file mamory-Matrixes.txt ha questo aspetto:

Una spiegazione più dettagliata di ciò che ha portato alla creazione del formalismo dei passi computazionali verrà fatta nel prossimo capitolo, ma ciò che è importante dire è che grazie ad essi si è potuta superare la rigida sequenzialità di jES ottenendo la possibilità di avere avvenimenti contemporanei e indipendenti. Tecnicamente quando il passo computazionale viene richiamato, viene attivata una sottoricetta che prosegue indipendentemente dalla ricetta che l'ha lanciato.

Lo scopo degli oggetti computazionali può essere quello di consentire all'impresa simulata di eseguire operazioni immateriali, fare previsioni, gestire aste, indirizzare procurement o archiviare informazioni in database. In questo modo, inoltre, è possibile simulare il sistema informativo dell'azienda.

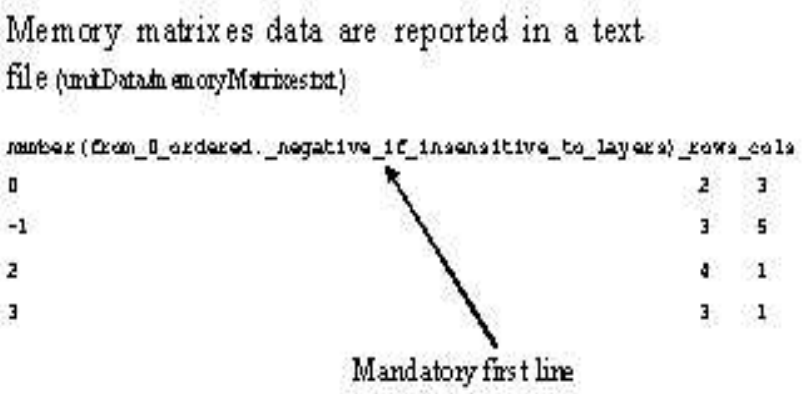


Figura 4.12: memoryMatrixes.txt

Bibliografia

[1] Askenazi M., Burkhart R., Langton C., Minar N., *The Swarm Simulation System: A Toolkit For Building Multi-Agent Simulations*, Santa Fe Working Paper, http://nelson.www.media.mit.edu/people/nelson/research/swarm, 1996

[2] Gilbert N., Terna P., *How to build and use agent-based models in social science*, Mind and Society, 1999

[3] Johnson, P., Lancaster, A., *Documentation set for swarm 2.1.1*, 2000a

[4] Johnson, P., Lancaster, A., *Swarm user guide*, 2000b

[5] Terna P, *Simulation Tools for Social Scientists: Building Agent Based Models with SWARM*, JASSS vol.1, no.2, 1998

[6] Terna P, *Economic Experiments with Swarm: a Neural Network Approach to the Self- Development of Consistency in Agents' Behaviour*, in Luna F., Stefansson B., *Economic Simulations in Swarm: Agent Based Modelling and Object Oriented Programming*, Kluver Academic, 1999

[7] Parisi D., *Simulazioni. La realtà ifatta nel computer*, Il mulino, 2001.

Capitolo 5

Il sistema di pronto soccorso

5.1 Dalla Croce Rossa al 118

In Italia la gestione delle risorse sanitarie è affidata alle regioni, che operano sulla base di leggi emesse dagli organi centrali[1]. La norma più importante riguardante l'emergenza sanitaria territoriale è il DPR 27 marzo 1992, che determina la nascita delle Centrali Operative, definisce i ruoli e le figure professionali impegnate nel settore, individua un numero telefonico gratuito e unico in tutto Italia, il 118.

Prima dell'entrata in vigore di questa legge il percorso della chiamata non era predeterminato in quanto mancava un organismo unico per la gestione dei mezzi dell' emergenza, con conseguente carenza di coordinamento, ritardi nei soccorsi, mancanza di professionisti e di protocolli, i soccorsi erano affidati unicamente a volontari, spesso con pochissima preparazione di base.

Il 118 è il numero unico per l'emergenza sanitaria adottato in misura crescente su tutto il territorio nazionale a partire dal 1992. Questo numero è indice dell'evoluzione di un sistema che, fino a pochi anni fa, non solo non esisteva ma era anche lontano dalla cultura tradizionale del Servizio Sanitario

[1]Questo capitolo è stato scritto in collaborazione con Fabiana Guerra e Carlo Badino

Nazionale rivolta unicamente alle emergenze che si sviluppavano all'interno degli ospedali stessi.

L'urgenza sul territorio era classicamente un patrimonio e un campo di interesse specifico di diverse associazioni o enti in larga parte appartenenti alla sfera del volontariato (dalla Croce Rossa Italiana, alle Misericordie, alle tante Croci locali, ecc.) che, nel corso degli anni, hanno sostituito l'istituzione pubblica che sotto questo aspetto è stata per lungo tempo latitante.

L'importanza delle associazioni di volontariato, che per lungo tempo sono state indispensabili nel garantire il servizio di emergenza sanitaria sul territorio, ha fatto sì che una piccola parte della mia tesi sia dedicata alla storia che descrive la nascita di una di queste associazioni: la Croce Rossa, che è indubbiamente la più significativa a livello internazionale.

5.2 La Croce Rossa internazionale

Quando si parla di attività di pronto intervento sanitario non si può non pensare alle associazioni che operano in questo campo e che permettono di salvare un numero rilevante di vite, usufruendo soprattutto dei fondi lasciati da privati e del servizio di molti volontari. Molto spesso però, quando si pensa al pronto soccorso, non si può fare a meno di richiamare alla mente un'associazione in particolare, e cioè la Croce Rossa, insieme alla sua contropartita che opera nei paesi arabi, la Mezzaluna Rossa. Questo collegamento avviene probabilmente perché Croce Rossa e Mezzaluna Rossa sono le uniche associazioni di primo intervento riconosciute a livello internazionale.

Come nacque l'idea di creare un organismo internazionale di questo tipo e soprattutto chi rese possibili i primi, difficili passi di quest'associazione presente in tutto il mondo e fondamentale in ogni situazione di emergenza sono quindi entrati a fare parte di questa tesi.

5.3 La Croce Rossa italiana

La Croce Rossa Italiana è un'Associazione riconosciuta come ente di diritto pubblico dalla Legge n. 70 del 1975. Ausiliaria dei poteri pubblici, partecipa a tutti gli sforzi di prevenzione, di educazione, di protezione sanitaria e medico-sociale su tutto il territorio nazionale. Per perseguire questi obiettivi la Croce Rossa Italiana (C.R.I.) lavora con i differenti attori sociali - imprese, associazioni, collettività locali e territoriali, Stato, Unione Europea e organismi dell'ONU - e si pone al centro del dispositivo dell'azione sociale per tutto quello che concerne i suoi campi di intervento: salute, solidarietà, soccorsi in Italia e nel mondo.

L'insieme delle particolarità della C.R.I. ne fa uno dei principali attori dell'azione sociale in Italia soprattutto perché è la principale promotrice di progetti e di programmi al servizio dei più deboli.

Il primo Comitato dell'Associazione Italiana per il soccorso ai feriti ed ai malati in guerra si costituisce a Milano ad opera del Comitato Medico Milanese dell'Associazione Medica Italiana il 15 giugno 1864, ben due mesi prima della firma della Convenzione di Ginevra. Questo inizia subito la sua attività sotto la presidenza del dottor Cesare Castiglioni, il quale, due mesi dopo la costituzione del Comitato, viene chiamato a Ginevra, insieme ad altri delegati italiani, per esporre quanto fatto a Milano e cosa pensa di fare in avvenire in favore dei feriti e dei malati in guerra. Il 22 agosto 1864 viene sottoscritta anche dall'Italia la Convenzione di Ginevra.

L'11 dicembre dello stesso anno si tiene, a Milano, un congresso in cui si approva il regolamento del Comitato di Milano come Comitato Centrale per il coordinamento delle attività dei nuovi comitati che andavano in quel periodo a costituirsi. Il 20 giugno 1866 l'Italia dichiara guerra all'Austria e le prime quattro squadriglie di volontari partono alla volta di Custoza. Nello stesso tempo si occupa della lotta alla tubercolosi e alla malaria creando stazioni,

ambulatori e ambulanze antimalariche nelle Paludi Pontine, in Sicilia e in Sardegna.

Da allora la Croce Rossa Italiana è sempre presente e attiva nei conflitti che vedono impegnata l'Italia, sino alla II Guerra Mondiale, ed è presente su tutto il territorio nazionale dall'alluvione nel Polesine del 1951 alla frana che ha colpito Sarno nel 1998, dai terremoti e le alluvioni che hanno colpito l'Italia in questi ultimi anni, ai soccorsi ai feriti di incidenti.

La Croce Rossa Italiana è oggi un Ente di diritto pubblico con prerogative di carattere internazionale, con lo scopo di assistenza sanitaria e sociale sia in tempo di pace che in tempo di guerra. E' posta sotto l'alto patronato del Presidente della Repubblica, sottoposta alla vigilanza dello Stato e sotto il controllo del Ministero della Sanità e del Ministero della Difesa per quanto di sua competenza.

E' un'associazione di soccorso volontaria senza scopo di lucro che ha per missione, in tempo di pace, di recare assistenza alla popolazione, integrando l'azione dello Stato e organizzando soccorsi all'estero mentre, in caso di conflitto, contribuisce con mezzi e personale propri alla sgombero ed alla cura dei feriti con l'allestimento di ospedali militari da campo, posti di pronto soccorso, ambulanze. Organizza la difesa sanitaria, si occupa dello scambio di prigionieri, dello scambio della corrispondenza e pacchi e della ricerca dei dispersi.

5.4 La gestione del soccorso: dalle associazioni di volontariato al 118

Fino all'istituzione del 118 la gestione delle emergenze sanitarie era affidata completamente alle associazioni di volontariato che agivano sul territorio provinciale. Chiaramente questo stato di cose creava non pochi problemi,

soprattutto al cittadino che aveva urgenza di essere trasportato all'ospedale perché non in grado di andarvi da solo. Infatti, prima di trovare un mezzo che potesse effettuare il trasporto, era necessario telefonare ad ogni associazione di volontariato fino a che non si riusciva a trovare qualcuno disponibile e nella possibilità di realizzare l'intervento.

Questo stato di cose era chiaramente preoccupante soprattutto per i pazienti gravi che avevano necessità di essere trasportati immediatamente. La gestione del pronto intervento affidata alle associazioni creava inoltre problemi riguardanti i tempi di intervento, il trasporto negli ospedali più adeguati e le professionalità operanti sulle ambulanze.

Con l'istituzione del servizio di pronto intervento sanitario nazionale, il quale è gestito regionalmente ma fa capo al SSN, diventa compito del 118 tutelare la salute dei cittadini garantendo a chiunque si trovi in una situazione di emergenza sanitaria, reale o potenziale, un intervento finalizzato a portare sul posto, interagendo con tutti gli altri enti non sanitari se necessario, il soccorso più qualificato con i mezzi disponibili sul territorio e favorendo una corretta ospedalizzazione. Grazie alla creazione del numero unico per le emergenze i cittadini sanno esattamente qual è il numero per ricevere soccorso senza doversi preoccupare di null'altro; sarà la centrale operativa ad occuparsi di inviare il mezzo di soccorso idoneo in base ai mezzi disponibili in quella zona ed in base alla criticità dell'intervento.

Il 118 non ha mezzi di proprietà, ma questi sono delle associazioni di volontariato, che li mettono a disposizione in base alla loro disponibilità di volontari. Le convenzioni tra 118 e associazioni di volontariato sono di due tipi, estemporanea e H24. Il primo tipo di convenzione mette a disposizione del 118 le ambulanze solo in alcuni giorni ed in alcune fasce orarie, mentre con il secondo tipo di convenzione le ambulanze sono a disposizione della centrale operativa 24 ore su 24. Questo tipo di convenzione risulta obbligatoria

per i mezzi di soccorso avanzato (MSA). Le convenzioni di disponibilità dei mezzi non riguardano solo le fasce orarie ma anche lo stato di manutenzione degli stessi, infatti le associazioni devono garantire oltre alla disponibilità del mezzo in determinati orari anche la manutenzione dello stesso e, se uno di questi non è disponibile, devono garantire un mezzo sostitutivo.

Secondo il DPR del 27 marzo 1992 le associazioni di volontariato convenzionate non possono pubblicizzare il proprio numero di telefono per i servizi di emergenza, questo al fine di far diventare il 118 l'unico riferimento per il cittadino in caso di emergenza sanitaria.

Un altro aspetto molto importante dell'avvento del 118 è la creazione di professionalità sulle ambulanze e il limitarsi dei tempi di intervento sul territorio, i quali devono essere entro otto minuti per gli interventi in città e venti minuti per gli interventi che si effettuano in provincia. Ultimo aspetto da sottolineare è come il 118 coordini, oltre alle emergenze sanitarie, anche il trasporto straordinario di organi e di sangue garantendo l'arrivo di questi in tempo utile per poter essere ancora utilizzati.

5.5 Il 118

Nel 1986 il Ministero della Sanità, su proposta del Consiglio Sanitario Nazionale (nel quale sono rappresentate tutte le Regioni), prese l'iniziativa di istituire, in analogia con quanto già attuato in molti altri Paesi, il numero telefonico per l'emergenza sanitaria.

Obiettivo della Centrale Operativa 118, secondo quanto definito nell' Atto di indirizzo e coordinamento alle Regioni per la determinazione dei livelli di assistenza sanitaria emanato, come già detto, con Decreto del Presidente della Repubblica il 27 marzo 1992, è nell'ambito di una riorganizzazione generale più funzionale dei servizi preposti all'emergenza sanitaria.

Il raggiungimento di tale obiettivo si ottiene attraverso l'attivazione di

una serie di procedure che interessano diverse strutture e diversi soggetti i quali, per operare in maniera coordinata necessitano di un coordinamento comune, che quindi deve conoscere le caratteristiche, le potenzialità e le esigenze operative delle parti coinvolte.

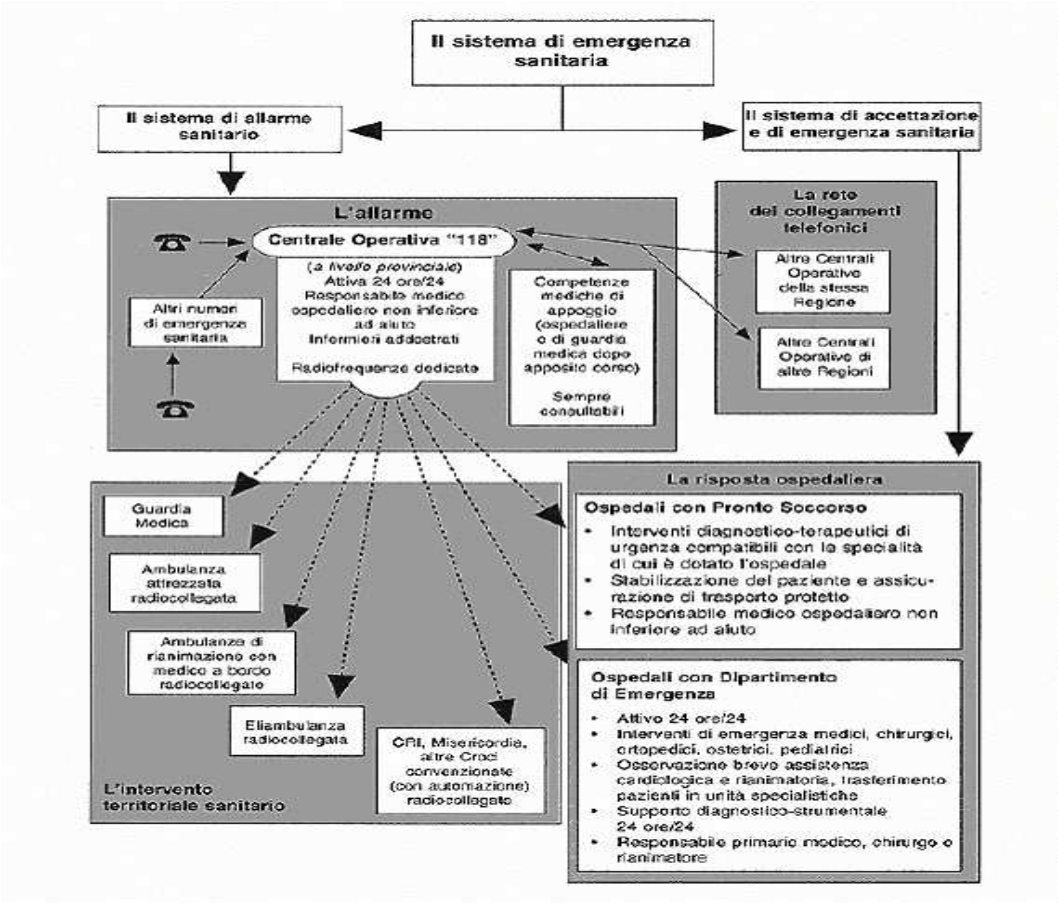


Figura 5.1: Il 118 come progettato nel DPR del 27 Marzo 1992

Con il Decreto del Presidente della Repubblica del 27 Marzo 1992, il 118 si configura come l'unico numero di riferimento per quanto riguarda il pronto intervento sanitario, aggiungendosi ad un vasto panorama fino ad allora formato esclusivamente da associazioni locali e volontarie.

Con il DPR il 118 è investito della capacità di cancellare d'ufficio tutti gli altri numeri che si riferiscono ad associazioni di pronto intervento, dirottando

l'intero traffico telefonico sulla propria linea. Questo potere è esplicitamente illustrato nell'articolo tre, comma tre della legge, il quale cita:

> L'attivazione della centrale operativa comporta il superamento degli altri numeri di emergenza sanitaria di enti, associazioni e servizi delle unità sanitarie locali nell'ambito territoriale di riferimento, anche mediante convogliamento automatico sulla centrale operativa del 118.

Durante questi anni sono state adottate principalmente due strutture organizzative, delle quali la prima è quella che gli addetti ai lavori chiamano *gestione verticale*. Questo modello organizzativo è adottato in presenza di modesti volumi di traffico (intesi come numero di richieste di assistenza). Il funzionamento di questo tipo di struttura è piuttosto semplice: la chiamata arriva direttamente ad un infermiere che, dopo aver interrogato il paziente per poter effettuare una valutazione del caso e assegnarvi un codice di urgenza, si vede costretto a interrompere il contatto per allertare un mezzo di soccorso. E' chiaro come in questo tipo di organizzazione vi sia solo una minima possibilità di effettuare interventi di primo soccorso telefonico, che in alcuni casi potrebbero risultare vitali.

La seconda possibile struttura organizzativa è quella denominata a *gestione orizzontale* e costituisce il modello organizzativo adottato dalla centrale operativa di Grugliasco, dove il traffico si aggira intorno alle 600.000 chiamate annue.

5.5.1 La centrale operativa di Grugliasco

La gestione della centrale operativa avviene come segue: la chiamata arriva al POF (Posto Operatore Filtro) il quale ha come unico compito ha quello di smistare le chiamate destinate alla guardia medica (che si trova fisicamente all'interno della struttura della centrale operativa anche se si tratta

di una realtà distinta ed autonoma rispetto a questa) da quelle indirizzate al 118, inoltrando queste ultime alle PVS (Postazioni di Valutazione Sanitaria). Per effettuare queste operazioni gli operatori assegnati al POF devono avere soprattutto competenze tecniche e minime competenze infermieristiche.

Dopo essere passata dal POF la chiamata arriva alle PVS nelle quali si effettua una valutazione dello stato del paziente e dove si compila una scheda che viene inoltrata al box ambulanze. Gli infermieri che si trovano alla postazione di valutazione possono prestare i primi interventi, cercando di far eseguire alcune semplici operazioni di medicina di base ai presenti sul luogo dell'incidente, questi *interventi a distanza* in alcuni casi possono risultare vitali. L'opportunità di soffermarsi maggiormente in linea, dando istruzioni sul comportamento da tenersi, è data dal fatto che l'allertamento del mezzo e la sua gestione sono affidati ad un'altra postazione e a persone diverse. Gli operatori che si occupano di questa fase del pronto intervento devono necessariamente essere infermieri professionali.

Le PVS generano degli eventi ai quali sono assegnati dei codici di gravità [2] e che vengono tempestivamente trasmessi al box ambulanze, nel quale si trovano due operatori tecnici; uno di questi si occupa di gestire gli interventi effettuati nella provincia di Torino mentre l'altro si occupa degli interventi effettuati nella città di Torino. Questi operatori svolgono le operazioni necessarie a gestire le comunicazioni tra ambulanza e centrale operativa; ad affiancarli c'è un infermiere professionale che interviene nel caso in cui vadano prese importanti decisioni sanitarie quali, per esempio, la scelta dell'ospedale, o l'invio di un altro mezzo nel caso vi sia stata una sottostima della gravità dell'intervento. Dopo la scelta del mezzo l'evento diviene una *missione operativa*, ed ogni missione si compone di otto step:

[2] Si possono leggere in appendice i codici riguardanti la criticità, la patologia ed il luogo dell'incidente.

- Ricerca mezzo: l'operatore si avvale dell'aiuto del computer il quale è in grado di fornire un ventaglio di scelte riguardo ai mezzi disponibili.

- Allarme al mezzo: l'ambulanza viene allertata dall'operatore il quale si preoccupa anche di dare i codici riguardanti la missione, questi codici sono composti da numeri che indicano la criticità dell'intervento, numeri che indicano la patologia e lettere che indicano il luogo [3].

- Partenza: il mezzo parte alla volta del luogo da dove è stata richiesta assistenza. Da qui comincia la parte della routine di soccorso durante la quale è presente un costante contatto radio con la centrale operativa al quale viene assegnato il nome monitoraggio. Il monitoraggio fa parte degli step di cui sopra e si compone di alcune comunicazioni che avvengono tra centrale operativa e mezzo che sta effettuando il soccorso.

- Sul posto: il mezzo segnala che si trova sul luogo dell'incidente.

- Paziente a bordo, in partenza: il mezzo segnala che il paziente si trova a bordo e che si sta partendo alla volta dell'ospedale più idoneo.

- Arrivo all'ospedale

- Di nuovo operativi: qui finisce la fase di monitoraggio ed il mezzo dopo le ordinarie operazioni di pulizia e di riattrezzaggio si rende nuovamente disponibile per nuove missioni.

- Chiusura servizio: in questo stadio l'equipaggio del mezzo contatta la centrale operativa per aggiornare definitivamente la banca dati per quanto riguarda il paziente e quindi per chiudere definitivamente il caso.

[3] Tutti i codici di gravità e di localizzazione si trovano in appendice.

Il 118 dispone di tre tipi di mezzi di soccorso: MSA (mezzo di soccorso avanzato) sul quale sono presenti 1 medico, 1 infermiere e 2 soccorritori volontari, MSAB (mezzo di soccorso avanzato di base) sul quale sono presenti 1 infermiere e 2 soccorritori volontari e l'MSB (mezzo di soccorso di base) sul quale sono presenti 2 soccorritori volontari. Nell'area urbana operano 5 MSA, 1 MSAB e 9 MSB. Nella provincia operano 14 MSA, 9 MSAB e 76 MSB. I mezzi appartengono alle associazioni di volontariato (es. Croce Rossa, Croce Verde ecc.) e vengono messi a disposizione del 118 in base ai due tipi di convenzioni elencati precedentemente:H24 ed estemporanea. È chiaro come queste diverse coperture incidano molto sul servizio di pronto intervento: potrebbe infatti capitare che l'ambulanza più vicina non sia disponibile in quanto è un mezzo in estemporanea. Dopo questo breve excursus sul modo di operare della centrale operativa proviamo a schematizzare quanto detto in Figura 2.

5.6 I dati forniti dalla centrale operativa e la creazione del 118 virtuale

In seguito alle visite compiute presso la sede della centrale operativa di Grugliasco ci è stato fornito un database, inerente all'anno 2002, che contiene i dati di tutti gli interventi effettuati dalle ambulanze in quell'anno. Il database è creato tramite Microsoft Access ed è composto dai seguenti campi:

- ID scheda

  In questo campo sono inseriti dei numeri in ordine crescente ed ognuno di essi corrisponde ad un caso.

- Collegata

  In questa colonna sono inseriti i numeri delle schede che richiedono

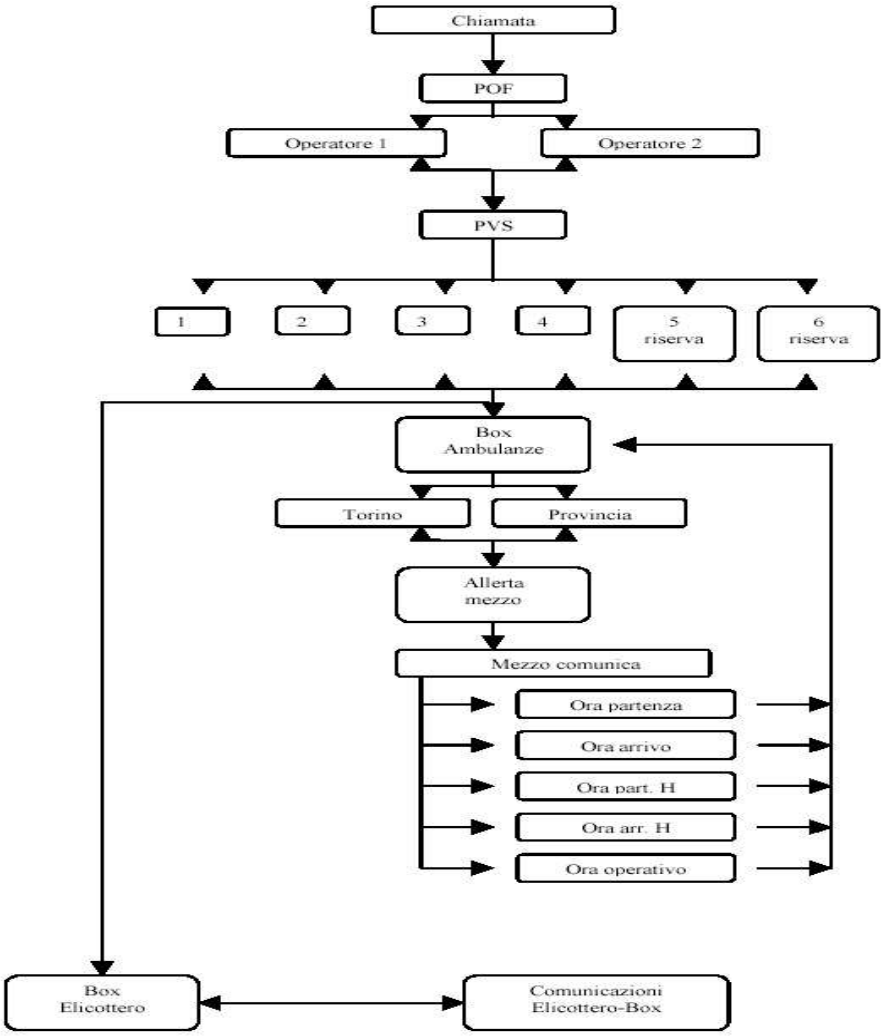


Figura 5.2: Funzionamento centrale operativa 118 di Grugliasco

l'invio di più ambulanze. Qualora il caso sia particolarmente grave, tale da richiedere che vengano mandati sul luogo dell'incidente più mezzi di soccorso, alla centrale operativa vengono create più schede legate tra di loro. Il legame tra le schede viene registrato sul database tramite il riempimento del campo Collegata con il numero della prima scheda generata dall'evento.

Esemplifichiamo attingendo un caso dal database.

Interventi 2002			
IDScheda	Collegata	DataScheda	OraScheda
000041	000041	01/01/02	01:25
000045	000041	01/01/02	01:25
000046	000041	01/01/02	01:25

Tabella 5.1: Dal database fornito dalla centrale operativa

Come si può notare la scheda 000041 ha generato un evento che richiede l'intervento di più di un'ambulanza, quindi gli operatori della centrale operativa hanno generato due schede successive collegate alla scheda in questione. Il collegamento è visibile perché nella colonna chiamata Collegata compare il numero della scheda generatrice del caso.

- Data scheda / Ora scheda

  Sono i campi in cui vengono inseriti l'ora della scheda, che corrisponde all'ora in cui la chiamata viene passata dal POF al PVS, e la data della scheda, che si riferisce alla data in cui è avvenuto l'incidente.

- Chiamante località/ Chiamante indirizzo/ Chiamante numero

  Qui sono inseriti i dati geografici che servono per indicare dove si trova la persona da trasportare.

- Tipo di risposta

  In questo campo sono inserite lo risposte che la centrale operativa fornisce. In alcuni casi, la centrale operativa non invia un mezzo di soccorso ma fornisce una risposta telefonica al chiamante. Questo capita quando gli infermieri professionali operativi sul POF non ritengono necessario l'invio di un mezzo.

- Codice mezzo

È il codice di riconoscimento dell'ambulanza, infatti ad ogni mezzo di soccorso corrisponde una sigla formata dalla sigla TO e seguita dal numero.

- Tipo mezzo

  Si riferisce alla distinzione tra Mezzi di Soccorso di Base, Mezzi di Soccorso di Base Avanzati e Mezzi di Soccorso Avanzati.

- Disponibilità

  Riguarda il tipo di convenzione che il 118 ha con le associazioni che sono le proprietarie delle ambulanze. Gli accordi possibili sono solo due, H24 ed estemporanea.

- Postazione

  Indica dove l'ambulanza è sita, questo tipo di informazione è importantissimo per mandare il mezzo di soccorso disponibile più vicino al luogo dell'incidente.

- Associazione

  Segnala a quale associazione appartiene il mezzo di soccorso utilizzato per far fronte all'evento.

- Ora allarme/ Ora partenza/ Ora arrivo

  Sono gli orari che indicano quando il mezzo è stato allertato, quando questo è partito per andare sul luogo dell'incidente e quando è arrivato sul luogo stesso.

- Ora partenza ospedale/ Ora arrivo ospedale

  Indicano quando il mezzo di soccorso giunto sul luogo dell'incidente riesce a ripartire per l'ospedale dopo aver compiuto le normali operazioni di stabilizzazione del malato, e quando arriva all'ospedale.

- Ora operativo

  Mostra l'orario in cui l'ambulanza è nuovamente operativa e quindi il momento in cui questa può essere riutilizzata per altri interventi.

- Cod. criticità rientro/ Cod. patologia rientro/ Cod. luogo

  Indicano quanto è grave il malato, il tipo di patologia che lo affligge ed il luogo in cui si è manifestata la patologia.

- Cod. rientro

  È riassuntivo di tutti i codici precedenti ed è strutturato come nell'esempio

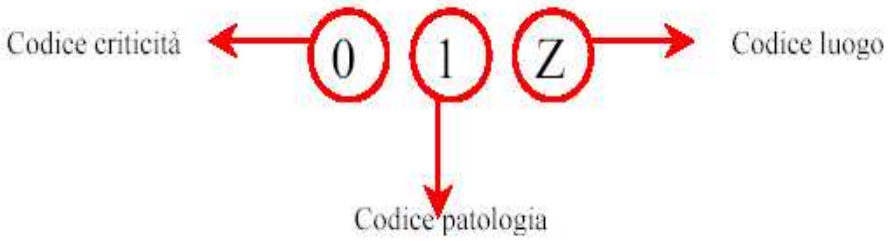


Figura 5.3: Struttura del codice di rientro

- Esito

  Indica se il trasporto è stato effettuato o meno, le motivazioni di un mancato trasporto sono indicate nelle due voci successive.

- Descrizione non trasportato/Descrizione non effettuato

  Illustra il motivo per cui il trasporto non è stato effettuato.

- Codice mezzo trasporto

  Indica il mezzo che ha effettivamente effettuato il trasporto, questo perché quando si hanno casi dove sono stati allertati diversi mezzi può

capitare che solo uno, o solo alcuni di essi, effettuino il trasporto mentre gli altri mezzi tornano indietro senza trasportare alcun paziente.

- Ospedale trasportato

  Indica dove il paziente è stato trasportato.

Dopo aver ottenuto il database è stata fatta un'iniziale analisi sui dati complessivi per conoscere e comprendere l'utilizzo dei mezzi sul territorio, evidenziando eventuali disparità e quindi possibili margini di miglioramento attraverso l'ottimizzazione dell'impiego dei mezzi di soccorso.

Per ciò che riguarda Torino è emerso immediatamente che durante l'anno 2002 vi sono stati alcuni mezzi utilizzati più frequentemente rispetto ad altri.

Per poter effettuare un analisi di questo tipo sono stati raccolti dal database i dati relativi ai mezzi di soccorso di Torino e si è poi calcolato il numero di interventi realizzati da ogni singola ambulanza. Dopo la raccolta dei dati è stata calcolata una media giornaliera di utilizzo tramite una semplice divisione:

$$\text{n°interventi} : 365$$

Come precedentemente detto, vi sono alcune ambulanze che vengono usate molto meno di altre. Tolti i casi dei mezzi estemporanei, la cui differenza di utilizzo potrebbe essere dovuta alle fasce orarie in cui si rendono disponibili, la differente frequenza di utilizzo degli altri mezzi può essere solo parzialmente spiegata dalla dislocazione geografica dei mezzi.

Dopo aver calcolato la media di interventi giornalieri per i mezzi che operano su Torino possiamo procedere alla costruzione del grafico a torta nel quale verranno inserite anche le percentuali di utilizzo di ogni mezzo.

Tali percentuali non fanno che rafforzare l'impressione di quanto già detto sopra, ossia che l'utilizzo dei mezzi di soccorso sia piuttosto disomogeneo.

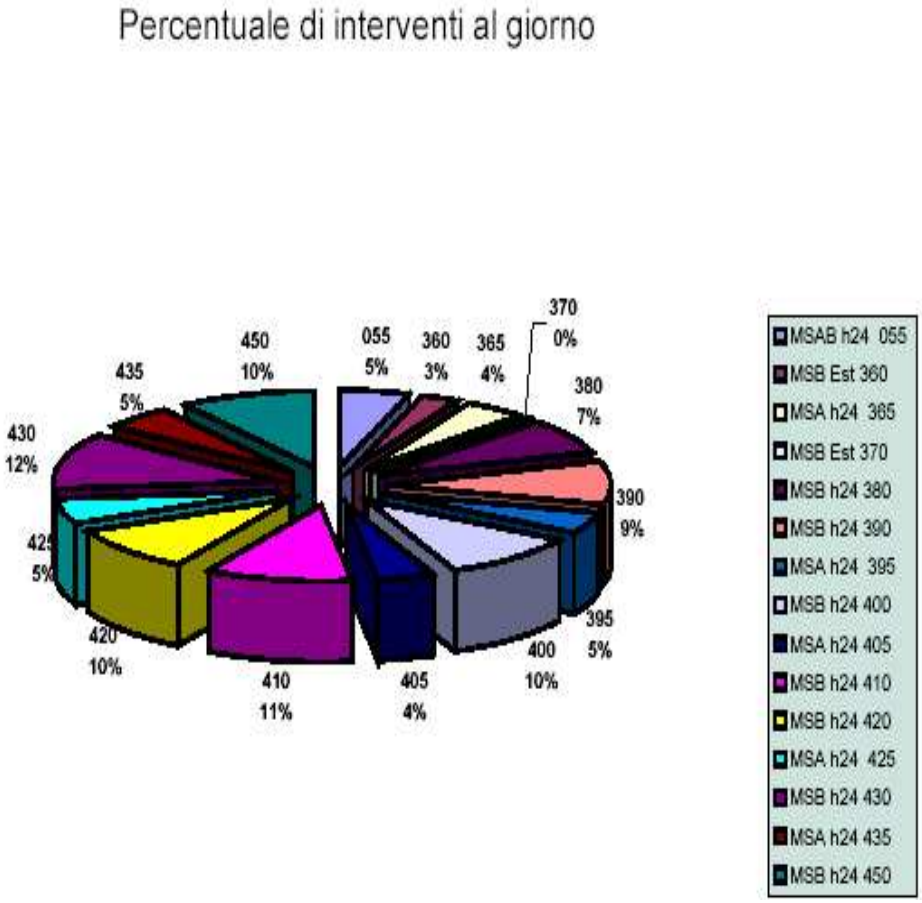


Figura 5.4: Grafico di utilizzo dei mezzi di soccorso su Torino

Dopo aver sommariamente visionato i grafici ed aver evidenziato questo utilizzo disomogeneo dei mezzi di soccorso si è giunti alla conclusione che sarebbe stato meglio ricavare dal database un giorno normale, intendendo per normale un giorno in cui non vi siano stati picchi di chiamate verso l'alto o verso il basso.

Per giungere a questo risultato sono stati estrapolati dal database tutti gli interventi dell'anno suddividendoli in mesi e giorni e raggruppando i dati in un foglio Excel.

Dopo aver suddiviso in questo modo il numero di interventi effettuati durante l'anno abbiamo calcolato quante chiamate sono state gestite dalla

centrale operativa durante ogni settimana. Questa operazione è stata effettuata tramite una somma per riga. I risultati ottenuti sono riassunti in un'altra tabella, nella quale abbiamo anche calcolato la media di interventi giornalieri nelle settimane in questione. Il calcolo è stato effettuato in modo molto semplice:

n°interventi settimanali : n° effettivi giorni di lavoro

E' stato utilizzato il numero di giorni effettivi lavorati perché la prima e l'ultima settimana del mese risultavano spesso interrotte e si voleva evitare che ciò incidesse sul numero medio di interventi giornalieri.

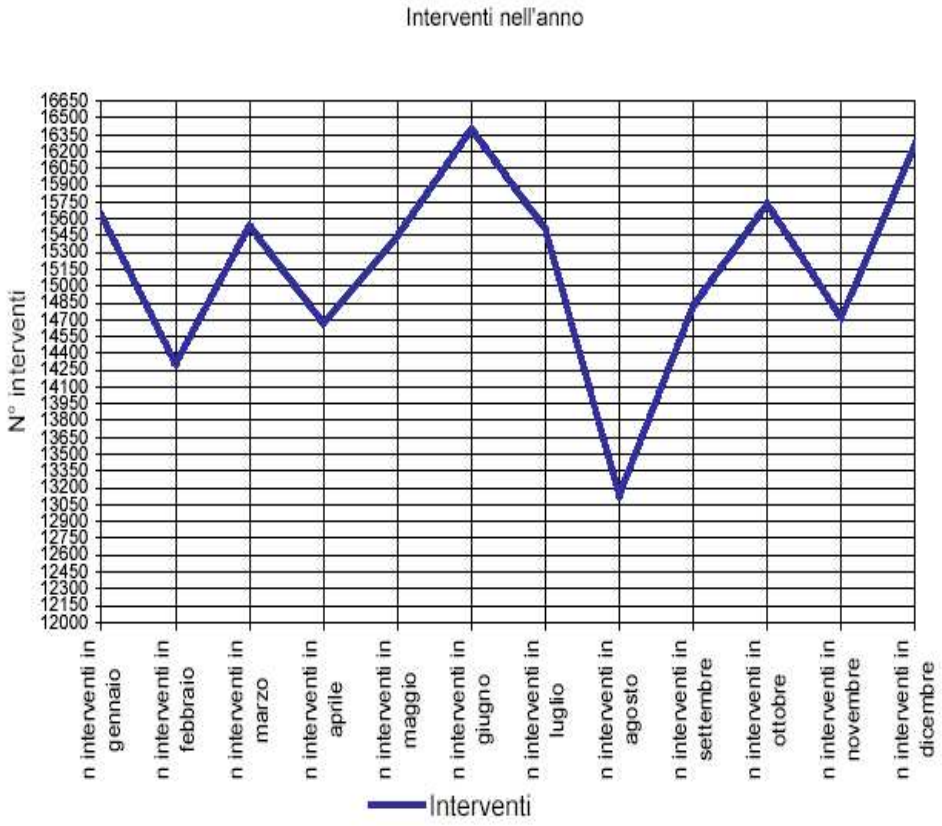


Figura 5.5: Trend degli interventi durante l'anno 2002

Dal grafico è evidente come vi siano dei picchi di interventi in alcuni mesi piuttosto che in altri. Per scegliere il giorno su cui costruire il modello si è analizzato quanto gli effettivi interventi si discostavano dalla media generale, così calcolata:

$$\text{Tot. Int.} : \text{gg. dell'anno.}$$

Dopo un'attenta analisi siamo giunti ad isolare un giorno che è il 23 Ottobre 2002, il quale risulta essere un giorno leggermente al di sotto della media .

Dopo aver stabilito il giorno rappresentativo per la costruire del modello, si è cominciato ad elaborare i dati in modo che potessero essere utili al fine di rappresentare i fatti realmente accaduti. A questo scopo è stato importante notare come i dati fossero ampiamente incompleti per quanto riguarda gli orari delle varie fasi presenti nel database e si è pensato ad un modo per poter approssimare la realtà. Dopo una discussione si è giunti alla conclusione che per i nostri scopi si sarebbero potute usare le cartine presenti tra il materiale fornito dalla centrale operativa del 118.

L'idea fu quella di suddividere le cartine in quadrati in modo che fosse possibile elaborare delle buone approssimazioni sui tempi di percorrenza; questo perché tra i dati è stato riscontrato che, oltre all'orario dell'apertura della scheda e a quello della chiusura del servizio, spesso è presente la voce di arrivo sul posto dell'incidente delle ambulanze e, dato che queste hanno l'abitudine di ritornare al luogo di partenza se sono ubicate presso gli ospedali è possibile che il tempo di percorrenza sia simile, e quindi anche dove non ci fossero i dati è stato possibile ricavarli tramite il teorema di Pitagora ed il calcolo delle distanze.

L'utilizzo delle cartine, tuttavia, ha creato dei problemi nel momento stesso in cui si è cominciato ad analizzarle per suddividerle in quadrati. La cartina che ci era stata fornita per Torino, infatti, era toppo semplificata

per poter essere utilizzata e poter fornire delle buone approssimazioni; per risolvere il problema ci siamo quindi dotati di una cartina della città di scala 1:15.000. Questa è stata inizialmente suddivisa in quadrati da 1 cm che però provocavano problemi di armonizzazione con la cartina della provincia che aveva scala 1:250.000. La soluzione adottata è stata quella di suddividere la cartina della città con quadrati di 1,2 cm, che equivalgono a 180 m, mentre la cartina della provincia con quadrati di 6 cm equivalenti a 15 km.

Capitolo 6

Il progetto del 118: pulizia dei dati, passi computazionali e un primo modello ex-ante

6.1 La pulizia dei dati

Durante la costruzione del modello, si è constatato che i dati presenti nel database fornito dalla centrale operativa di Grugliasco sono spesso incompleti. Dalla figura sottostante si può vedere come per ogni intervento vengano registrate molte informazioni, ma quelle che risultano maggiormente affette da errori sono quelle riguardanti gli orari. Un esempio può aiutare a spiegare quali siano gli errori presenti nel database. Dalla figura 2 si può notare che diversi orari sono uguali. In particolare, l'orario di partenza dall'ospedale, l'orario di arrivo dal paziente e quello di ripartenza verso l'ospedale hanno tutti le 10:27, e l'orario di arrivo all'ospedale e quello in cui l'ambulanza si segnala nuovamente operativa riportano entrambi le 11:11.

Cosa significa questo? La spiegazione è che l'ambulanza non ha comunicato alla centrale operativa l'orario di partenza verso il luogo dell'incidente

IDScheda	OraScheda	ChiamanteLocalita	ChiamanteIndirizzo	TipoRisposta	CodMezzo	TipoMezzo	Disponibilita	Postazione	OraAllarme	OraPartenza	OraArrivo	OraPartenzaH	OraArrivoH	OraOperativo
157403	04:07	BORGARO TORIN	CASELLE	Invio Mezzo	TT0270	MSB	H 24	CASELLE	04:10	04:17	04:17	04:32	04:32	04:32
157404	03:47	BORGOFRANCO	MAZZINI	Invio Mezzo	TT0110	MSB	H 24	IVREA C.R	04:11	04:17	04:26	04:33	04:40	04:56
157405	04:17	BARDONECCHIA	VEDI NOTE	Invio Mezzo	TT0530	MSB	ESTEMPOR	BARDONE	04:21	04:24	04:24	04:24	04:24	04:24
157406	04:22	TORINO	GIULIO CESARE	Invio Mezzo	TT0420	MSB	H 24	TORINO G	04:24	04:25	04:48	04:48	04:48	04:48
157407	04:23	TORINO	VERDI GIUSEPPE	Invio Mezzo	TT0410	MSB	H 24	TORINO M	04:26	05:09	05:09	05:09	05:09	05:09
157408	04:28	TORINO	ROSSELLI CARLO	Invio Mezzo	TT0405	MSA	H 24	TORINO M	04:30	04:34	04:53	04:53	04:53	04:53
157409	04:26	TORINO	MARMOLADA	Risposta Tele										
157410	03:56	VILLASTELLONE	SANTENA	Invio Mezzo	TT0465	MSA	H 24	CARMAGN	04:29	04:45	04:45	05:30	05:30	05:52
157411	04:29	TORINO	SIRACUSA	Invio Mezzo	TT0395	MSA	H 24	TORINO M	04:32	04:37	04:43	05:20	05:26	05:39
157412	04:30	MONCALIERI	CARIGNANO	Invio Mezzo	TT0470	MSB	ESTEMPOR	MONCALIE	04:32	04:36	04:47	04:52	04:52	04:52
157413	04:46	VILLANOVA CAN	MAGNONI	Invio Mezzo	TT0040	MSB	ESTEMPOR	MATHI CAI	04:49	04:51	04:54	05:13	05:13	05:13
157414	04:49	TORINO	VITTORIO VENET	Invio Mezzo	TT0450	MSB	H 24	TORINO G	04:51	04:59	04:59	05:19	05:25	05:34
157499	09:33	BRUINO	PIOSSASCO	Invio Mezzo	TT0320	MSB	ESTEMPOR	PIOSSAS	09:36	09:48	09:48	09:57	09:57	09:57
157500	09:49	SETTIMO TORINE	VERDI GIUSEPPE	Risposta Tele										
157501	09:51	VALENZA	VITTIME DI VIA FA	Risposta Tele										
157502	09:57	RIVAROLO CANA	MARTIRI DI BELFI	Invio Mezzo	TT0100	MSB	H 24	CASTELLA	10:00	10:02	10:10	10:12	10:34	10:34
157503	09:57	CIRIE'	GIORDANO DON L	Invio Mezzo	TT0240	MSB	ESTEMPOR	CIRIE'	10:01	10:02	10:05	10:18	10:23	10:36
157504	09:58	TORINO	SERVAIS GIOVA	Invio Mezzo	TT0430	MSB	H 24	TORINO M	10:08	10:27	10:27	10:27	11:11	11:11
157350	00:03	ROMANO CANAV	MARCONI	Invio Mezzo	TT0180	MSB	ESTEMPOR	CARAVINC	00:06	00:53	00:53	00:53	00:53	00:53
157351	00:03	ROMANO CANAV	MARCONI	Invio Mezzo	TT0115	MSA	H 24	IVREA ASI	00:09	00:21	00:21	00:54	00:54	00:54
157352	00:10	TORINO	BUSCA	Invio Mezzo	TT0390	MSB	H 24	TORINO M	00:14	00:14	00:19	00:19	00:19	00:19
157353	00:11	CHIVASSO	ROMA	Invio Mezzo	TT0650	MSB	ESTEMPOR	CHIVASSC	00:15	00:40	00:40	00:40	00:40	00:40
157354	00:18	TORINO	SAN GIOVANNI BI	Invio Mezzo	TT0425	MSA	H 24	TORINO G	00:20	00:24	01:08	01:08	01:08	01:20
157355	00:19	PAVONE CANAVI	VIGNALE	Invio Mezzo	TT0110	MSB	H 24	IVREA C.R	00:20	00:23	00:41	00:41	01:05	01:05
157356	00:25	TORINO	CASTELLO DI MIR	Invio Mezzo	TT0380	MSB	H 24	TORINO O	00:30	00:39	00:39	00:39	00:47	00:54
157357	00:33	SETTIMO TORINE	VERCELLI	Invio Mezzo	TT0280	MSB	ESTEMPOR	SETTIMO '	00:35	01:20	01:20	01:20	01:20	01:20
157358	00:33	A32 TORINO > BA	VEDI NOTA	Invio Mezzo	TT0520	MSB	H 24 + ESTI	SUSA	00:39	00:40	00:51	01:56	01:56	01:56
157359	00:33	SETTIMO TORINE	VERCELLI	Invio Mezzo	TT0285	MSA	H 24	SETTIMO '	00:39	00:47	00:47	01:39	01:39	02:02
157360	00:43	GRUGLIASCO	GENERAL PEROT	Invio Mezzo	TT0950	MSB	ESTEMPOR	CASCINE '	00:45	00:47	01:09	01:09	01:16	01:34
157361	00:44	LANZO TORINESE	CHALLANT	Invio Mezzo	TT0060	MSB	ESTEMPOR	LANZO TO	00:46	00:48	01:16	01:16	01:16	02:12
157362	01:05	VIGONE	VEDI NOTE	Invio Mezzo	TT0660	MSB	H 24	VIGONE	01:07	01:11	01:18	02:24	02:24	02:24
157363	01:07	TORINO	VEROLENGO	Invio Mezzo	TT0435	MSA	H 24	TORINO M	01:10	01:46	01:46	01:46	01:49	02:03
157364	01:10	TORINO	VITTORIO EMANU	Invio Mezzo	TT0400	MSB	H 24	TORINO M	01:12	01:15	01:42	01:42	01:42	01:42
157365	01:11	TORINO	SACCHI PAOLO	Invio Mezzo	TT0410	MSB	H 24	TORINO M	01:15	01:37	01:37	01:37	01:37	01:37
157365	01:11	TORINO	SACCHI PAOLO	Invio Mezzo	TT0410	MSB	H 24	TORINO M	01:15	01:37	01:37	01:37	01:37	01:37
157366	01:15	LA LOGGIA	SAN REMO	Invio Mezzo	TT0475	MSAB	H 24	MONCALIE	01:17	01:29	01:29	01:42	02:01	02:01

Figura 6.1: Il database fornito dai responsabili della centrale operativa

e l'orario di arrivo sul luogo. In questo caso l'operatore del box ambulanze segna nel campo corretto l'orario ricevuto, che in questo caso è quello della ripartenza dell'ambulanza verso l'ospedale, ed i campi rimasti vuoti si riempiono automaticamente dello stesso orario. Un caso completo si riconosce invece perché tutti gli orari sono diversi, come si può vedere in figura 3.

Un altro problema riguardante gli orari è che all'interno del database nessuna voce ci dice quando realmente una chiamata giunge alla centrale operativa ma solo quando viene aperta una scheda al PVS. Una prima approssimazione necessaria è dunque quella del tempo impiegato dai POF, che può essere breve nel caso l'utente trovi immediatamente un operatore libero,

IDScheda	OraScheda	OraAllarme	OraPartenza	OraArrivo	OraPartenzaH	OraArrivoH	OraOperativo
157504	09:58	10:08	10:27	10:27	10:27	11:11	11:11

Figura 6.2: Un esempio di dati incompleti

IDScheda	OraScheda	OraAllarme	OraPartenza	OraArrivo	OraPartenzaH	OraArrivoH	OraOperativo
157385	02:36	02:39	02:43	02:58	03:06	03:22	03:46

Figura 6.3: Un esempio di dati completi

ma che può anche essere un po' più lungo nel caso in cui tutti i PVS siano occupati e l'operatore filtro sia costretto a inoltrare la chiamata manualmente. Il grafico sottostante [1] rappresenta questa situazione.

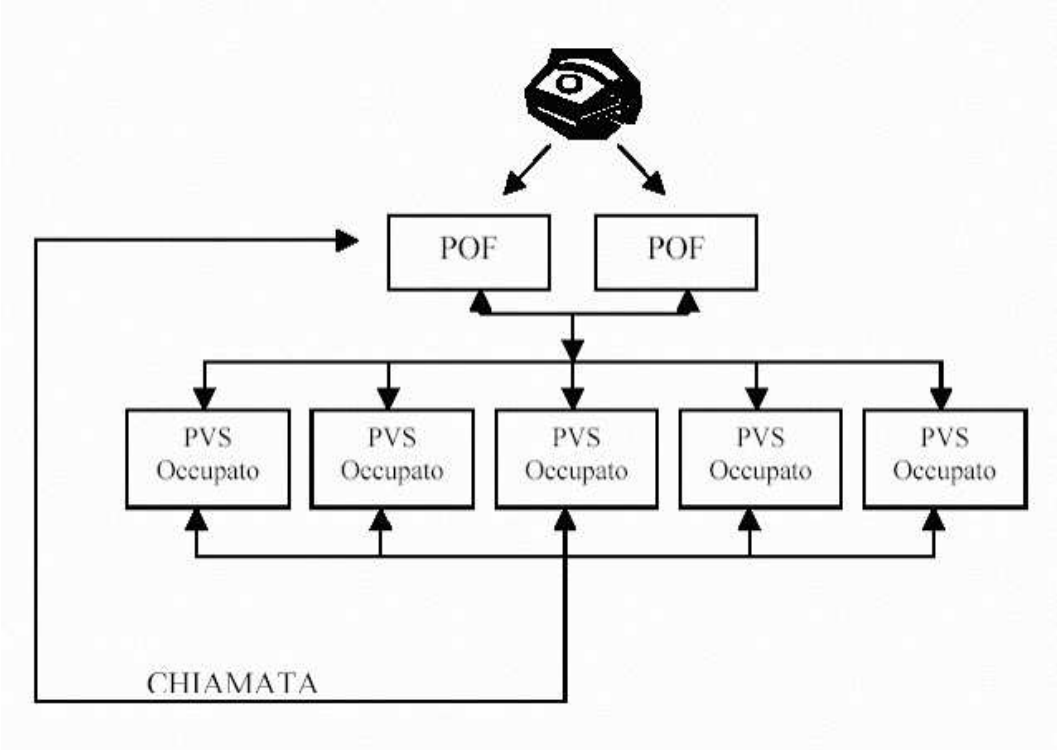


Figura 6.4: Iter della chiamata nel caso le linee PVS siano tutte occupate

I responsabili della centrale operativa hanno comunque assicurato che tale tempo di attesa non è mai superiore ai due minuti e quindi l'errore di

[1] La figura è stata presa dalla tesi di Guerra(2004)

approssimazione commesso non dovrebbe essere rilevante. Per poter costruire il modello ex-post, ma soprattutto per lo sviluppo della simulazione e la creazione di modelli ex-ante, si è dovuti ricorrere a delle approssimazioni anche sui tempi intermedi dell'ambulanza; questo perché ci si è accorti che le indicazioni sull'orario di apertura della scheda, quello di allarme dell'ambulanza e l'orario di operatività dell'ambulanza erano sempre presenti. Un primo rimedio è stato quindi quello di fare una media dei tempi disponibili per ottenere quelli mancanti. Riprendendo l'esempio precedente, ecco come sono stati ottenuti i tre orari mancanti:

IDScheda	OraScheda	OraAllarme	OraPartenza	OraArrivo	OraPartenzaH	OraArrivoH	OraOperativo
157504	09:58	10:08			10:27		11:11

Figura 6.5: Caso incompleto

per calcolare gli orari di partenza e arrivo dell'ambulanza dal paziente si è effettuato il seguente calcolo:

$$\frac{OraPartenzaH - OraAllarme}{3} = \frac{10:27 - 10:08}{3} = 6,3 \qquad (6.1)$$

Partendo dalle 10:08 dell'orario di allarme, verranno quindi aggiunti sei minuti per ricavare l'orario di partenza ed altri sei per quello di arrivo. I campi incompleti saranno così diventati:

IDScheda	OraScheda	OraAllarme	OraPartenza	OraArrivo	OraPartenzaH	OraArrivoH	OraOperativo
157504	09:58	10:08	10.14	10.20	10:27		11:11

Figura 6.6: Completamento di un caso incompleto

Ovviamente lo stesso discorso è stato fatto sia per calcolare l'orario di arrivo dell'ambulanza all'ospedale in questo caso, che in tutti i casi in cui i dati fossero incompleti.

Tutto ciò è servito per avere un primo modello funzionante ex-post, ma si è subito ovviamente pensato a come migliorare la situazione. Si è quindi deciso di utilizzare delle cartine di Torino e della provincia di Torino per poter calcolare la distanza tra ospedale e paziente ed ottenere così le velocità degli interventi rilevati in maniera completa. Facendone una media si poteva poi applicare la velocità media alle distanze percorse ottenendo così dei tempi intermedi più verosimili di quelli precedentemente calcolati.

Dopo aver riprodotto le cartine su computer è cominciato il lavoro di rilevamento delle coordinate di riga e colonna per ogni intervento, ma i gli interventi rilevati correttamente erano così pochi che la velocità media non sarebbe stata attendibile; si è allora deciso di utilizzare i dati di due settimane in modo da avere un campione di dati sufficiente ad avere una stima attendibile della velocità media delle ambulanze. Vista la mole dati da analizzare il lavoro è stato diviso in due. Per quanto riguarda la città di Torino i due gruppi hanno rilevato un esito simile e, su un campione di 200 dati, la velocità media riscontrata è stata all'incirca di 25 km/h.

Per quanto riguarda la provincia di Torino è stato impossibile utilizzare lo stesso metodo perché non è stata trovata una cartina abbastanza dettagliata. L'idea[2] è stata quindi quella di trovare un portale internet che fornisse la distanza percorsa dalle ambulanze; il portale scelto è stato tuttocittà[3] grazie al quale si è potuto calcolare la velocità media delle ambulanze fuori Torino. In questo caso l'esito è stato abbastanza differente tra i due gruppi, perché il primo ha rilevato una velocità media di 39 km/h, mentre il secondo di 51 km/h. Tale differenza può essere spiegata dalla eterogeneità di situazioni degli interventi nella provincia di Torino, in cui a volte gli interventi si svolgono completamente all'interno di un paese, mentre altre volte c'è un'elevata distanza tra ospedale e malato. Si è quindi giunti alla decisione di considera-

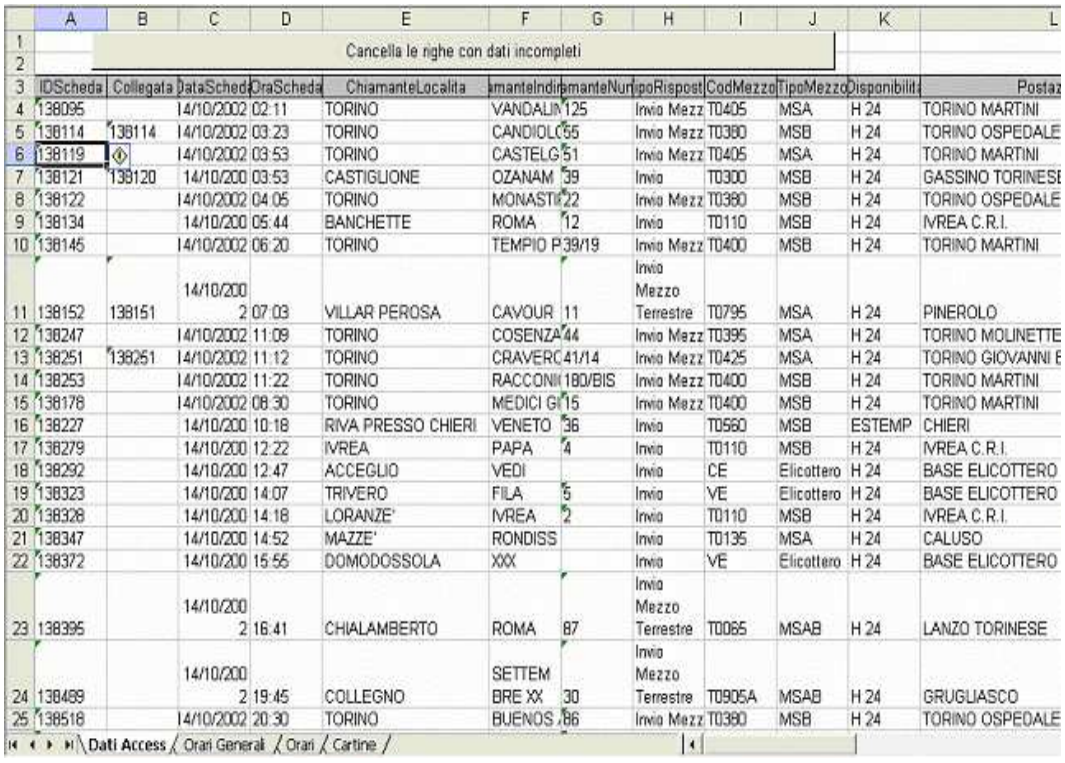
[2]L'idea è da attribuire al dottor Raimondi

[3]www.tuttocittà.it

re come velocità di riferimento la media dei due risultati che quindi diventa 45 km/h.

6.1.1 Le macro

La mole di dati necessaria per ottenere delle stime attendibili sulla velocità media delle ambulanze fuori e dentro Torino ha reso necessario l'utilizzo del programma Microsoft Excel per calcolare distanze e velocità. Il file riguardante la pulizia dei dati si compone di quattro fogli di lavoro. Nel primo, denominato Dati Access sono presenti i dati raccolti nel database fornito dalla centrale del 118; in cima al foglio c'è un bottone che cancella gli interventi i cui dati sono incompleti.



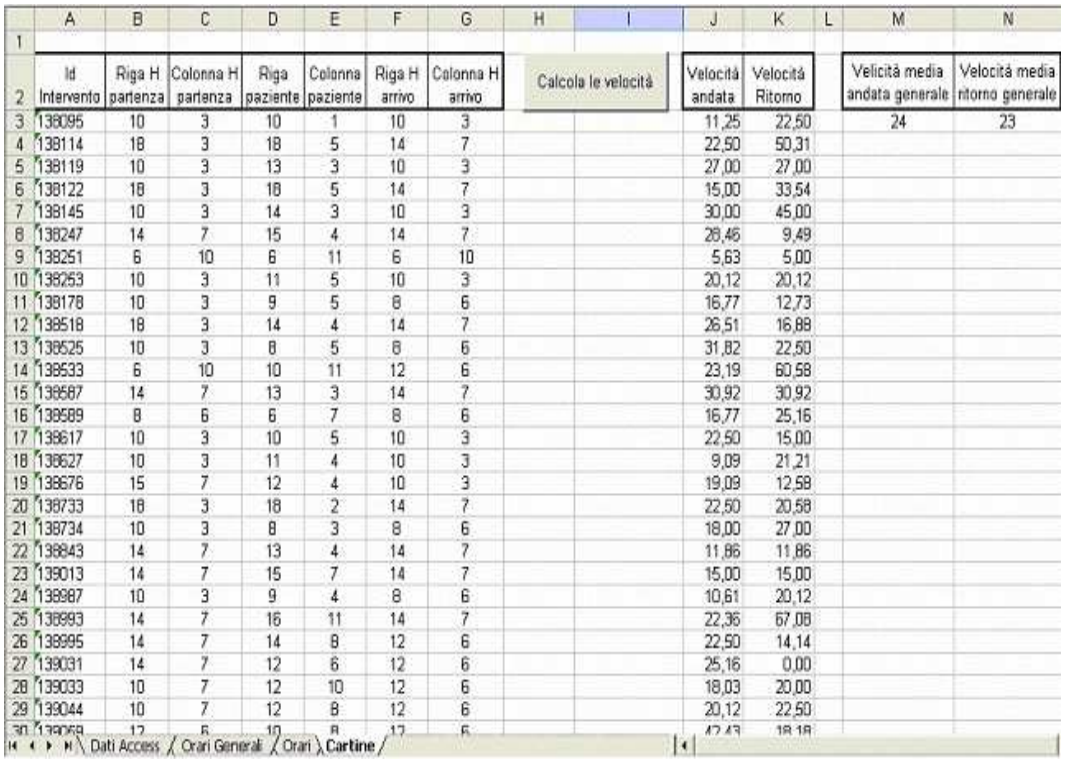
	A	B	C	D	E	F	G	H	I	J	K	L
1					Cancella le righe con dati incompleti							
2												
3	IDScheda	Collegata	DataScheda	OraScheda	ChiamanteLocalita	amanteIndir	amanteNur	ipoRispost	CodMezzo	TipoMezzo	Disponibilit	Postaz
4	138095		14/10/2002	02:11	TORINO	VANDALIN	125	Invio Mezz	T0405	MSA	H 24	TORINO MARTINI
5	138114	138114	14/10/2002	03:23	TORINO	CANDIOLO	55	Invio Mezz	T0380	MSB	H 24	TORINO OSPEDALE
6	138119		14/10/2002	03:53	TORINO	CASTELG	51	Invio Mezz	T0405	MSA	H 24	TORINO MARTINI
7	138121	138120	14/10/200	03:53	CASTIGLIONE	OZANAM	39	Invio	T0300	MSB	H 24	GASSINO TORINESE
8	138122		14/10/2002	04:05	TORINO	MONASTI	22	Invio Mezz	T0380	MSB	H 24	TORINO OSPEDALE
9	138134		14/10/200	05:44	BANCHETTE	ROMA	12	Invio	T0110	MSB	H 24	IVREA C.R.I.
10	138145		14/10/2002	06:20	TORINO	TEMPIO P	39/19	Invio Mezz	T0400	MSB	H 24	TORINO MARTINI
11	138152	138151	14/10/200 2	07:03	VILLAR PEROSA	CAVOUR	11	Invio Mezzo Terrestre	T0795	MSA	H 24	PINEROLO
12	138247		14/10/2002	11:09	TORINO	COSENZA	44	Invio Mezz	T0395	MSA	H 24	TORINO MOLINETTE
13	138251	138251	14/10/2002	11:12	TORINO	CRAVERC	41/14	Invio Mezz	T0425	MSA	H 24	TORINO GIOVANNI E
14	138253		14/10/2002	11:22	TORINO	RACCONI	180/BIS	Invio Mezz	T0400	MSB	H 24	TORINO MARTINI
15	138178		14/10/2002	08:30	TORINO	MEDICI GI	15	Invio Mezz	T0400	MSB	H 24	TORINO MARTINI
16	138227		14/10/200	10:18	RIVA PRESSO CHIERI	VENETO	36	Invio	T0560	MSB	ESTEMP	CHIERI
17	138279		14/10/200	12:22	IVREA	PAPA	4	Invio	T0110	MSB	H 24	IVREA C.R.I.
18	138292		14/10/200	12:47	ACCEGLIO	VEDI		Invio	CE	Elicottero	H 24	BASE ELICOTTERO
19	138323		14/10/200	14:07	TRIVERO	FILA	5	Invio	VE	Elicottero	H 24	BASE ELICOTTERO
20	138328		14/10/200	14:18	LORANZE'	IVREA	2	Invio	T0110	MSB	H 24	IVREA C.R.I.
21	138347		14/10/200	14:52	MAZZE'	RONDISS		Invio	T0135	MSA	H 24	CALUSO
22	138372		14/10/200	15:55	DOMODOSSOLA	XXX		Invio	VE	Elicottero	H 24	BASE ELICOTTERO
23	138395		14/10/200 2	16:41	CHIALAMBERTO	ROMA	87	Invio Mezzo Terrestre	T0065	MSAB	H 24	LANZO TORINESE
24	138489		14/10/200 2	19:45	COLLEGNO	SETTEM BRE XX	30	Invio Mezzo Terrestre	T0905A	MSAB	H 24	GRUGLIASCO
25	138518		14/10/2002	20:30	TORINO	BUENOS	86	Invio Mezz	T0380	MSB	H 24	TORINO OSPEDALE

Dati Access / Orari Generali / Orari / Cartine /

Figura 6.7: Foglio di Excel: Dati Access

I dati così ottenuti sono copiati nel secondo foglio, nominato Orari Generali; in questo si selezionano, grazie ad un filtro, solo gli interventi compiu-

ti a Torino (o viceversa) e, i dati rimanenti sono copiati nel terzo foglio, chiamato Orari. Nel foglio Cartine, infine, si inseriscono gli indici di riga e colonna rispettivamente di: partenza dell'ambulanza, prelievo del paziente, arrivo dell'ambulanza. Nello stesso foglio, cliccando sul bottone Calcola le velocità, le distanze di tutti gli interventi, le velocità di andata e ritorno delle ambulanze, e la velocità media sono calcolate automaticamente.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	Id Intervento	Riga H partenza	Colonna H partenza	Riga paziente	Colonna paziente	Riga H arrivo	Colonna H arrivo		Calcola le velocità	Velocità andata	Velocità Ritorno		Velicità media andata generale	Velocità media ritorno generale
3	138095	10	3	10	1	10	3			11,25	22,50		24	23
4	138114	18	3	18	5	14	7			22,50	50,31			
5	138119	10	3	13	3	10	3			27,00	27,00			
6	138122	18	3	18	5	14	7			15,00	33,54			
7	138145	10	3	14	3	10	3			30,00	45,00			
8	138247	14	7	15	4	14	7			28,46	9,49			
9	138251	6	10	6	11	6	10			5,63	5,00			
10	138253	10	3	11	5	10	3			20,12	20,12			
11	138178	10	3	9	5	8	6			16,77	12,73			
12	138518	18	3	14	4	14	7			26,51	16,88			
13	138525	10	3	8	5	6	6			31,82	22,50			
14	138533	6	10	10	11	12	6			23,19	60,58			
15	138587	14	7	13	3	14	7			30,92	30,92			
16	138589	8	6	6	7	8	6			16,77	25,16			
17	138617	10	3	10	5	10	3			22,50	15,00			
18	138627	10	3	11	4	10	3			9,09	21,21			
19	138676	15	7	12	4	10	3			19,09	12,58			
20	138733	18	3	18	2	14	7			22,50	20,58			
21	138734	10	3	8	3	8	6			18,00	27,00			
22	138843	14	7	13	4	14	7			11,86	11,86			
23	139013	14	7	15	7	14	7			15,00	15,00			
24	138987	10	3	9	4	8	6			10,61	20,12			
25	138993	14	7	16	11	14	7			22,36	67,08			
26	138995	14	7	14	8	12	6			22,50	14,14			
27	139031	14	7	12	6	12	6			25,16	0,00			
28	139033	10	7	12	10	12	6			18,03	20,00			
29	139044	10	7	12	8	12	6			20,12	22,50			
30	139068	12	6	10	8	12	6			42,43	18,18			

Dati Access / Orari Generali / Orari \ Cartine /

Figura 6.8: Foglio di Excel: Cartine

Per eseguire i calcoli si è fatto ricorso alla programmazione grazie alla possibilità di utilizzare une versione di Visual Basic per i programmi Microsoft:VBA(Visual Basic for Application). Tramite VBA si scrivono delle macro, cioè dei programmi che consentono di automatizzare delle procedure. Ad ogni bottone è assegnata una macro che esegue il piccolo (nel nostro caso) programma associato. Al bottone seguente corrisponde la macro riportata sotto.

Figura 6.9: Bottone corrispondente alla macro che cancella gli interventi incompleti

```
Sub CancellaIncompleti()
'
' CancellaIncompleti Macro
' Macro registrata il 08/05/2004 da
  '
'
Dim i, spia As Integer
    i = 4
    While Worksheets("Dati Access").Range("A" \& i) <> ""
        spia = 0
        If Worksheets("Dati Access").Range("N" \& i) =
                Worksheets("Dati Access").Range("O" \& i) Then
            Worksheets("Dati Access").Rows(i).Delete
            i = i - 1
            spia = 1
        End If
        If Worksheets("Dati Access").Range("O" \& i) = Worksheets
                ("Dati Access").Range("P" \& i) And spia = 0 Then
            Worksheets("Dati Access").Rows(i).Delete
            i = i - 1
            spia = 1
        End If
        If Worksheets("Dati Access").Range("P" \& i) = Worksheets
                ("Dati Access").Range("Q" \& i) And spia = 0 Then
            Worksheets("Dati Access").Rows(i).Delete
            i = i - 1
            spia = 1
        End If
```

```
        If Worksheets("Dati Access").Range("Q" \& i) = Worksheets
                ("Dati Access").Range("R" \& i) And spia = 0 Then
            Worksheets("Dati Access").Rows(i).Delete
            i = i - 1
            spia = 1
        End If
        If Worksheets("Dati Access").Range("R" \& i) = Worksheets
                ("Dati Access").Range("S" \& i) And spia = 0 Then
            Worksheets("Dati Access").Rows(i).Delete
            i = i - 1
        End If
        i = i + 1
    Wend
End Sub
```

Questo programma cancella le righe che contengono dati incompleti dal foglio Excel Dati Access. Per ogni intervento (ogni riga), il programma controlla tutti i campi contenenti gli orari e se trova due valori uguali cancella la riga corrispondente.

Quando, in questa maniera, sono stati selezionati solo i casi completi, si può calcolare la velocità media cliccando sul bottone seguente contenuto nel foglio Cartine.

Figura 6.10: Bottone corrispondente alla macro che calcola la velocità media delle ambulanze

A questo bottone corrisponde la macro sottostante, che calcola le distanze percorse dalle ambulanze, i tempi intermedi impiegati e , da questi due valori, ricava la velocità media.

```
Sub LetturaDati()
```

```
'
' LetturaDati Macro
' Macro registrata il 30/12/2003 da Carlo Desotgiu
'

    ' vettori contenenti i volori di riga e colonna della posizione
    ' di partenza-arrivo delle ambulanze
    Dim posPartenza(10000, 1), posPaziente(10000, 1),
        posArrivo(10000, 1) As Integer

    Dim intervento, i, j, k, oraPartenza(1), oraArrivo(1),
        oraPartenzaH(1), oraArrivoH(1), andata(1), ritorno(1)
        As Integer

    Dim sommaVelocità Andata, sommaVelocità Ritorno, distanzaAndata,
        distanzaRitorno, distanzaRealeAndata, distanzaRealeRitorno
        As Double


    i = j = k = 0
    intervento = Worksheets("Cartine").Range("A2")
    While intervento <> ""
        ' indice di riga sulla cartina della partenza dell'ambulanza
        posPartenza(i, 0) = Worksheets("Cartine").Range("B" \& i + 3)
        ' indice di col sulla cartina della partenza dell'ambulanza
        posPartenza(i, 1) = Worksheets("Cartine").Range("C" \& i + 3)
        ' indice di riga sulla cartina dell'arrivo dell'ambulanza
        posPaziente(i, 0) = Worksheets("Cartine").Range("D" \& i + 3)
        ' indice di colonna sulla cartina dell'arrivo dell'ambulanza
        posPaziente(i, 1) = Worksheets("Cartine").Range("E" \& i + 3)
        posArrivo(i, 0) = Worksheets("Cartine").Range("F" \& i + 3)
        posArrivo(i, 1) = Worksheets("Cartine").Range("G" \& i + 3)

        i = i + 1
```

```
        ' intervento successivo
        intervento = Worksheets("Orari").Range("A" \& i + 3)
    Wend

    i = 0

    intervento = Worksheets("Cartine").Range("A2")

    While intervento <> ""
        ' teorema di Pitagora per calcolare la distanza
        ' in linea d'aria tra partenza e arrivo dell'ambulanza
        distanzaAndata = ((posPartenza(i, 0) - posPaziente(i, 0)) ^ 2
            + (posPartenza(i, 1) - posPaziente(i, 1)) ^ 2) ^ 0.5
        distanzaRitorno = ((posArrivo(i, 0) - posPaziente(i, 0)) ^ 2
            + (posArrivo(i, 1) - posPaziente(i, 1)) ^ 2) ^ 0.5
        ' calcolo della distanza reale con la scala della cartina
        distanzaRealeAndata = distanzaAndata * 0.75
        distanzaRealeRitorno = distanzaRitorno * 0.75
        ' lettura degli orari e trasformazione in minuti e secondi
        ora_Partenza = Worksheets("Orari").Range("O" \& i + 2)
        ' per ogni vettore riguardante gli orari si hanno
        oraPartenza(1) = Minute(ora_Partenza)
        ' le ore nella prima posizione ed i minuti nella seconda
        oraPartenza(0) = Hour(ora_Partenza)
        ora_Arrivo = Worksheets("Orari").Range("P" \& i + 2)
        oraArrivo(1) = Minute(ora_Arrivo)
        oraArrivo(0) = Hour(ora_Arrivo)
        ora_PartenzaH = Worksheets("Orari").Range("Q" \& i + 2)
        oraPartenzaH(1) = Minute(ora_PartenzaH)
        oraPartenzaH(0) = Hour(ora_PartenzaH)
        ora_ArrivoH = Worksheets("Orari").Range("R" \& i + 2)
        oraArrivoH(1) = Minute(ora_ArrivoH)
        oraArrivoH(0) = Hour(ora_ArrivoH)
```

```
' calcolo dei tempi per i tragitti ospedale-malato e
' malato-ospedale
andata(1) = oraArrivo(1) - oraPartenza(1)
andata(0) = oraArrivo(0) - oraPartenza(0)
If andata(1) < 0 Then
    If oraArrivo(0) <> 0 Then
        andata(1) = 60 + andata(1)
        andata(0) = andata(0) - 1
    Else
        andata(1) = 60 + andata(1)
        andata(0) = 0
    End If
End If

' trasformazione delle ore in minuti per il calcolo della
' velocità

andata(0) = andata(0) * 60
ritorno(1) = oraArrivoH(1) - oraPartenzaH(1)
ritorno(0) = oraArrivoH(0) - oraPartenzaH(0)
If ritorno(1) < 0 Then
    If oraArrivoH(0) <> 0 Then
        ritorno(1) = 60 + ritorno(1)
        ritorno(0) = ritorno(0) - 1
    Else
        ritorno(1) = 60 + ritorno(1)
        ritorno(0) = 0
    End If
End If

' trasformazione delle ore in minuti per il calcolo della
' velocità
ritorno(0) = ritorno(0) * 60
```

```
        ' calcolo della velocità di entrambi i tragitti
        velocità Andata = (distanzaRealeAndata / andata(1) +
            andata(0)) * 60
        velocità Ritorno = (distanzaRealeRitorno / ritorno(1) +
            ritorno(0)) * 60

        ' scrive le velocità cacolate nel foglio Cartine
        Worksheets("Cartine").Range("J" \& i + 3) = velocità Andata
        Worksheets("Cartine").Range("K" \& i + 3) = velocità Ritorno

        ' calcola la somma delle velocità per farne poi una media
        If velocità Andata <> 0 Then
            sommaVelocità Andata = sommaVelocità Andata +
                velocità Andata
            j = j + 1
        End If

        If velocità Ritorno <> 0 Then
            sommaVelocità Ritorno = sommaVelocità Ritorno +
                velocità Ritorno
            k = k + 1
        End If

        i = i + 1

        intervento = Worksheets("Cartine").Range("A" \& i + 3)
    Wend

    ' calcola la media delle velocità di andata e ritorno
    Worksheets("Cartine").Range("M3") = sommaVelocità Andata \ j
    Worksheets("Cartine").Range("N3") = sommaVelocità Ritorno \ k

End Sub
```

Nel file contenente gli interventi nella provincia di Torino, allo stesso

bottone è associata una macro diversa, perché le distanze non vengono più calcolate in base agli indici di riga e colonna della cartina, ma vengono scritti direttamente dopo averli ottenuti dal portale Tuttocittà.

La macro è quindi la seguente:

```
Sub LetturaDatiProvincia()

'
' LetturaDati2 Macro
' Macro registrata il 09/09/2004 da Carlo Desotgiu
'
    ' vettori contenenti i volori di riga e colonna della posizione
    ' di partenza-arrivo delle ambulanze
    Dim posPartenza(10000, 1), posPaziente(10000, 1),
        posArrivo(10000, 1) As Integer
    Dim intervento, i, j, k, oraPartenza(1), oraArrivo(1),
        oraPartenzaH(1), oraArrivoH(1), andata(1), ritorno(1)
        As Integer
    Dim sommaVelocità Andata, sommaVelocità Ritorno, distanzaAndata,
        distanzaRitorno,distanzaRealeAndata, distanzaRealeRitorno
        As Double


    i = j = k = 0
    intervento = Worksheets("Cartine").Range("A3")
    While intervento <> ""
        i = i + 1
        ' intervento successivo
        intervento = Worksheets("Cartine").Range("A" \& i + 3)
    Wend

    i = 0
    intervento = Worksheets("Cartine").Range("A2")
```

```
While intervento <> ""
    If Worksheets("Cartine").Range("B" \& i + 3) <> "" Then
    distanzaAndata = Worksheets("Cartine").Range("B" \& i + 4)
    distanzaAndata = distanzaAndata / 1000
    distanzaRitorno = ((posArrivo(i, 0) - posPaziente(i, 0))
      ^ 2 + (posArrivo(i, 1) - posPaziente(i, 1)) ^ 2) ^ 0.5

    ' lettura degli orari e trasformazione in minuti e secondi
    ora_Partenza = Worksheets("Orari2").Range("O" \& i + 4)
    ' per ogni vettore riguardante gli orari si hanno
    oraPartenza(1) = Minute(ora_Partenza)
    ' le ore nella prima posizione ed i minuti nella seconda
    oraPartenza(0) = Hour(ora_Partenza)
    ora_Arrivo = Worksheets("Orari2").Range("P" \& i + 4)
    oraArrivo(1) = Minute(ora_Arrivo)
    oraArrivo(0) = Hour(ora_Arrivo)
    ora_PartenzaH = Worksheets("Orari2").Range("Q" \& i + 4)
    oraPartenzaH(1) = Minute(ora_PartenzaH)
    oraPartenzaH(0) = Hour(ora_PartenzaH)
    ora_ArrivoH = Worksheets("Orari2").Range("R" \& i + 4)
    oraArrivoH(1) = Minute(ora_ArrivoH)
    oraArrivoH(0) = Hour(ora_ArrivoH)

    ' calcolo dei tempi per i tragitti ospedale-malato e
    ' malato-ospedale
    andata(1) = oraArrivo(1) - oraPartenza(1)
    andata(0) = oraArrivo(0) - oraPartenza(0)
    If andata(1) < 0 Then
        If oraArrivo(0) <> 0 Then
            andata(1) = 60 + andata(1)
            andata(0) = andata(0) - 1
        Else
            andata(1) = 60 + andata(1)
            andata(0) = 0
```

```
    End If
End If

' trasformazione delle ore in minuti per il calcolo della
' velocità
andata(0) = andata(0) * 60
ritorno(1) = oraArrivoH(1) - oraPartenzaH(1)
ritorno(0) = oraArrivoH(0) - oraPartenzaH(0)
If ritorno(1) < 0 Then
    If oraArrivoH(0) <> 0 Then
        ritorno(1) = 60 + ritorno(1)
        ritorno(0) = ritorno(0) - 1
    Else
        ritorno(1) = 60 + ritorno(1)
        ritorno(0) = 0
    End If
End If

' trasformazione delle ore in minuti per il calcolo della
' velocità
ritorno(0) = ritorno(0) * 60

' calcolo della velocità  di entrambi i tragitti
velocità Andata = (distanzaAndata / andata(1) + andata(0)) * 60
velocità Ritorno = (distanzaRealeRitorno / ritorno(1) +
   ritorno(0)) * 60

' scrive le velocità  cacolate nel foglio Cartine
Worksheets("Cartine").Range("J" \& i + 4) = velocità Andata
Worksheets("Cartine").Range("K" \& i + 4) = velocità Ritorno

' calcola la somma delle velocità  per farne poi una media
If velocità Andata <> 0 Then
    sommaVelocità Andata = sommaVelocità Andata + velocità Andata
```

```
            j = j + 1
        End If

        If velocità Ritorno <> 0 Then
            sommaVelocità Ritorno = sommaVelocità Ritorno +
                velocità Ritorno
            k = k + 1
        End If
    End If

    i = i + 1
    intervento = Worksheets("Cartine").Range("A" \& i + 4)

    Wend


    ' calcola la media delle velocità  di andata e ritorno e le scrive
    Worksheets("Cartine").Range("M3") = sommaVelocità Andata \ j
    Worksheets("Cartine").Range("N3") = sommaVelocità Ritorno \ k

End Sub
```

6.2 Uso di passi computazionali per lanciare ricette

Durante la costruzione del modello ci si è resi conto di alcune eccessive semplificazioni.

In primo luogo la comunicazione tra le ambulanze ed il box, e viceversa, non era considerata all'interno delle ricette, il che portava a nascondere un collo di bottiglia che nella realtà esiste ed è una seria preoccupazione dei dirigenti della centrale operativa.

Nello specifico il problema è rappresentato dal fatto che le ambulanze ed il box, durante la loro operatività, sarebbero tenuti a comunicare sei volte tra loro:

- nel momento in cui il box allerta le ambulanze;
- quando queste partono per raggiungere il luogo dell'incidente;
- quando arrivano sul luogo dell'incidente;
- quando ripartono verso l'ospedale;
- quando raggiungono l'ospedale;
- quando si danno nuovamente operative e quindi pronte a effettuare un altro intervento.

In realtà i dati forniti risultano fortemente incompleti poiché questa procedura raramente viene eseguita fedelmente.

Il problema della comunicazione potrebbe verificarsi anche nell'operazione di chiusura del servizio. Avviene infatti, in seguito all'arrivo all'ospedale dell'ambulanza e all'accettazione del paziente al pronto soccorso, una comunicazione tra gli operatori che hanno effettuato il servizio e la centrale operativa, riguardo ai dati della persona soccorsa. Questa fase avviene in contemporanea con le operazioni necessarie per rendere l'ambulanza nuovamente operativa e non interrompendole per effettuarla.

Essendo queste delle problematiche reali e gravose per il corretto servizio di pronto intervento sanitario si è deciso di sviluppare una procedura adeguata, ponendo particolare attenzione alla prima delle due perché è sicuramente quella che creerebbe maggiori disagi nel caso in cui dovessero esserci molte richieste d'aiuto e quindi molte ambulanze da gestire.

In prima analisi la soluzione più facilmente praticabile sembrava essere quella di dedicare delle unità di tempo (tic dell'orologio della simulazione)

all'intervento del box durante l'operatività delle ambulanze, al fine di rappresentare la comunicazione tra le due unità. E' immediatamente emerso che l'idea, seppur plausibile, peccava di realismo in quanto significava che ad ogni comunicazione l'ambulanza si fermasse per un unità temporale parlando con il box, ripartendo poi al momento successivo. Oltre all'ovvia considerazione che nella realtà questo non avviene e le operazioni si svolgono in contemporanea, una soluzione di questo tipo avrebbe posto un grave problema dovuto alla sequenzialità del programma: nel caso in cui la comunicazione non avvenisse, infatti, la simulazione si sarebbe bloccata in attesa. Questo ha reso necessaria un'altra soluzione.

Una seconda ipotesi è stata quella di sviluppare il formalismo AND, (attualmente non ancora attivo in jES) per le sue caratteristiche di contemporaneità. In figura 11[4] viene rappresentato un esempio del funzionamento di questo meccanismo: nella ricetta si possono avere percorsi paralleli che devono essere eseguiti simultaneamente in un dato momento. In particolare, nella ricetta sottostante la fase n1 viene eseguita per m1 tic e, in contemporanea, vengono eseguite le seguenti fasi così inserite nella ricetta:

&& n2 ts m2 n3 ts m3 && 2 n22 ts m22 && 0 n4 ts m4.

La spiegazione è che il branch 1 (indicato con && 1) esegue le fasi n2 e n3; in contemporanea il branch 2 (indicato con && 2) esegue la fase n22. Quando entrambi rami sono conclusi viene eseguita la fase n4. &&0 indica la conclusione del processo AND.

Dalla spiegazione si capisce il motivo per cui si è ricorsi ad un una procedura differente. Si ha infatti la necessità non solo che i processi siano paralleli, ma che la fase successiva proceda indipendentemente dalla conclusione di entrambi. Nel nostro caso questo implica che l'ambulanza, dopo aver

[4]Questa e la maggiorparte delle figure sono prese da: How To Use jEs, Terna(2001)

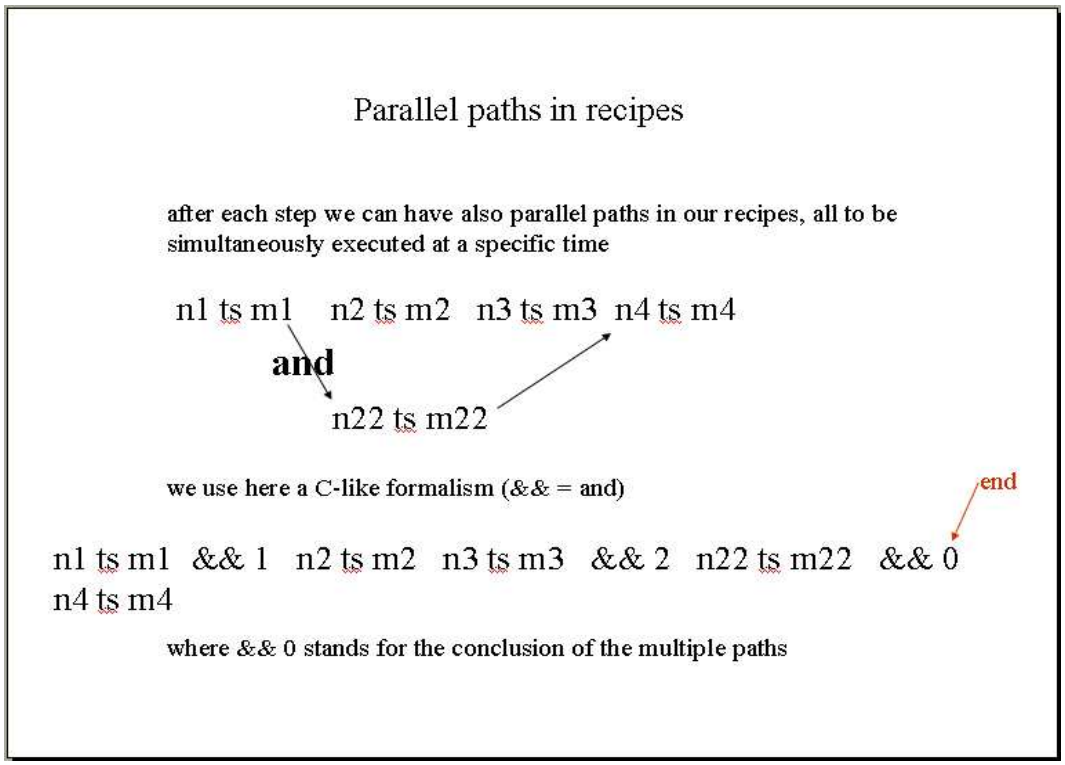


Figura 6.11: An AND process, with its branches (&& 1 and && 2)

comunicato con il box, prosegua il servizio indipendentemente dall'avvenuta ricezione o meno da parte del box. Lo stesso discorso si applica al caso inverso.

La conclusione a cui si è giunti consiste nell'eseguire, all'interno della ricetta principale, un passo computazionale, il quale attiva una sottoricetta che esegue la comunicazione; le due ricette proseguono poi indipendentemente. Ecco un esempio di come il formalismo viene inserito nelle ricette:

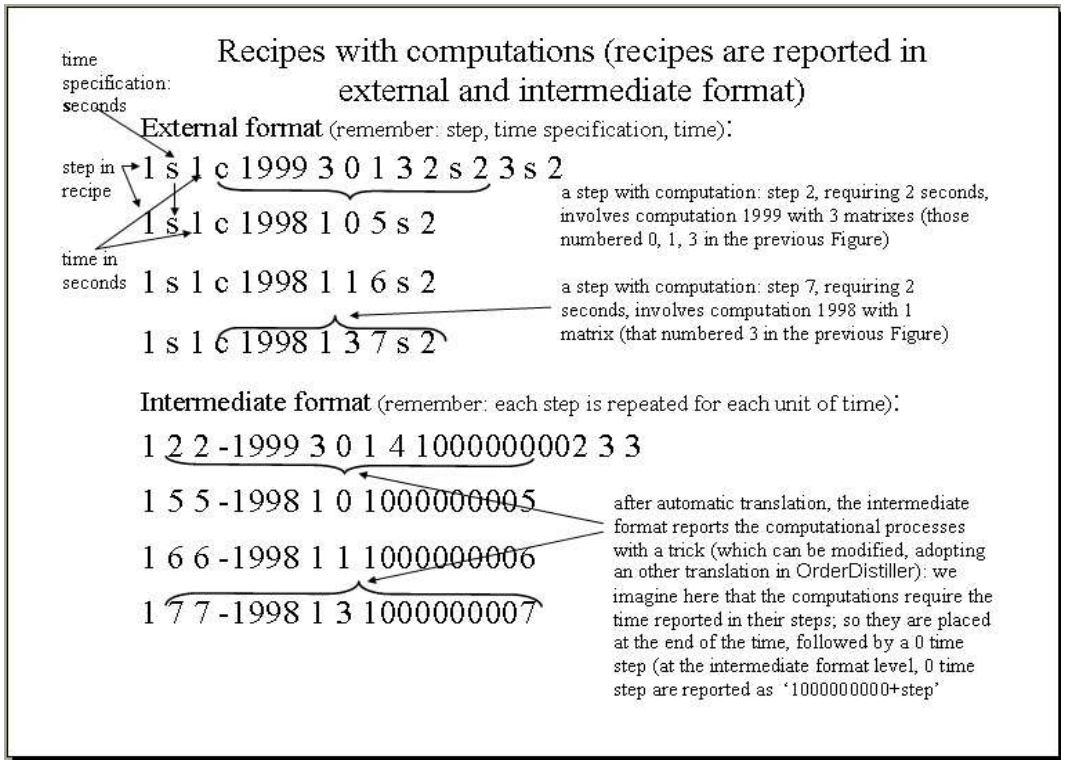


Figura 6.12: The format of the computational processes

La fase 1 viene eseguita per un tic, dopodiché la fase 2 richiama il passo computazionale numero 1999 (indicato con c 1999) il quale indica quante e quali matrici devono essere usate. In questo esempio il numero di matrici da utilizzare è 3 e le matrici da usare sono identificate con i numeri 0, 1, 3; queste contengono le procedure che devono essere eseguite. In figura 13 viene riportato il codice corrispondente il passo computazionale dell'esempio.

The Java Swarm code used by the recipes with the example code c 1999

```
/** computational operations with code -1999 (a code for the checking
   *  phase of the program
   *
   *  this computational code verifies position 0,0 of the three
   *  received matrixes; only if these positions are all not empty
   *  the code empties them and set the status to done
   */
   public void c1999(){
            mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
               getMemoryMatrixAddress(0);
            mm1=(MemoryMatrix) pendingComputationalSpecificationSet.
               getMemoryMatrixAddress(1);
            mm2=(MemoryMatrix) pendingComputationalSpecificationSet.
               getMemoryMatrixAddress(2);
            layer=pendingComputationalSpecificationSet.
               getOrderLayer();

            if(! (mm0.getEmpty(layer,0,0) || mm1.getEmpty(layer,0,0)
                              || mm2.getEmpty(layer,0,0) ) )
               {
                        mm0.setEmpty(layer,0,0);
                        mm1.setEmpty(layer,0,0);
                        mm2.setEmpty(layer,0,0);
                        done=true;
               }
} // end c1999
```

Figura 6.13: The Java code (simplified eliminating a control statement related to the consistence of the declared number of matrixes with the internal ones

Nel nostro caso il passo computazionale attiva la ricetta indicata all'interno delle matrici integrando nell'OrderDistiller i dati dell'OrderStartingSequence e dell'OrderSequence. Come risultato si avrà l'aggiunta in coda alla riga della Sequence, processata al momento, di un secondo evento, che fa riferimento alla ricetta di cui sopra. I due eventi sono simultanei, in quanto, l'OrderDistiller lancia le ricette e poi le lascia al loro destino. jES opera in modo distribuito con ogni ricetta che procede, come gli avvenimenti nella realtà, indipendentemente. jES, infatti, non ha controllo centralizzato e la ricetta madre è eseguita passando di unit in unit e così si esegue un passo computazionale che fa sì che OrderDistiller lanci una ricetta.

Ciò che accade quando la simulazione è lanciata è che la gestione degli ordini nelle unità incontra, durante la lettura delle ricette, il passo computazionale, generando un'informazione che sta sospesa in OrderDistiller sino al momento di eseguire il prossimo lancio di una ricetta. Ciò che si è dovuto creare è uno specifico passo che facesse un set in OrderDistiller con il codice tratto dalla matrice usata.

La costruzione di questo meccanismo ha posto dei problemi; era necessario capire dove l'OrderDistiller andasse a cercare il numero relativo alla ricetta da eseguire. La soluzione più appropriata è sembrata quella di custodire quest'informazione all'interno di un vettore posizionato in OrderDistiller stesso. Questo è composto esclusivamente da zeri all'inizio della simulazione; man mano che la simulazione procede e che i passi computazionali vengono eseguiti, il vettore si riempie con i numeri che corrispondono alle ricette. OrderDistiller legge all'interno del vettore e se trova i numeri delle ricette le esegue e svuota il vettore.

Nello sviluppo del formalismo si è deciso di procedere a piccoli passi in modo da testare, volta per volta, la correttezza dei cambiamenti. Durante la prima fase sono stati creati nuovi passi computazionali inseriti in Com-

putationalAssembler.java[5]. La prima versione del file svolgeva una funzione piuttosto elementare. Definiva una matrice, ne leggeva l'indirizzo di memoria ed inseriva il numero 287 nelle posizioni di riga e colonna (0,0). Alla fine il contenuto della matrice veniva visualizzato a video per permettere di capire se il passo computazionale fosse stato eseguito o meno. Tutto questo è stato creato aggiungendo le seguenti righe di codice nel file:

```
public void c1001()
{
    if(pendingComputationalSpecificationSet.
       getNumberOfMemoryMatrixesToBeUsed()!=1)
       {
         System.out.println("Code -1001 requires three matrix; " +
                     pendingComputationalSpecificationSet.
                     getNumberOfMemoryMatrixesToBeUsed() +
                     " found in order \# " +
                     pendingComputationalSpecificationSet.
                     getOrderNumber());
         MyExit.exit(1);
       }

       mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
             getMemoryMatrixAddress(0);

       mm0.setValue(layer,0,0,287.0);
       mm0.print();

       done=true;
} // end c1001
```

[5]Si deve copiare ComputationalAssembler.java nella cartella principale del programma da src/; questo perché il comando make run usa le classi contenute in lib/jesframe.jar, che sono quelle contenute in src/, ma le classi in /., cioè la cartella corrente, sovrascrivono quelle in jesframe.jar)

Il metodo esegue una parte di controllo comune a tutti i passi computazionali:

```
if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1)
   {
     System.out.println("Code -1001 requires three matrix; " +
                  pendingComputationalSpecificationSet.
                  getNumberOfMemoryMatrixesToBeUsed() +
                  " found in order \# " +
                  pendingComputationalSpecificationSet.
                  getOrderNumber());
     MyExit.exit(1);
}
```

nella quale cambia unicamente il riferimento al passo specifico, viene letto l'indirizzo di memoria della matrice utilizzata (la numero 0); alla matrice viene poi assegnato il valore 287 che viene scritto a video per verificare il corretto funzionamento del passo.

Inizialmente si era pensato di creare tanti passi computazionali simili a quello appena descritto, con l'unica differenza che invece di visualizzare a video il numero identificativo della ricetta, lo avrebbero scritto nel vettore contenuto in OrderDistiller; in un secondo momento si è invece deciso di modificare leggermente l'impostazione del formalismo. Si è preferito creare un passo computazionale[6] che si occupasse di leggere in un file di testo[7] i numeri delle ricette che simulano la comunicazione ed inserirli poi nelle matrici di memoria. L'altro passo[8] interagisce con l'OrderDistiller riempiendo volta per volta il vettore. Il codice dei due passi computazionali è il seguente:

```
public void c1001()
{
```

[6] nello specifico si tratta del passo 1001.

[7] recipesFromRecipes.txt.

[8] il passo 1002, corretto successivamente da Alessandro Balla e Roberto Crosetto.

```
try
{
  BufferedReader fileRecipesFromRecipes = null;
  StringTokenizer t;
  String line = " ";

  numberOfMatrixes=pendingComputationalSpecificationSet.
      getNumberOfMoryMatrixesToBeUsed();
  fileRecipesFromRecipes=newBufferedReader(new fileReader(
      "recipesFromRecipes.txt"));
  line=fileRecipesFromRecipes.readLine();
  line=fileRecipesFromRecipes.readLine();
  t = new StringTokenizer(line, " ");

  for (i = 0; i < numberOfMatrixes; i++)
  {
      t=MyReader.check("recipesFromRecipes.txt",
              fileRecipesFromRecipes, line, t);
      recipesToLaunch[i] = Integer.parseInt(t.nextToken());
  }

  fileRecipesFromRecipes.close();
}
catch (FileNotFoundException fnfe)
{
  System.out.println(fnfe);
  MyExit.exit(1);
}
catch (IOException e)
{
  System.out.println("IOException:" + e.toString());
  MyExit.exit(1);
}
```

```
  for (i = 0; i < numberOfMatrixes; i++)
  {
    mm =(MemoryMatrix) pendingComputationalSpecificationSet.
         getMemoryMatrixAddress(i);
    layer=pendingComputationalSpecificationSet.
    getOrderLayer();

    mm.setValue(layer,0,0,recipesToLaunch[i]);
  }
  done=true;
} // end c1001



public void c1002()
{
  if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1)
   {
       System.out.println("Code -1002 requires one matrix; " +
                          pendingComputationalSpecificationSet.
                          getNumberOfMemoryMatrixesToBeUsed() +
                          " found in order \# " +
                          pendingComputationalSpecificationSet.
                          getOrderNumber());
       MyExit.exit(1);
   }

 mm =(MemoryMatrix) pendingComputationalSpecificationSet.
    getMemoryMatrixAddress(0);

  aRecipe =(int) mm.getValue(layer,0,0);
  myOrderDistiller.setComputationalRecipe(aRecipe);
```

```
  done=true;
} // end c1002
```

6.2.1 Primi risultati

Il passo computazionale è stato creato inizialmente nel seguente modo: dopo un controllo sul numero delle matrici richieste si è letto l"indirizzo di memoria della matrice utilizzata dal passo (la numero zero); in questa si è inserito nelle posizioni di riga e colonna (0,0) il numero 287, che corrisponde alla ricetta che deve essere lanciata in contemporanea a quella che la richiama. Si è poi fatto scrivere a video il valore presente nella matrice come verifica[9].

All'inizio della simulazione compare immediatamente a video una scritta in cui viene indicato, nell'ordine, il numero di matrice utilizzata, la fase che richiama il passo computazionale e per quante unità temporali lavora la fase in questione. Utilizzando quest'unica ricetta:

ex-post1 142408
100 s 1 200 s 23 301 s 1 c1001 1 0 450 s 12 450 s 42 450 s 114
700 s 0 800 s 6 450 s 180;

dove era inserito il nostro passo computazionale la simulazione presentava questa scritta:

0 450 s 12

che corrisponde al numero della matrice utilizzata dal passo computazionale (0), la fase che richiama il passo (450) ed i tic della fase (12). Questa scritta è data dal gestore degli ordini, che all'inizio del nostro mondo controlla che tutto sia in ordine e rileva anche la presenza di passi computazionali. In un

[9]Per visualizzarlo a video si è dovuto settare a true il campo printMatrixes in jesframe.scm

secondo momento è stata sviluppata una versione intermedia, in cui c'era un passo che caricava le matrici (c1001) ed uno che riempiva il vettore con i numeri letti dalle matrici di memoria precedentemente riempite (c1002). In questa versione però i numeri identificativi delle ricette da lanciare dovevano essere scritti all'interno del codice e questo rendeva ovviamente il formalismo poco flessibile e quindi migliorabile. La soluzione è stata quella di fare leggere questi numeri da un file di testo[10] in modo da consentire una più facile modifica degli stessi da parte dell'utente. Il file deve avere questa struttura[11]:

M0 M1 M2

287 288 289

Ad ogni matrice di memoria utilizzata corrisponde il numero identificativo della ricetta che verrà assegnato a quella matrice. Nell'esempio precedente vengono utilizzate la matrice numero 0 (M0) in cui verrà inserito il numero 287 (ricetta da lanciare) e le matrici numero uno e due (M1, M2) che verranno riempite con 288 e 289. Ovviamente, ogni volta che il numero delle matrici dovesse variare, oltre al file recipesFromRecipes.txt dovranno essere modificati anche jesframe.scm e memoryMatrixes.txt

Il codice del passo 1001 che esegue la fase di lettura è il seguente:

```
try
  {
    BufferedReader fileRecipesFromRecipes = null;
    StringTokenizer t;
    String line = " ";
```

[10] recipesFromRecipe.txt

[11] gli spazi tra i numeri delle ricette devono essere dati esclusivamente con la barra spaziatrice o con il tasto di ritorno a capo. Altri tasti (ad esempio il Tab) non vengono accettati e mandano in errore il programma.

```
    numberOfMatrixes = pendingComputationalSpecificationSet.
                getNumberOfMemoryMatrixesToBeUsed();
    fileRecipesFromRecipes = new BufferedReader(
        new FileReader("recipesFromRecipes.txt"));
    line=fileRecipesFromRecipes.readLine();
    line=fileRecipesFromRecipes.readLine();
    t = new StringTokenizer(line, " ");

    for (i = 0; i < numberOfMatrixes; i++)
    {
        t=MyReader.check("recipesFromRecipes.txt",
                fileRecipesFromRecipes, line, t);
        recipesToLaunch[i] = Integer.parseInt(t.nextToken());
    }

    fileRecipesFromRecipes.close();
  }
  catch (FileNotFoundException fnfe)
  {
    System.out.println(fnfe);
    MyExit.exit(1);
  }
  catch (IOException e)
  {
    System.out.println("IOException:" + e.toString());
    MyExit.exit(1);
  }
```

La parte di codice che si occupa del caricamento delle matrici di memoria è variato rispetto alle versioni iniziali, perché il numero di matrici non è più definito nel codice, ma possono variare a seconda di quante ne sono contenute nel file recipesFromRecipes.txt.

In particolare la versione definitiva è la seguente:

```
for (i = 0; i < numberOfMatrixes; i++)
```

```
{
  mm=(MemoryMatrix) pendingComputationalSpecificationSet.
       getMemoryMatrixAddress(i);
   layer=pendingComputationalSpecificationSet.
   getOrderLayer();
  mm.setValue(layer,0,0,recipesToLaunch[i]);
}
```

6.2.2 Interazione tra ComputationalAssembler e OrderDistiller

Il vettore all'interno di OrderDistiller si chiama computationalRecipes; esso viene creato dal costruttore di OrderDistiller in una misura sovrabbondante (1000 elementi), in modo da contemplare la possibilità che ci siano moltissime ricette da lanciare per ogni unità di tempo; ovviamente, nel caso in cui non fosse comunque sufficiente a contenere i numeri identificativi di tutte le ricette la simulazione verrà interrotta visualizzando a video il tipo di errore verificatosi. In particolare, la dichiarazione del vettore avviene nella seguente forma:

int[ ] computationalRecipes;

cioè il tipo di variabile che si vuole usare (in questo caso il vettore è composto da numeri interi), le due parentesi quadre che significano che la variabile è un vettore ed il nome della variabile stessa. Il vettore deve poi essere effettivamente creato e di questo si occupa il costruttore della classe OrderDistiller nella seguente forma:

computationalRecipes new int [1000]

cioè si crea la nuova (new) variabile che è un vettore composto da 1000 elementi. Il vettore è subito inizializzato a zero, il che significa che ad ogni

elemento del vettore viene assegnato il valore zero, come si può vedere dal codice sottostante:

```
// CONSTRUCTOR
public OrderDistiller (Zone aZone, int msn, int msl, ListImpl ul,
ListImpl eul, ListImpl ol, int tln, ESFrameModelSwarm mo,
        AssigningTool at)
        { //pt
                super(aZone, msn, msl, ul, eul, ol, tln, mo, at);

                computationalRecipes = new int[1000];
                for(int i=0; i<1000; i++)
                {
                    computationalRecipes[i]=0;
                }

                unitList=ul;
                endUnitList=eul;
                orderList=ol;
                assigningTool=at; //pt
        }
```

Questa operazione è compiuta perché successivamente il vettore sarà oggetto di controllo ad ogni tic e saranno lanciate tutte le ricette inserite nel vettore; questo finché non si trova uno zero, dopodiché il vettore viene riazzerato. In seguito si è passati alla modifica di ComputationalAssembler in modo da poterlo fare interagire con OrderDistiller.

Le attuali modifiche[12] sono state effettuate per dotare ComputationalAssembler dell'indirizzo di memoria di OrderDistiller, in modo da permettere l'interazione tra le due classi. Il codice corrispondente è:

```
public class ComputationalAssemblerBasic extends SwarmObjectImpl
```

[12]In seguito sono state effettuate delle correzioni da parte di Alessandro Balla e Roberto Crosetto.

```
{

  /** the memory matrixes used in computations; we can add a lot
  of these */
  MemoryMatrix mm0, mm1, mm2, mm3, mm4, mm5, mm6, mm7, mm8, mm9;
  /** the current layer */
  public int layer;

  /** the unit for which we are assembling for */
  public Unit myUnit;
  /** the list of the order requiring computations */
  public ListImpl waitingList;
  /** a computational specification set */
  ComputationalSpecificationSet.
        pendingComputationalSpecificationSet;
  /** the status of a specific computation */
  public boolean done;

  OrderDistiller myOrderDistiller;
  /**
   * the constructor for ComputationalAssembler
  */

  public ComputationalAssemblerBasic(Zone aZone, OrderDistiller od)
  {
     // Call the constructor for the parent class.
        super(aZone, od);

        myOrderDistiller= od;

        // the list of orders with computational processes to
        // be assembled
        waitingList = new ListImpl (getZone());
    }
```

Dopo aver modificato i parametri da passare alla classe OrderDistiller si è dovuto modificare anche la classe ESFrameModelSwarm; ogni volta che si incontravano le seguenti righe di codice:

```
aComputationalAssembler = new ComputationalAssembler (getZone());
```

si è dovuto aggiungere anche myOrderDistiller, in questo modo:

```
aComputationalAssembler = new ComputationalAssembler(getZone(),
myOrderDistiller);
```

E' stato inoltre importante conoscere la struttura di OrderDistiller e capirne il significato, in quanto è questa classe che, attraverso il metodo Distill, si occupa del lancio delle ricette. Di seguito vengono illustrati il significato del codice contenuto nella classe OrderDistiller. Quando devono essere lanciate le ricette viene innanzitutto eseguito il metodo setDictionary, che si occupa di creare la lista delle ricette (recipeList), creare l'oggetto aRecipe che contiene le ricette prese dal foglio Excel Recipes ed inserire tali ricette nella lista.

```
recipeWorksheet = new ExcelReader(recipeFile);

while(! recipeWorksheet.eof())
  {
        aRecipe = new Recipe(getZone());
        aRecipe.setRecipeFrom(recipeWorksheet);
        recipeList.addLast(aRecipe);
  }
```

Nel metodo Distill vengono poi lanciate le ricette precedentemente inserite nella lista. In particolare il metodo Distill compie i seguenti passi:

- controlla se trova un ";" nel qual caso passa alla riga della Sequence successiva;

```
if(orderSequenceWorksheet.checkForLabelCell())
  {
```

```
        String aString = orderSequenceWorksheet.getStrValue();
        if(aString.equals(semicolon))
        {
                isEndOfShift = true;
                break;
        }
        else
        {
                orderSequenceWorksheet.goBack();
        }
    }
```

- legge l'ordine della ricetta dal foglio Excel OrderStartingSequence (o OrderSequence);

```
//Reading the order from the worksheet

readOrderFrom(orderSequenceWorksheet);//see below
```

e attraverso la funzione privata readOrderFrom() inserisce in currentOrder[0] l'ordine da lanciare, ed in currentOrder[1] le volte che deve essere ripetuto;

```
private int[] readOrderFrom(ExcelReader e)
  {
    orderSequenceWorksheet = e;
    currentOrder = new int[2];

    checkForComment();
    checkForLayer();
    currentOrder[0] = orderSequenceWorksheet.getIfInteger();
    orderSequenceWorksheet.getIfString();
    currentOrder[1] = orderSequenceWorksheet.getIfInteger();
    return currentOrder;
  }
```

- controlla se una ricetta con quel codice effettivamente esiste scorrendo tutta la lista caricata precedentemente;

```
for (int r = 0; r < recipeList.getCount() &&
        !recipeCodeFound; r++)
        {
          aRecipe = (Recipe) recipeList.atOffset(r);
             recipeCodeFound = true;
        }
```

- dopodiché avviene l'effettivo lancio della ricetta per il numero di volte contenute in currentOrder[1];

```
for(int q = 0; q < currentOrder[1]; q++)
```

il nuovo ordine viene inserito nell'oggetto anOrder,

```
anOrder = new Order(getZone(), orderCount,
Globals.env.getCurrentTime(),aRecipe.getLength(), aRecipe.
        getRecipeSteps(),eSFrameModelSwarm, endUnitList);
```

gli viene dato il nome dell'ordine corrente contenuto nella lista,

```
anOrder.setRecipeName(aRecipe.getRecipeName());
```

e viene inserito nella lista generale degli ordini,

```
orderList.addLast(anOrder);
```

infine l'ordine viene spedito alla prima unità produttiva.

```
assigningTool.assign(anOrder);
```

A questo punto l'attenzione si è spostata sulla creazione del metodo setComputationalRecipe, il quale appartiene alla classe OrderDistiller. Tale metodo esplora il vettore sino a trovare la prima posizione messa a 0, nella quale si inserisce il numero identificativo della ricetta da lanciare.

```
public void setComputationalRecipe(int aRecipeToBeAdded)
        {
            int k = 0;
            while (computationalRecipes[k] != 0)
            {
                k++;
            }

            computationalRecipes[k] = aRecipeToBeAdded;
        }
```

L'ultimo problema è stato l'aggiunta del codice che ha permesso effettivamente alle ricette caricate nel vettore computationalRecipes di essere lanciate. Tale codice è stato inserito nella classe OrderDistiller ed in particolare nel metodo Distill.

Il codice sottostante ha il seguente significato: quando avviene il controllo sul ";" il programma effettua un controllo anche sul vettore computationalRecipes e, nel caso in cui non sia vuoto, le ricette corrispondenti ai numeri contenuti dal vettore sono lanciate; successivamente il vettore è riazzerato.

```
if(aString.equals(semicolon))
{
                boolean recipeCodeFound = false;
for (int r = 0; r < recipeList.getCount() &&
        !recipeCodeFound; r++)
        {
          int i=0;
          while (computationalRecipes[i]!= 0)
          {
            aRecipe = (Recipe) recipeList.atOffset(r);
            if(aRecipe.getRecipeCode() ==
                computationalRecipes[i])
            {
                System.out.println("ricetta " +
```

```
                computationalRecipes[i] +
                " trovata!");
            computationalRecipes[i]=0;
            recipeCodeFound = true;
        }
        i++;
    }
}

if(recipeCodeFound == true)
{
    orderCount++;
    anOrder = new Order(getZone(), orderCount,
            Globals.env.getCurrentTime(),
            aRecipe.getLength(), aRecipe.getRecipeSteps(),
            eSFrameModelSwarm, endUnitList);

    anOrder.setRecipeName(aRecipe.getRecipeName());
    // setting the layer
    // (from 0 to totalLayerNumber-1)
    if(currentLayer <0 || currentLayer >= totalLayerNumber)
    {
            System.out.println("A layer in the sequence is
                not inclued\n"+"in the interval from 0 to
                totalLayerNumber-1");
            MyExit.exit(1);
    }
    anOrder.setOrderLayer(currentLayer);

    // add the active orders to the general order list
    // (they will be eliminated when dropped in a unit,
    // being finished);
    // this list has been introduced for accounting purposes
    // [may be it would be better substitute it with a get
```

```
            // to the units to know their waiting lists]
                        orderList.addLast(anOrder);

            // sending the order to the first production unit
            // (we are acting as the Front End of the ES)
            assigningTool.assign(anOrder);
            }

            isEndOfShift = true;
            break;
          }
          else
          {
            orderSequenceWorksheet.goBack();
          }
}
```

6.3 Un primo modello ex-ante

Con la messa a punto del modello ex-post e la pulizia dei dati lo scopo del lavoro è cambiato. La simulazione riproduceva ormai fedelmente la realtà ed era interessante, a questo punto, utilizzare il modello per ipotizzare dei cambiamenti all'interno della realtà organizzativa del 118, provarli sul computer e vagliare il risultato ottenuto.

I possibili cambiamenti all'interno di un'organizzazione sono tantissimi, provarli a caso può risultare poco costruttivo. Il procedimento più logico è quello di trovare i punti più sensibili all'interno dell'organizzazione (i punti in corrispondenza dei quali la produzione del servizio rallenta eccessivamente a fronte di un aumento importante delle richieste di assistenza) e operarvi dei cambiamenti.

Per ottenere questo risultato è necessario sovraccaricare il modello (far

sì che debba processare carichi di lavoro molto più grandi di quelli che normalmente si sono presentati in una giornata normale) e osservare in corrispondenza di quali unità produttive si formano le code di attesa più grandi. Per poterlo fare è però necessario passare prima dal modello ex-post ad un modello ex-ante[13].

Nel modello ex-post ogni unità produttiva si limitava a svolgere un compito specifico che le era effettivamente appartenuto durante la giornata simulata. Per esempio, se l'ambulanza 835 aveva effettuato una missione di soccorso alle ore 14:35, anche nella simulazione avrebbe dovuto ripetere lo stesso intervento. Se fossero state libere al momento due ambulanze abilitate ad effettuare lo stesso tipo di interventi, il lavoro sarebbe dovuto comunque essere svolto dall'ambulanza 835.

Nel modello ex-ante, invece, le unità produttive non devono limitarsi a svolgere un compito solo (come saper intervenire una volta alle ore 14:35 e un'altra alle ore 17:58), ma devono essere in grado di svolgere diversi compiti.

Per introdurre queste unità *complesse* bisogna effettuare una serie di modifiche all'interno del jesframe, modificando prima di tutto il file unitBasicData.txt.

Nella colonna intitolata production phase, invece di inserire il numero relativo al compito specifico che l'unità è in grado di eseguire, è necessario inserire il numero '0'.

L'esistenza di numeri zero all'interno di unitBasicData.txt obbliga il programma ad andare alla ricerca del file units.xls, custodito anch'esso all'interno della cartella unitData.

La creazione del file units.xls è la seconda modifica necessaria per creare un modello ex-ante. Questo file contiene un foglio di calcolo per ogni unità complessa presente nel modello; ogni foglio di calcolo ha il nome dell'unità

[13]Il lavoro sul modello ex-ante e la descrizione che qui ne viene fatta sono opera dell'ormai dottore Carlo Badino, con la mia collaborazione.

Simple production unit data are reported in a text file
(unitData/unitBasicData.txt)

unit_#	useWarehouse	prod.phase_#	fixed_costs	variable_costs
1	1	11	12	1
2	1	0	0	0
3	1	3	15	2
4	1	0	0	0
5	1	51	12	2
6	1	6	11	20
7	1	12	23	1
8	1	8	22	11
9	1	13	7	12
10	1	18	40	7
11	1	11	5	1

Figura 6.14: Il file unitBasicData.txt

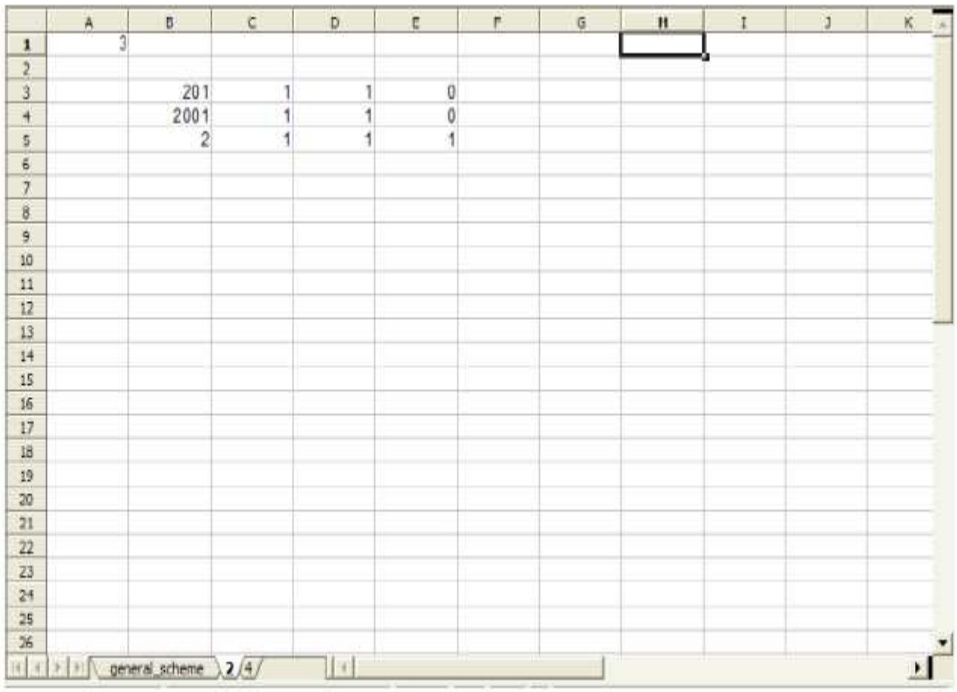


Figura 6.15: Il file units.xls

complessa che rappresenta. Il primo dato che figura è quello relativo a quante operazioni l'unità è capace di svolgere.

In figura 2 l'unità sa svolgere tre compiti: l'operazione 201, l'operazione 2001 e l'operazione 2. Gli altri numeri presenti nel foglio di calcolo si riferiscono ai compiti che l'unità è in grado di portare a termine (201, 2001 e 2), ai costi fissi che le unità devono affrontare per la produzione (1,1,1) ai costi variabili per la produzione (1,1,1) e alla possibilità che l'unità stia producendo un inventario o meno (1 in caso positivo, 0 nel caso contrario). Queste informazioni sono riassunte nella figura relativa allo schema generale di un'unità complessa.

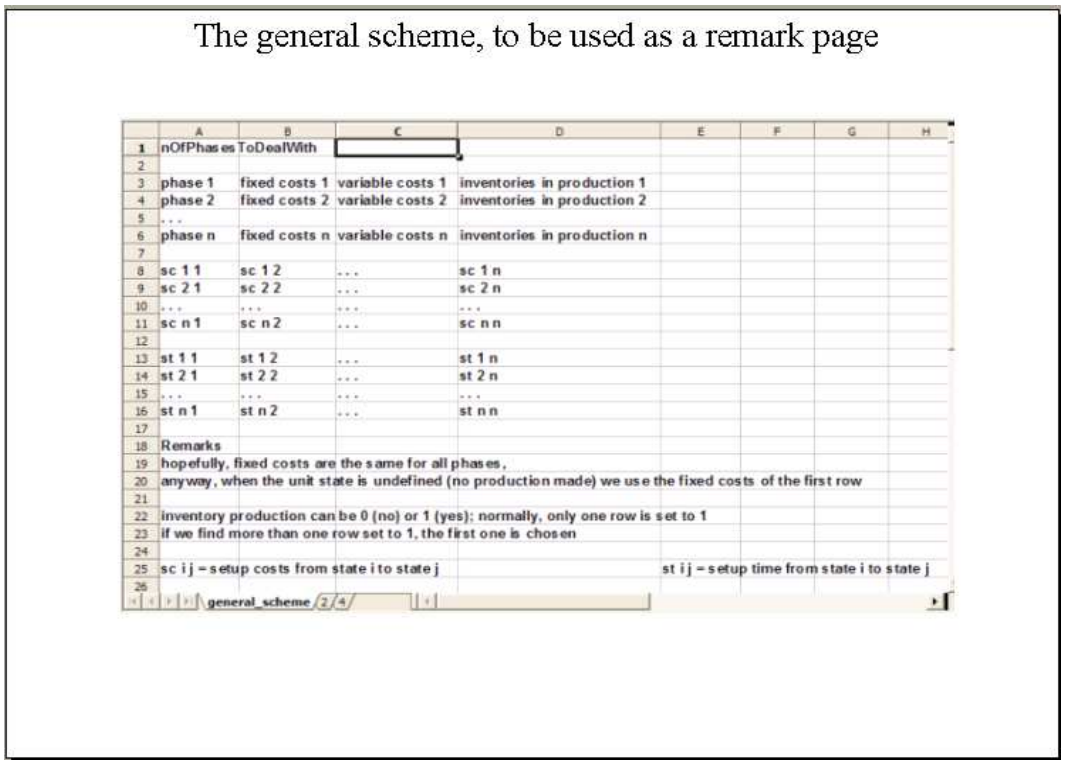


Figura 6.16: Lo schema generale di un'unità complessa

Per passare al modello ex-ante, nel file units.xls relativo alla simulazione del 118 sono state introdotte una serie di unità complesse rappresentanti le ambulanze che intervengono per prestare soccorso.

Ad ogni ambulanza è stata data la possibilità di svolgere due tipi di compiti diversi: i compiti che aveva effettivamente svolto durante la giornata simulata e quelli attinenti alla sua tipologia di ambulanza (per esempio le ambulanze meglio equipaggiate, le MSA, potevano intervenire per ripetere gli interventi da loro effettuati durante la giornata oggetto di simulazione oppure potevano svolgere le mansioni che erano state proprie di un'altra ambulanza MSA).

Introdotte queste modifiche nel modello si è proceduto come segue:

- per prima cosa abbiamo fatto funzionare il modello chiedendo ad ogni ambulanza di ripetere gli interventi effettivamente svolti durante la giornata oggetto di simulazione (ad ulteriore conferma della solidità e della bontà del modello). Coerentemente con le nostre previsioni non

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
1	#	Recipes	;																																	
2		ex-post0	10000		c	1001	4	0	1	2	3	100	s	1	;																					
3		ex-post01	1		302	s	2		2	s	1	;																								
4		ex-post02	2		303	s	2		2	s	1	;																								
5		ex-post03	3		2	s	2		302	s	1	;																								
6		ex-post04	4		2	s	2		303	s	1	;																								
7		ex-post1	409		100	s	1		200	s	6	;																								
8		ex-post2	481		100	s	1		200	s	6	;																								
9		ex-post3	488		100	s	1		200	s	6	;																								
10		ex-post4	507		100	s	1		200	s	6	;																								
11		ex-post5	524		100	s	1		200	s	6	;																								
12		ex-post6	543		100	s	1		200	s	6	;																								
13		ex-post7	552		100	s	1		200	s	6	;																								
14		ex-post8	563		100	s	1		200	s	6	;																								
15		ex-post9	608		100	s	1		200	s	6	;																								
16		ex-post10	617		100	s	1		200	s	6	;																								
17		ex-post11	621		100	s	1		200	s	6	;																								
18		ex-post12	623		100	s	1		200	s	6	;																								
19		ex-post13	647		100	s	1		200	s	6	;																								
20		ex-post14	692		100	s	1		200	s	6	;																								
21		ex-post15	717		100	s	1		200	s	6	;																								
22		ex-post16	726		100	s	1		200	s	6	;																								
23		ex-post17	751		100	s	1		200	s	6	;																								
24		ex-post18	768		100	s	1		200	s	6	;																								
25		ex-post19	824		100	s	1		200	s	6	;																								
26		ex-post20	838		100	s	1		200	s	6	;																								
27		ex-post21	858		100	s	1		200	s	6	;																								
28		ex-post22	874		100	s	1		200	s	6	;																								
29		ex-post23	880		100	s	1		200	s	6	;																								
30		ex-post24	881		100	s	1		200	s	6	;																								
31		ex-post25	583		100	s	1		200	s	23		302	s	1		c	1002	1	0	3115	s	12		c	1002	1	2	3115	s	60		c	1002	1	2
32		ex-post26	503		100	s	1		200	s	35		302	s	1		c	1002	1	0	1260	s	30		c	1002	1	2	1260	s	168		c	1002	1	2

Figura 6.17: Il file recipes.xls

si sono verificate code;

- abbiamo modificato il file recipes.xls affinché le ricette non chiamassero più in causa le singole ambulanze, ma segnalassero la necessità di portare a termine una tipologia di intervento. In questo caso i numeri identificativi delle ambulanze sono stati sostituiti con le cifre "1", "2" e "3" a seconda che fosse richiesto un intervento di un MSB, di un MSAB o di un MSA;

- abbiamo fatto nuovamente girare il modello senza che si verificassero delle code.

A questo punto è stato finalmente possibile sovraccaricare il modello, mettendolo "sotto stress".

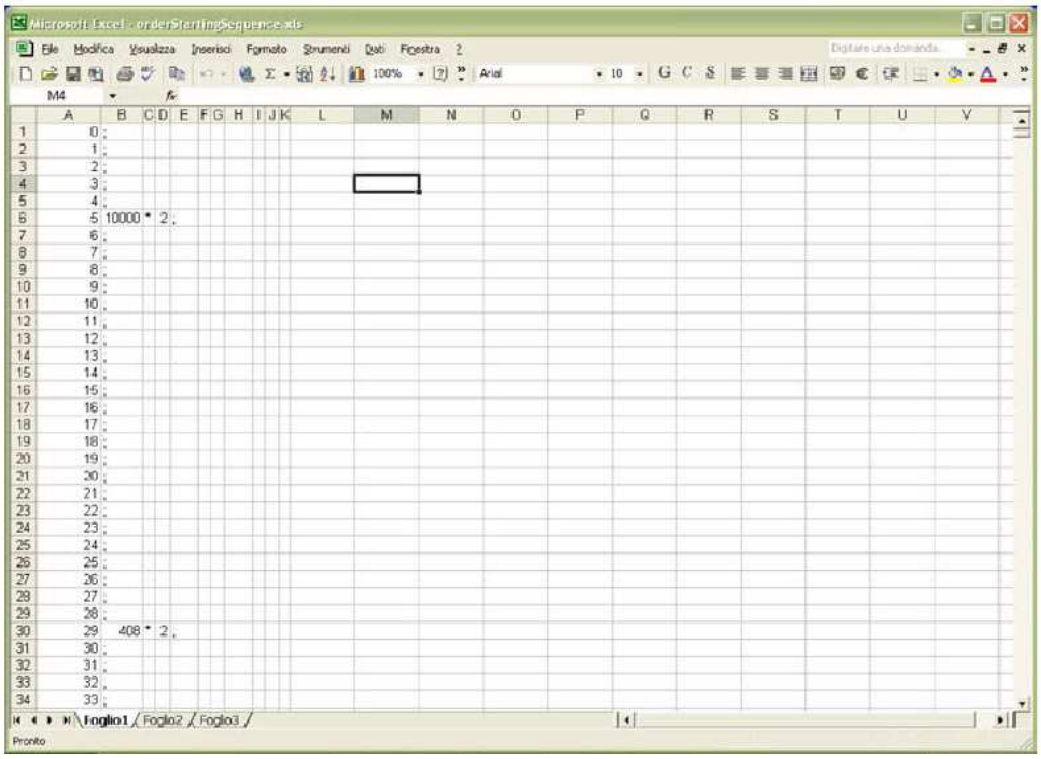


Figura 6.18: Il file orderStartingSequence.xls

Questo risultato è stato ottenuto agendo sul file orderStartingSequence.xls.

Tale file contiene le informazioni riguardanti quali interventi avvengono in una giornata e a che ora essi si verificano.

Per sovraccaricare il modello è stato sufficiente raddoppiare gli interventi previsti per la simulazione iniziale. I risultati hanno dimostrato che le prime unità produttive ad andare in crisi sono le PVS e la postazione di chiusura.

Abbiamo allora pensato di introdurre una unità PVS in più, per verificare in che misura un aumento della capacità produttiva dell'organizzazione, concentrato nel punto più esposto ad un rallentamento nell'erogazione del servizio, potesse far fronte alle code verificatesi.

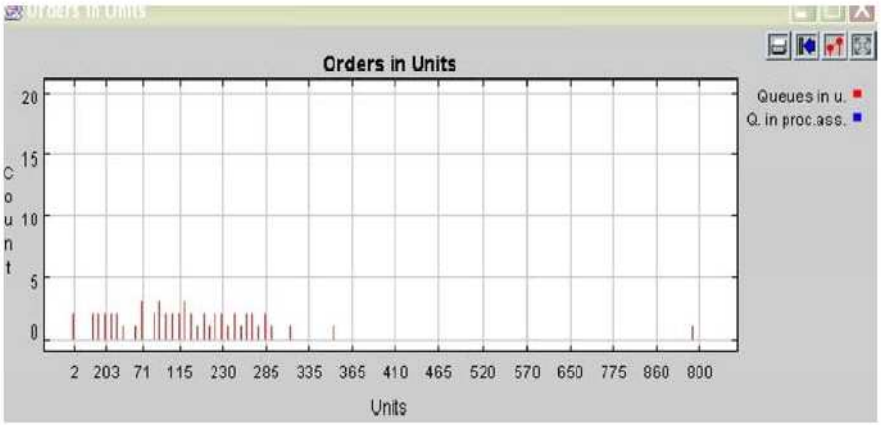


Figura 6.19: Il modello in condizioni normali

Teoricamente, a fronte di un aumento degli interventi pari al totale degli interventi effettuati in una giornata, basterebbe raddoppiare le PVS per essere sicuri che il servizio sia erogato in maniera ottimale, passando così dalle sei postazioni esistenti a dodici.

La simulazione ha invece mostrato che basta introdurre tre nuove PVS perché la situazione si normalizzi e le code scompaiano.

E' perciò possibile far fronte ad una situazione di eccezionale emergenza senza essere costretti ad assumere un numero elevato di nuovi operatori. Questo fatto è interessante in quanto mette in luce uno dei possibili vantaggi

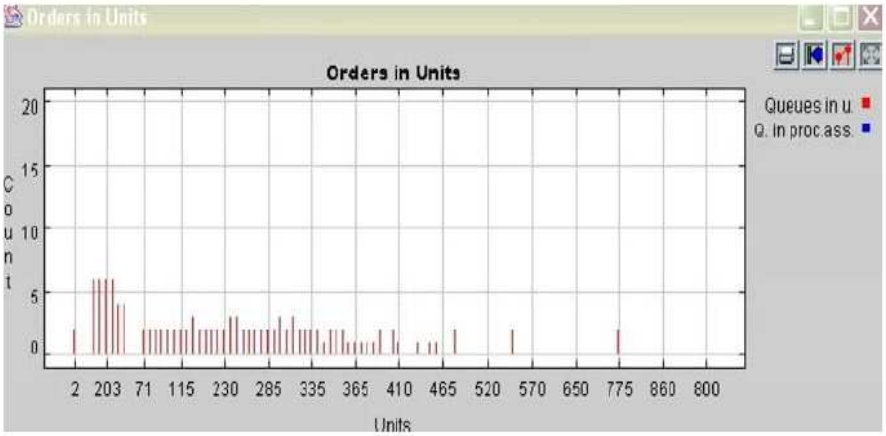


Figura 6.20: Il modello sovraccaricato

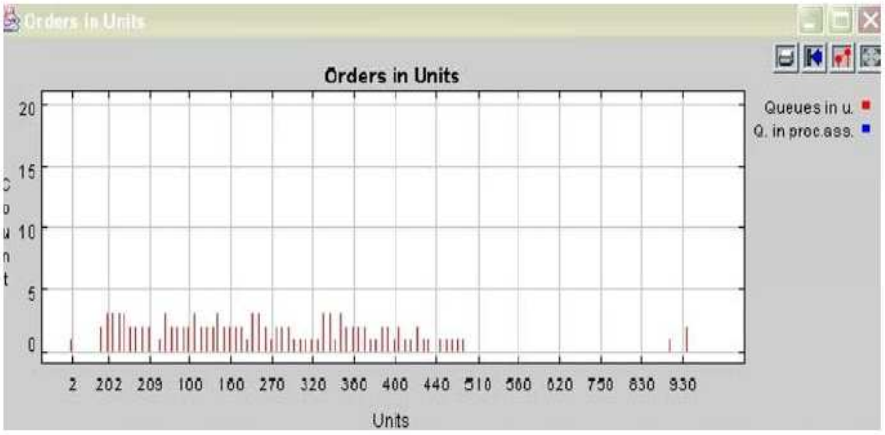


Figura 6.21: Le code si normalizzano con le nuove PVS

del metodo simulativo; il servizio di pronto soccorso si occupa di assistere persone che sono rimaste vittime di incidenti o che hanno accusato dei malori; l'intervento di infermieri, medici e volontari riesce in molti casi a scongiurare che le conseguenze siano fatali. Un servizio così importante, dove un ritardo o un disguido possono avere conseguenze pesantissime, non può permettersi di effettuare troppi esperimenti per migliorare il servizio. Ogni cambiamento porta con sé incertezze e contrattempi.

La simulazione può fornire un aiuto importante nel processo di miglioramento del servizio fornito ai cittadini. I cambiamenti non saranno più un'incognita e se una modifica risulterà inutile o dannosa nella simulazione non verrà più presa in considerazione nella realtà. Se, invece, un cambiamento si sarà dimostrato responsabile di un miglioramento nel funzionamento dell'organizzazione, si potrà valutare, con cautela, l'ipotesi di introdurlo nella realtà.

Il caso concreto è importante per il seguente motivo: per essere sicuri di offrire la stessa qualità del servizio in caso di forte emergenza, la centrale operativa, se non fosse a conoscenza dei risultati della simulazione, sarebbe costretta ad assumere sei nuovi operatori.

La simulazione dimostra che per ottenere lo stesso risultato bastano tre nuovi operatori, con un evidente risparmio sia in termini economici che di coordinamento tra i nuovi operatori ed il personale già presente.

E' utile aggiungere che in un'organizzazione come il 118, dove la qualità del servizio è estremamente importante, è utile che i cambiamenti siano i più mirati e della minore entità possibili, per non stravolgere il normale funzionamento della centrale operativa.

6.4 Sviluppi futuri

Dopo aver costruito il modello ex-post ed aver effettuato una prima prova di modello ex-ante il lavoro non è certamente terminato. Si apre invece la fase forse più interessante dal punto di vista della ricerca.

Alcuni aspetti del modello devono ancora essere sviluppati. In futuro diventerà importante introdurre delle modifiche che permettano di rendere la simulazione ancora più realistica e capace di effettuare previsioni che aiutino l'organizzazione del 118 a migliorare il servizio offerto. Le prospettive di sviluppo del modello sono, anche in relazione ai desideri espressi dai responsabili della centrale operativa di Grugliasco, le seguenti:

- la possibilità di "spegnere" le unità ambulanze. Non tutti i mezzi, infatti, sono disponibili a tutte le ore. Questo rende necessario che alcuni mezzi (quelli estemporanei) non rispondano in determinate fasce orarie; le unità corrispondenti vanno quindi "spente";

- la localizzazione delle ambulanze. Lavorando su un territorio molto vasto come quello della provincia torinese, non si può pensare che tutti i mezzi possano intervenire su tutti i casi; nella realtà questa divisione è abituale ma nella simulazione è necessario introdurre un limite agli interventi entro un raggio d'azione delimitato;

- la visualizzazione grafica del modello. E' in atto una modifica sostanziale di jES, che permetta una visualizzazione grafica del modello. Si vuole una struttura simile a quella creata con jESEvol che, discostandosi dal rigido istogramma, permetta di vedere tutte le singole unità in un piano e alcuni indicatori che indicano come le unità lavorano e perché si verificano determinati fenomeni. Il risultato che si vuol raggiungere è di questo tipo:

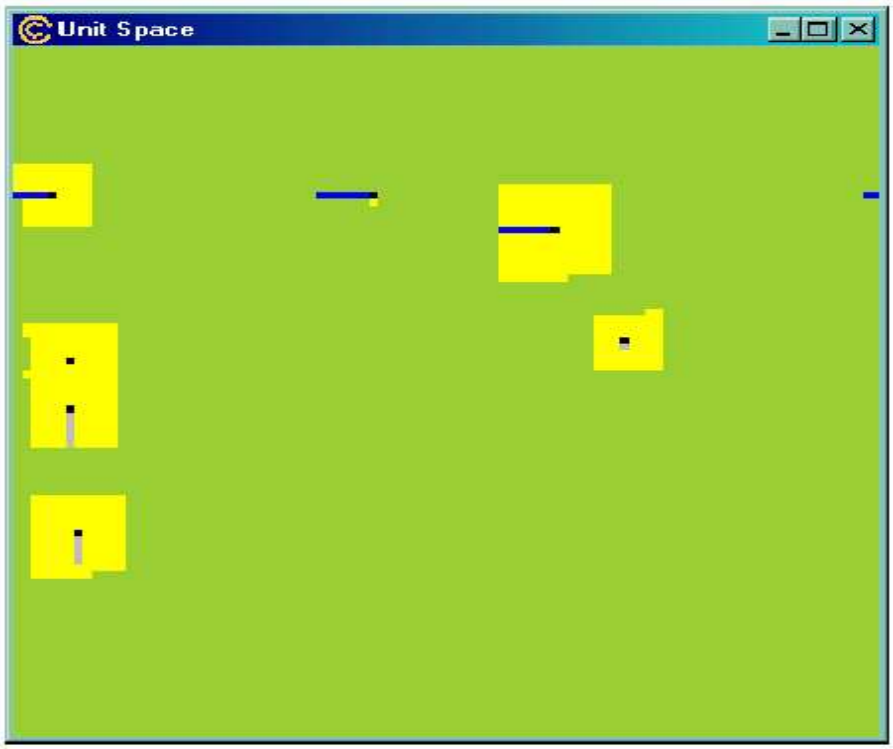


Figura 6.22: jESEvol

- diverse squadre con mezzi in grado di muoversi più agilmente nel traffico. Secondo l'idea del dottor Ghiselli, direttore della centrale operativa di Grugliasco, potrebbe essere utile introdurre delle autovetture di soccorso a fianco delle ambulanze. Le autovetture servirebbero per portare sul luogo di un incidente medici e personale specializzato; le ambulanze di base, raggiunte sul luogo dell'incidente dalle autovetture di soccorso, permetterebbero di ricomporre in loco equipaggi di cui finora dispongono solo le ambulanze avanzate.

- possibilità di ridefinire la missione di un'ambulanza già destinata. In caso di emergenza, un'ambulanza già impegnata in una missione di soccorso può essere reindirizzata verso un altro intervento. Questo caso si verifica soprattutto se il primo intervento è effettuato per prestare soccorso in caso di malori di lieve entità;

- scelta dell'ospedale di destinazione. Spesso, infatti, l'ospedale di destinazione è una scelta obbligata. In caso di serio trauma il paziente sarà trasportato al C.T.O., in caso di infarto sarà necessario recarsi presso un ospedale dotato di unità coronarica;

- controlli per evitare di mandare più mezzi nello stesso luogo per evitare di sprecare inutilmente risorse.

Capitolo 7

Conclusioni

La simulazione al computer è indubbiamente una valido aiuto nello studio delle organizzazioni. L'elaborazione di modelli complessi permette di studiare le scienze sociali rendendo meno arduo il compito di analizzare fenomeni caratterizzati dall'interazione tra le persone; tali fenomeni non si possono spiegare utilizzando solo i metodi letterari e matematici.

In quest'ottica è stato sviluppato il modello del 118, secondo le richieste ricevute dalla centrale operativa di Grugliasco. Le motivazioni emerse durante i primi incontri, quando il modello del 118 era solo un'idea, indicarono una buona comprensione dell'utilizzo della simulazione: ci dissero infatti che erano contenti del servizio offerto, ma che avrebbero voluto fare dei cambiamenti. Data la natura del servizio, però, non potevano permetterseli, poichè rischiavano di creare dei problemi legati ai ritardi non accettabili. La simulazione era quindi un valido aiuto alla capacità di valutare possibili cambiamenti.

Ora che il modello è stato creato e sviluppato, ci sono state poste delle domende sulle ulteriori possibilità offerte da questo strumento. In seguito ad un incontro, tenutosi il sette Aprile 2004, le richieste sono le seguenti:

- visualizzazione più chiara delle unità che lavorano. E' stato richiesto

soprattutto di poter separare la gestione inerente alla centrale operativa dalla gestione delle ambulanze. Per quanto riguarda queste ultime è inoltre emersa la necessità di separarle in base all'operatività delle stesse in città o in provincia. Inoltre è stato richiesto di sostituire i numeri delle unità con nomi in modo da rendere più chiaro il funzionamento del modello;

- visualizzazione migliore del tempo della simulazione, per evitare di doverlo determinare ogni volta a partire dai tic;

- molto interesse è stato manifestato riguardo alla possibilità di visualizzare il carico di lavoro complessivo del box ambulanze e delle postazioni di chiusura per poter verificare l'impegno dei singoli operatori sulle missioni che devono gestire contemporaneamente; sulla base di questi dati verranno poi provate delle possibili soluzioni per migliorare il modo di operare dei singoli operatori quando il sistema è sotto stress;

- in riferimento alle priorità di assegnazione delle ambulanze da parte del Box ambulanze è stato trattato l'argomento dei raggi d'azione delle singole unità per ambulanza e a questo scopo sono stati forniti, per quanto riguarda la provincia, i dati sulle competenze per comuni. Ci è stato però riferito che esiste una certa flessibilità dovuta al fattore umano nelle decisioni;

- è stato espresso il desiderio di poter utilizzare i dati in tempo reale;

- infine si è discusso sulla possibilità di utilizzare la simulazione per addestrare il personale della centrale operativa. Un ulteriore interesse del direttore della centrale operativa è sembrato quello di utilizzare la simulazione come sistema automatizzato per consigliare all'operatore umano il mezzo più opportuno da inviare.

A prescindere da quali delle precedenti richieste verranno sviluppate, è importante sottolineare come il lavoro svolto finora sia stato compreso e apprezzato. In particolare, è fondamentale la comprensione del ruolo che le simulazioni possono avere nell'analisi dei fenomeni sociali; se non si crede che questo strumento possa predire il futuro, e non lo si presenta come tale, credo che la sua diffusione sarà di grande aiuto nello studio dei numerosi fenomeni complessi che caratterizzano la nostra vita.

Appendice

DPR 27 marzo 1992

Atto di indirizzo e coordinamento alle regioni per la determinazione dei livelli di assistenza sanitaria di emergenza

(G.U. 31 marzo 1992, n.76, serie generale)

IL PRESIDENTE DELLA REPUBBLICA

Visto l'art. 4 della legge 30 dicembre 1991, n. 412, che detta norme in materia di assistenza sanitaria per l'anno 1992; Visto il comma 1 della richiamata norma che autorizza il Governo ad emanare un atto di indirizzo e di coordinamento per la determinazione dei livelli di assistenza sanitaria da assicurare in condizioni di uniformità sul territorio nazionale sulla base dei limiti e principi di cui alle successive lettere a), b), c), d) ed e);

Vista la deliberazione del CIPE in data 3 agosto 1990 che ha disciplinato, su conforme parere della Conferenza permanente per i rapporti tra lo Stato, le regioni e le province autonome, le priorità degli interventi relativi all'emergenza-urgenza sanitaria ed al rischio anestesiologico anche utilizzan-

do con vincolo di destinazione le risorse in conto capitale del Fondo sanitario nazionale;

Visto l'art. 22 dell'accordo collettivo nazionale per la regolamentazione dei rapporti con i medici addetti al servizio di guardia medica e di emergenza territoriale, reso esecutivo con decreto del Presidente della Repubblica 25 gennaio 1991, n. 41;

Visto il documento tecnico di intesa approvato dalla Conferenza Stato-regioni nella seduta del 14 gennaio 1992;

Visto il parere espresso dal Consiglio superiore di sanità in data 12 febbraio 1992;

Ritenuto che, nelle more della definizione degli standard organizzativi e dei costi unitari dei livelli di assistenza uniformi di cui all'art. 4 della legge 30 dicembre 1991, n. 412, la Conferenza Stato-regioni in data 7 febbraio 1992 ha definito l'intesa sul livello uniforme di assistenza del sistema dell'emergenza sanitaria;

Ritenuto che le spese in conto capitale per l'organizzazione del livello assistenziale fanno carico agli stanziamenti di cui all'art. 20 della legge 11 marzo 1988, n. 67, nonché agli

Appendici stanziamenti in conto capitale del Fondo sanitario nazionale, mentre quelle correnti fanno carico al Fondo sanitario nazionale di parte corrente di cui all'art. 51 della legge 23 dicembre 1978, ne; 833, nella misura che sarà determinata ai sensi del combinato disposto della norma di cui ai commi 1 e 16 dell'art. 4 della legge 30 dicembre 1991, n. 412;

Vista la deliberazione del Consiglio dei Ministri, adottata nella riunione del 13 marzo 1992, su proposta del Ministro della sanità, di concerto con il Ministro per le riforme istituzionali e gli affari regionali;

DECRETA:

È approvato il seguente atto di indirizzo e coordinamento delle attività delle regioni e delle province autonome di Trento e di Bolzano, in materia di emergenza sanitaria.

Art. 1 Il livello assistenziale di emergenza sanitaria

1. Ai sensi del comma 1 dell'art. 4 della legge 30 dicembre 1991, n. 412, il livello assistenziale di emergenza sanitaria da assicurare con carattere di uniformità in tutto il territorio nazionale è costituito dal complesso dei servizi

e delle prestazioni di cui agli articoli successivi.

Art. 2 Il sistema di emergenza sanitaria

1. Le regioni e le province autonome di Trento e di Bolzano organizzano le attività di urgenza e di emergenza sanitaria articolate su:

a. il sistema di allarme sanitario;

b. il sistema di accettazione e di emergenza sanitaria.

Art. 3 Il sistema di allarme sanitario

1. Il sistema di allarme sanitario è assicurato dalla centrale operativa, cui fa riferimento il numero unico telefonico nazionale 118. Alla centrale operativa affluiscono tutte le richieste di intervento per emergenza sanitaria. La centrale operativa garantisce il coordinamento di tutti gli interventi nell'ambito territoriale di riferimento.

2. Le centrali operative della rete regionale devono essere compatibili tra loro e con quelle delle altre regioni e delle province autonome di Trento e di

Bolzano in termini di standard telefonici di comunicazione e di servizi per consentire la gestione del traffico

Appendici interregionale. Con decreto del Ministro della sanità, di concerto con il Ministro delle poste e delle telecomunicazioni, entro sessanta giorni dalla data di pubblicazione del presente atto nella Gazzetta Ufficiale della Repubblica, sono definiti gli standard di comunicazione e di servizio.

3. L'attivazione della centrale operativa comporta il superamento degli altri numeri di emergenza sanitaria di enti, associazioni e servizi delle unità sanitarie locali nell'ambito territoriale di riferimento, anche mediante convogliamento automatico delle chiamate sulla centrale operativa del 118.

4. Le centrali operative sono organizzate, di norma su base provinciale. In ogni caso nelle aree metropolitane dove possono all'occorrenza sussistere più centrali operative, è necessario assicurare il coordinamento tra di esse.

5. Le centrali operative assicurano i radiocollegamenti con le autoambulanze e gli altri mezzi di soccorso coordinati e con i servizi sanitari del sistema di emergenza sanitaria del territorio di riferimento, su frequenze dedicate e riservate al servizio sanitario nazionale, definite con il decreto di cui al comma 2.

6. Il dimensionamento e i contenuti tecnologici delle centrali operative sono definiti sulla base del documento approvato dalla Conferenza Stato-regioni in data 14 gennaio 1992, che viene allegato al presente atto.

Art. 4 Competenze e responsabilità nelle centrali operative

1. La responsabilità medico-organizzativa della centrale operativa è attribuita nominativamente, anche a rotazione, a un medico ospedaliero con qualifica non inferiore ad aiuto corresponsabile, preferibilmente anestesista, in possesso di documentata esperienza ed operante nella medesima area dell'emergenza.

2. La centrale operativa è attiva per 24 ore al giorno e si avvale di personale infermieristico adeguatamente addestrato, nonché di competenze mediche di appoggio. Queste devono essere immediatamente consultabili e sono assicurate nominativamente anche a rotazione, da medici dipendenti con esperien-

za nel settore dell'urgenza ed emergenza e da medici del servizio di guardia medica di cui all'art. 22 dell'accordo collettivo nazionale per la regolamentazione dei rapporti con i medici addetti al servizio di guardia medica e di emergenza territoriale, reso esecutivo con decreto del Presidente della Repubblica 25 gennaio 1991, n. 41. La responsabilità operativa è affidata al personale infermieristico

Appendici professionale della centrale, nell'ambito dei protocolli decisi dal medico responsabile della centrale operativa.

Art. 5 Disciplina delle attività

1. Gli interventi di emergenza sono classificati con appositi codici. Il Ministro della sanità, con proprio decreto da emanarsi entro sessanta giorni dalla data di pubblicazione del presente atto nella Gazzetta Ufficia/e della Repubblica, stabilisce criteri e requisiti cui debbono attenersi le regioni e le province autonome di Trento e di Bolzano nella definizione di tale codificazione, anche ai fini delle registrazioni necessarie per documentare le attività svolte e i soggetti interessati.

2. L'attività di soccorso sanitario costituisce competenza esclusiva del Servizio sanitario nazionale. Il Governo determina gli standard tipologici e di dotazione dei mezzi di soccorso ed i requisiti professionali del personale di bordo, di intesa con la Conferenza Stato-regioni.

3. Ai fini dell'attività di cui al precedente comma, le regioni e le province autonome di Trento e di Bolzano possono avvalersi del concorso di enti e di associazioni pubbliche e private, in possesso dell'apposita autorizzazione sanitaria, sulla base di uno schema di convenzione definito dalla Conferenza Stato-regioni, su proposta del Ministro della sanità.

Art. 6 Il sistema di accettazione e di emergenza sanitaria

1. Fermo restando quanto previsto dall'art. 14 del decreto del Presidente della Repubblica 27 marzo 1969, n. 128, in materia di accettazione sanitaria, il sistema di emergenza sanitaria assicura:

a. il servizio di pronto soccorso;

b. il dipartimento di emergenza.

2. Le regioni e le province autonome di Trento e di Bolzano individuano gli ospedali sedi di pronto soccorso e di dipartimento di emergenza.

Art. 7 Le funzioni di pronto soccorso

1. L'ospedale sede di pronto soccorso deve assicurare, oltre agli interventi diagnostico-terapeutici di urgenza compatibili con le specialità di cui e dotato, almeno il primo accertamento diagnostico, clinico, strumentale e di laboratorio e gli interventi necessari alla stabilizzazione del paziente, nonché garantire il trasporto protetto.

Appendici 2. La responsabilità delle attività del pronto soccorso e il collegamento con le specialità di cui è dotato l'ospedale sono attribuiti nominativamente, anche a rotazione non inferiore a sei mesi, ad un medico con qualifica non inferiore ad aiuto, con documentata esperienza nel settore.

Art. 8 Le funzioni del dipartimento di emergenza

1.il dipartimento di emergenza deve assicurare nell'arco delle 24 ore, anche attraverso le unità operative specialistiche di cui è dotato l'ospedale, oltre alle funzioni di pronto soccorso, anche:

a. interventi diagnostico-terapeutici di emergenza medici, chirurgici, ortopedici, ostetrici e pediatrici;

b. osservazione breve, assistenza cardiologica e rianimatoria.

2. Al dipartimento di emergenza sono assicurate le prestazioni analitiche, strumentali e di immunoematologia per l'arco delle 24 ore giornaliere .

3. La responsabilità delle attività del dipartimento e il coordinamento con le unità operative specialistiche di cui è dotato l'ospedale sono attribuiti

nominativamente, anche a rotazione non inferiore a sei mesi, ad un primario medico, chirurgo o rianimatore, con documentata esperienza nel settore .

Art. 9 Le funzioni regionali

1. Le regioni e le province autonome di Trento e di Bolzano, anche a stralcio del Piano sanitario regionale, determinano, entro centoventi giorni, dalla data di pubblicazione del presente atto nella Gazzetta Ufficiale della Repubblica, la ristrutturazione del sistema di emergenza sanitaria, con riferimento alle indicazioni del parere tecnico fornito dal Consiglio superiore di sanità, in data 12 febbraio 1991, e determinano le attribuzioni dei responsabili dei servizi che compongono il sistema stesso.

Il provvedimento di cui al comma precedente determina altresì le modalità di accettazione dei ricoveri di elezione in relazione alla esigenza di garantire adeguate disponibilità di posti letto per l'emergenza. Con il medesimo provvedimento sono determinate le dotazioni di posti letto per l'assistenza subintensiva da attribuire alle singole unità operative

Appendici Art. 10 Prestazioni del personale infermieristico

1. Il personale infermieristico professionale, nello svolgimento del servizio di emergenza, può essere autorizzato a praticare iniezioni per via endovenosa

e fleboclisi, nonché a svolgere le altre attività e manovre atte a salvaguardare le funzioni vitali, previste dai protocolli decisi dal medico responsabile del servizio.

Art. 11 Onere del trasporto di emergenza

1. Gli oneri delle prestazioni di trasporto e soccorso sono a carico del servizio sanitario nazionale solo se il trasporto è disposto dalla centrale operativa e comporta il ricovero del paziente. Detti oneri sono altresì a carico del Servizio sanitario nazionale anche in mancanza di ricovero determinata da accertamenti effettuati al pronto soccorso.

Art. 12 Attuazione

1. All'attuazione di quanto disposto dal presente atto provvedono le regioni e le province autonome.

2. Le spese in conto capitale per l'organizzazione del livello assistenziale fanno carico come priorità agli stanziamenti di cui all art. 20 della legge 11 marzo 1988, n. 67, nonché agli stanziamenti in conto capitale del Fondo sanitario nazionale, mentre quelle correnti fanno carico al Fondo sanitario nazionale di parte corrente di cui all'art. 51 della legge 23 dicembre 1978, n. 833, nella misura che sarà determinata al sensi del combinato disposto delle

norme di cui ai commi 1 e 16 dell'art. 4 della legge 30 dicembre 1991, n 412.

3. Entro sei mesi dalla data di pubblicazione del presente atto nella Gazzetta Ufficiale della Repubblica, la Conferenza Stato regioni verifica le iniziative assunte, lo stato di attuazione del sistema emergenza sanitaria in ciascuna regione e provincia autonoma, nonché le risorse finanziarie impiegate. Allo scopo di attuare il sistema di emergenza sanitaria nelle regioni che non lo abbiano attuato, in tutto o in parte la Conferenza Stato-regioni approva uno schema tipo di accordo di programma, che, sottoscritto dal Ministro della sanità e dal Presidente della regione interessata, determina tempi, modi e risorse finanziarie per l'attuazione, anche avvalendosi di apposite conferenze dei servizi. L'accordo di programma può essere attivato anche prima della verifica, su richiesta della regione e provincia autonoma.

Il presente decreto sarà pubblicato nella Gazzetta Ufficiale della Repubblica italiana.

ComputationalAssemblerBasic.java

```
//  ComputationalAssemblerBasic.java

import swarm.Globals;
import swarm.defobj.Zone;
import swarm.objectbase.SwarmObjectImpl;

import swarm.collections.ListImpl;
import swarm.collections.ListIndex;

import java.io.*;
import java.util.*;

import java.lang.reflect.Method;

/**
 * The ComputationalAssemblerBasic class instances make
 * computational processes; we have a computationa assembler
 * instance for each unit
 *
 * THIS CLASS IS NOT USED DIRECTLY, BUT VIA THE INHERITING CLASS
 * ComputationalAssembler.java
 *
 * @author Pietro Terna; Carlo Desotgiu made the steps
 * 1001 and 1002;
 * AlessandroBalla and Roberto Crosetto discovered a severe bug
 * in 1002
 */
public class ComputationalAssemblerBasic extends SwarmObjectImpl{

/** the memory matrixes used in computations; we can add a lot
    of these */
MemoryMatrix mm0, mm1, mm2, mm3, mm4, mm5, mm6, mm7, mm8, mm9;
```

```
/** the current layer */
    public int layer;

/** the unit for which we are assembling for */
    public Unit myUnit;
/** the list of the order requiring computations */
    public ListImpl waitingList;
/** a computational specification set */
    ComputationalSpecificationSet
        pendingComputationalSpecificationSet;
/** a flag to avoid the control of the number of matrixes
*  this flag has to be used from a computational step using
*       an existing step and modifying on the fly the memoryand
*  modifying on the fly the memory matrix number of the used
*  computational step  */
    public boolean checkMatrixNumber;
/** the status of a specific computation */
    public boolean done;

/** internal displacements of rows and cols to be used when
*  invoking the methods of the BASIC computational steps via
*       other computational steps <br><br>
*
*  REMEMBER TO RESET ALWAYS TO ZERO THE DISPLACEMENTS AFTER
*  THE USE
*
* */
public int rd, cd;

// object to be addressed
OrderDistiller myOrderDistiller;
ESFrameModelSwarm eSFrameModelSwarm;

/**
```

```
* the constructor for ComputationalAssembler
*/
public ComputationalAssemblerBasic (Zone aZone,
        OrderDistiller od, ESFrameModelSwarm mo)
{
// Call the constructor for the parent class.
        super(aZone);

myOrderDistiller = od;
eSFrameModelSwarm=mo;

// the list of orders with computational processes
// to be assembled
waitingList = new ListImpl (getZone());

// the diplacements
rd=0; cd=0;
// the check of the matrix number
checkMatrixNumber=true;
}

/** setting the unit we are assembling for */
public void setUnit(Unit u){
myUnit = u;
}

/** adding an order to the waitingList */
public boolean setComputationalWaitingList(Order anOrder){
waitingList.addLast(anOrder);
return true;
}

/** making computations and eliminating the orders from our
*  waitingList */
```

```
public void checkingComputationsAndFreeingOrders(){
int n, nmax, t;
Order anOrder;

nmax = waitingList.getCount();
if(nmax>0)
for (n=0;n<nmax;n++)
{
        anOrder = (Order) waitingList.removeFirst();

        pendingComputationalSpecificationSet =
                (ComputationalSpecificationSet)
        anOrder.getPendingComputationalSpecificationSet();

        if(pendingComputationalSpecificationSet == null){
                System.out.println("Internal error in" +
                " computationalAssembler of unit " +
                myUnit.getUnitNumber() +
                " - order # " +
                anOrder.getOrderNumber() +
                "waiting for computations "+
                "has no computational specifications.");
                MyExit.exit(1);
          }

        // computations
        done=false;
        if(StartESFrame.verbose)
                System.out.println
                ("Making computations for order # " +
                pendingComputationalSpecificationSet.
                getOrderNumber() + " of type " +
                pendingComputationalSpecificationSet.
                getComputationType() + " with " +
```

```
pendingComputationalSpecificationSet.
getNumberOfMemoryMatrixesToBeUsed() +
" matrixes");

t=pendingComputationalSpecificationSet.
    getComputationType();

// transforming the code contained in t (a
// number in the range -1000 -1999) in a
// method label (c1000 c1999) via reflection
// methods
try{
    Class c = this.getClass();
    Method m = c.getMethod("c"+(-1*t),null);
    m.invoke(this, null);
}
catch (Exception e) {
    System.out.println(e);
    System.out.println("Invalid computational"+
                       " code in order: " +
                       anOrder.
                       getOrderNumber());
    MyExit.exit(1);
}


// exiting
if(done)
{
   if(StartESFrame.verbose)
        System.out.println
        ("Made computations for order # " +
        pendingComputationalSpecificationSet.
        getOrderNumber());
```

```
                        pendingComputationalSpecificationSet.
                        setDone();
                        // the last set at present has not
                        // effect it is introduced for
                        // possible future uses
                }

                else waitingList.addLast(anOrder);

        }
}

/**
* return the waiting list length
*/
public int getWaitingListLength()
{
        return waitingList.getCount();
}

/**
* checking if an order is in the waitingList
*/
public boolean thisOrderIsInTheWaitingList(Order o){

// the result is true if o is a member of waitingList;
// false in the opposite case
        return waitingList.contains(o);
}

/** removing an order form the waintingList */
public void removeThisOrderFromTheWaitingList(Order o){
        waitingList.remove(o);
}
```

```
/** computational operations with code -1995 (a code for
*  the checking phase of the program <br><br>
*
*  this computational code place a 1 in position 1,1 of
*  the unique received matrix and set the status to done
*/
public void c1995(){
if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1 && checkMatrixNumber)
    {
        System.out.println("Code -1995 requires one matrix;
                    " + pendingComputationalSpecificationSet.
                    getNumberOfMemoryMatrixesToBeUsed() +
                    " found in order # " +
                    pendingComputationalSpecificationSet.
                    getOrderNumber());
        MyExit.exit(1);
    }

mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
    getMemoryMatrixAddress(0);
layer=pendingComputationalSpecificationSet.
getOrderLayer();

mm0.setValue(layer,1+rd,1+cd,1.0);
mm0.print();

done=true;
} // end c1995

/** computational operations with code -1996 (a code for the
*  checking phase of the program <br><br>
*
```

```
*  this computational code place a 1 in position 0,0 of the
*  unique received matrix and set the status to done
*/
public void c1996(){
if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1 && checkMatrixNumber)
    {
        System.out.println("Code -1996 requires one matrix;
                    " + pendingComputationalSpecificationSet.
                    getNumberOfMemoryMatrixesToBeUsed() +
                    " found in order # " +
                    pendingComputationalSpecificationSet.
                    getOrderNumber());
        MyExit.exit(1);
    }

mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
   getMemoryMatrixAddress(0);
layer=pendingComputationalSpecificationSet.
   getOrderLayer();

mm0.setValue(layer,0+rd,0+cd,1.0);
mm0.print();

done=true;
} // end c1996

/** computational operations with code -1997 (a code for
*  the checking phase of the program <br><br>
*
*  this computational code place a 1 in position 1,0 of the
*  unique received matrix and set the status to done
*/
public void c1997(){
```

```
if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1 && checkMatrixNumber)
    {
        System.out.println("Code -1997 requires one matrix; "
                    + pendingComputationalSpecificationSet.
                    getNumberOfMemoryMatrixesToBeUsed() +
                    " found in order # " +
                    pendingComputationalSpecificationSet.
                    getOrderNumber());
        MyExit.exit(1);
    }

mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
    getMemoryMatrixAddress(0);
layer=pendingComputationalSpecificationSet.
    getOrderLayer();

mm0.setValue(layer,1+rd,0+cd,1.0);
mm0.print();

done=true;
} // end c1997

/** computational operations with code -1998 (a code for
*  the checking phase of the program <br><br>
*
*  this computational code place a number in position 0,0
*  of the unique received matrix and set the status to done
*/
public void c1998(){
if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1 && checkMatrixNumber)
    {
        System.out.println("Code -1998 requires one matrix;
```

```
                    " + pendingComputationalSpecificationSet.
                    getNumberOfMemoryMatrixesToBeUsed() +
                    " found in order # " +
                    pendingComputationalSpecificationSet.
                    getOrderNumber());
        MyExit.exit(1);
    }

mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
    getMemoryMatrixAddress(0);
layer=pendingComputationalSpecificationSet.
    getOrderLayer();

mm0.setValue(layer,0+rd,0+cd,1.0);
mm0.print();

done=true;
} // end c1998

/** computational operations with code -1999 (a code for
*  the checking phase of the program <br><br>
*
*  this computational code verifies position 0,0 of the
*  three received matrixes; only if these positions are
*  all not empty the code empties them and set the status
*  to done
*/
public void c1999(){
if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=3 && checkMatrixNumber)
    {
        System.out.println("Code -1999 requires three matrixes;
                    "+ pendingComputationalSpecificationSet.
                    getNumberOfMemoryMatrixesToBeUsed() +
```

```
                    " found in order # " +
                    pendingComputationalSpecificationSet.
                    getOrderNumber());
            MyExit.exit(1);
        }

mm0=(MemoryMatrix) pendingComputationalSpecificationSet.
        getMemoryMatrixAddress(0);
mm1=(MemoryMatrix) pendingComputationalSpecificationSet.
        getMemoryMatrixAddress(1);
mm2=(MemoryMatrix) pendingComputationalSpecificationSet.
        getMemoryMatrixAddress(2);
layer=pendingComputationalSpecificationSet.
        getOrderLayer();

if(! (mm0.getEmpty(layer,0+rd,0+cd) || mm1.getEmpty
            (layer,0+rd,0+cd) || mm2.getEmpty
            (layer,0+rd,0+cd) ) )
        {
            mm0.setEmpty(layer,0+rd,0+cd);
            mm1.setEmpty(layer,0+rd,0+cd);
            mm2.setEmpty(layer,0+rd,0+cd);
            done=true;
        }

} // end c1999


// computational steps 1001 and 1002 to launch
// recipes from recipes

/**
* computational operations with code -1001
*
```

```
* this computational code places the numbers
* of the recipes to launch from other recipes,
* via the c step 1002,
* in position 0,0 of the received matrixes and
* set the status to done.
*
* the number of the matrixes used here must be
* less or equal to totalMemoryMatrixNumber
* (the first m matrixes of these ones); beside
* this, m has to be less than 10
*/
public void c1001()
{
MemoryMatrix mm;

int m, i;

int[] recipesToLaunch;
recipesToLaunch = new int[10];

// from pending specifications related to the
// recipe launching this computational step
m = pendingComputationalSpecificationSet.
        getNumberOfMemoryMatrixesToBeUsed();
if(m > eSFrameModelSwarm.
        getTotalMemoryMatrixNumber() || m > 10)
{
        System.out.println("The number of types
        of the recipes launched by recipes\n"+
        "cannot exceed 10 or
        the totalMemoryMatrixNumber.");
        MyExit.exit(1);
}
```

```
try
{
    BufferedReader fileRecipesFromRecipes = null;
    StringTokenizer t;
    String line = " ";

    fileRecipesFromRecipes = new BufferedReader(new
    FileReader("recipeData/recipesFromRecipes.txt"));

    t = new StringTokenizer(line, " ");
    for (i = 0; i < m; i++)
    {
        t=MyReader.check("recipeData/
        recipesFromRecipes.txt",
        fileRecipesFromRecipes, line, t);
        recipesToLaunch[i] = Integer.
        parseInt(t.nextToken());
    }

    fileRecipesFromRecipes.close();
}
catch (FileNotFoundException fnfe)
{
    System.out.println(fnfe);
    MyExit.exit(1);
}
catch (IOException e)
{
    System.out.println("IOException:"
    + e.toString());
    MyExit.exit(1);
}

for (i = 0; i < m; i++)
```

```
{
    mm =(MemoryMatrix)
     pendingComputationalSpecificationSet.
          getMemoryMatrixAddress(i);
     layer=pendingComputationalSpecificationSet.
     getOrderLayer();

    mm.setValue(layer,0+rd,0+cd,
          recipesToLaunch[i]);
}
done=true;
} // end c1001

public void c1002()
{
MemoryMatrix mm;
int aRecipe;

if(pendingComputationalSpecificationSet.
   getNumberOfMemoryMatrixesToBeUsed()!=1 &&
          checkMatrixNumber)
   {
        System.out.println("Code -1002 requires one matrix;
                    " + pendingComputationalSpecificationSet.
                    getNumberOfMemoryMatrixesToBeUsed() +
                    " found in order # " +
                    pendingComputationalSpecificationSet.
                    getOrderNumber());
        MyExit.exit(1);
   }

mm =(MemoryMatrix) pendingComputationalSpecificationSet.
     getMemoryMatrixAddress(0);
layer=pendingComputationalSpecificationSet.
```

```
    getOrderLayer();

aRecipe =(int) mm.getValue(layer,0+rd,0+cd);

// a recipe to be launched  at the end of this cycle
myOrderDistiller.setComputationalRecipe(aRecipe);


done=true;
} // end c1002

}
```

OrderDistiller.java

```
//OrderDistiller.java modified by Pietro Terna
//(look below at rows signed //pp)
//OrderDistiller.java modified by Marco Lamieri
//and Francesco Merlo (look below at rows signed //lm)
import swarm.Globals;
import swarm.defobj.Zone;
import swarm.objectbase.SwarmObjectImpl;
import swarm.collections.ListImpl;

/**
 * This class is used to read data from two worksheets.
 * The first one contains the list of recipes of our
 * virtual enterprise.<br/>
 * The second one contains a sequence of orders to be
 * launched, shift by shift, in order to makeshift by
 * shift, in order to make the daily production activities.
 *
 * @author Cristian Barreca, Elena Bonessa, Antonella Borra.
 * Modified by: Marco Lamieri,Francesco Merlo.
 * Refactored by Francesco Merlo. Carlo Desotgiu added the
 * capability ofdealing with recipes launched by
 * computational steps;Alessandro Balla and Roberto Crosetto
 *
 *
 */
public class OrderDistiller extends OrderGenerator
{
   String
         semicolon = ";",
         gate = "#",
         layer="l",
         computation="c",
```

```
        //The first worksheet of orders
        orderSequences1 ="recipeData/orderStartingSequence.xls",
        //The second worksheet of orders
        orderSequences2 ="recipeData/orderSequence.xls",
        //The worksheet of recipes
        recipeFile = "recipeData/recipes.xls",
        currentOrderSequenceWorksheet;
int
        orderCount = 0,
        currentLayer = 0,
        i = 0;
boolean
        firstTime = true,
        orderSequenceWorksheetOpen = false;
int[]
        currentOrder;
int[]
        computationalRecipes;
ExcelReader
        orderSequenceWorksheet,
        recipeWorksheet;
AssigningTool
        assigningTool;//pt
Recipe
        aRecipe;
Unit
        aUnit;
Order
        anOrder;
ListImpl
        unitList,
        endUnitList,
        orderList,
        recipeList;
```

```
// CONSTRUCTOR
public OrderDistiller (Zone aZone, int msn, int msl, ListImpl ul,
        ListImpl eul, ListImpl ol, int tln, ESFrameModelSwarm mo,
        AssigningTool at)
{ //pt
   super(aZone, msn, msl, ul, eul, ol, tln, mo, at);

   // dealing with computational recipes, added in real time by
   // computational steps
   computationalRecipes = new int[1000];
   for (int i = 0; i < 1000; i++)computationalRecipes[i] = 0;

   unitList=ul;
   endUnitList=eul;
   orderList=ol;
   assigningTool=at; //pt
}

/**This is the method containing the iterator needed to launch
* the daily production of recipes. It take a look at the
* orderSequence arrays to determine which recipes must be done
* and how many times. A request for which unit can do the first
* production phase of each recipe will be done to units or
* endUnits.
*/
public void distill()
{

if(firstTime == true)
{
   currentOrderSequenceWorksheet = orderSequences1;
   firstTime = false;
}
```

```
else
   currentOrderSequenceWorksheet = orderSequences2;

   if(!orderSequenceWorksheetOpen)
   {
        orderSequenceWorksheet = new ExcelReader
            (currentOrderSequenceWorksheet);
        orderSequenceWorksheetOpen = true;
   }

   checkForComment(); //see below
   //Getting the number of the shift
   int shiftNumber = orderSequenceWorksheet.getIfInteger();

   boolean isEndOfShift = false;
   while(!isEndOfShift)
   {
      if(orderSequenceWorksheet.checkForLabelCell())
      {
        String aString = orderSequenceWorksheet.getStrValue();
        if(aString.equals(semicolon))
        {
          // launching recipes from computational steps

          i=0;

          while (computationalRecipes[i]!= 0)
          {

            boolean recipeCodeFound = false;
            for (int r = 0; r < recipeList.getCount() &&
                !recipeCodeFound; r++)
            {
```

```
      aRecipe = (Recipe) recipeList.atOffset(r);
      if(aRecipe.getRecipeCode() ==
              computationalRecipes[i])
      {
        computationalRecipes[i]=0;
        recipeCodeFound = true;
      }
  }

  if(recipeCodeFound == false && i != 0)
  {
    System.out.println("OrderDistiller error: There
      is no Recipe with code '" +
      computationalRecipes[i-1] +"
      ' in computationalRecipes["+ i +"] check '"
      + recipeFile + "' and '" +
      currentOrderSequenceWorksheet + "'");
    MyExit.exit(1);
  }

  if(recipeCodeFound == true)
  {
    orderCount++;
    anOrder = new Order(getZone(), orderCount,
      Globals.env.getCurrentTime(),
    aRecipe.getLength(), aRecipe.getRecipeSteps(),
      eSFrameModelSwarm, endUnitList);

    anOrder.setRecipeName(aRecipe.getRecipeName());
  //setting the layer (from 0 to totalLayerNumber-1)
    if(currentLayer <0 ||
      currentLayer >= totalLayerNumber)
    {
      System.out.println("A layer in the sequence is
```

```
                    not inclued\n"+"in the interval from 0 to
                    totalLayerNumber-1");
                MyExit.exit(1);
            }
            anOrder.setOrderLayer(currentLayer);
            // add the active orders to the general order list
            // (they will be eliminated when dropped in a unit,
            // being finished); this list has been introduced
            // for accounting purposes [may be it would
            // be better substitute it with a get to the units
            // to know their waiting lists]
            orderList.addLast(anOrder);
            // sending the order to the first production unit
            // (we are acting as the FrontEnd of the ES)
            assigningTool.assign(anOrder);
         }
         i++;
      }


      isEndOfShift = true;
      break;
   }
   else
   {
      orderSequenceWorksheet.goBack();
   }
}

//Reading the order from the worksheet
readOrderFrom(orderSequenceWorksheet);//see below

//Checking if the recipe cose exists
boolean recipeCodeFound = false;
```

```
for (int r = 0; r < recipeList.getCount() &&
    !recipeCodeFound; r++)
{
  aRecipe = (Recipe) recipeList.atOffset(r);
  if(aRecipe.getRecipeCode() == currentOrder[0])
  recipeCodeFound = true;
}

if(recipeCodeFound == false)
{
  System.err.println("OrderDistiller error: There is no Receipe
  with code '" + currentOrder[0] +
  "' check '" + recipeFile + "' and '" +
  currentOrderSequenceWorksheet + "'");
  MyExit.exit(1);
}

for(int q = 0; q < currentOrder[1]; q++)
{
 orderCount++;
 anOrder = new Order(getZone(), orderCount,
   Globals.env.getCurrentTime(),
 aRecipe.getLength(), aRecipe.getRecipeSteps(),
   eSFrameModelSwarm, endUnitList);

 anOrder.setRecipeName(aRecipe.getRecipeName());
 // setting the layer (from 0 to totalLayerNumber-1)
 if(currentLayer <0 || currentLayer >= totalLayerNumber)
 {
   System.out.println("A layer in the sequence is not inclued\n"
     + "in the interval from 0 to totalLayerNumber-1");
         MyExit.exit(1);
 }
 anOrder.setOrderLayer(currentLayer);
```

```
  // add the active orders to the general order list (they will be
  // eliminated when dropped in a unit, being finished); this list
  // has been introduced for accounting purposes [may be it would
  // be better substitute it with a get to the units to know their
  // waiting lists]
  orderList.addLast(anOrder);

  // sending the order to the first production unit (we are acting
        as the Front End of the ES)
  assigningTool.assign(anOrder);

  if(StartESFrame.verbose)
  {
    System.out.println("OrderDistiller: Order #" +
        anOrder.getOrderNumber() +
    " with Name '"+ anOrder.getRecipeName() +"' and layerNumber '"
    + anOrder.getOrderLayer() + "' is starting production");
  }
 }
}
if(orderSequenceWorksheet.eof())
   orderSequenceWorksheetOpen = false;
}

/**
* This method is used to collect the names of the units and of the
* end units, so that it can operate the check of corrispondency
* between the production phases required by recipes and the phases
* of production the units can do.
*/
public void setDictionary()
{
  super.setDictionary();
```

```
  //The List of Recipes
  recipeList = new ListImpl(getZone());
  recipeWorksheet = new ExcelReader(recipeFile);

  while(! recipeWorksheet.eof())
  {
    aRecipe = new Recipe(getZone());
    aRecipe.setRecipeFrom(recipeWorksheet);
    recipeList.addLast(aRecipe);
  }

  if(StartESFrame.verbose)
  {
    System.out.println("OrderDistiller: " + recipeList.getCount()
        + "recipes readed.");
    for (int i = 0; i < recipeList.getCount(); i++)
    {
      aRecipe = (Recipe) recipeList.atOffset(i);
      System.out.println("Recipe #" + i + " with name '" +
      aRecipe.getRecipeName() + "' and code '"
      + aRecipe.getRecipeCode() +"'");
    }
  }
}


/** setting in real time the recipes coming from computational
    steps */
public void setComputationalRecipe(int aRecipeToBeAdded)
{
  int k = 0;
  while (computationalRecipes[k] != 0)
  {
    k++;
```

```
  }

  computationalRecipes[k] = aRecipeToBeAdded;
}



// *****************
// PRIVATE FUNCTIONS
// *****************
private int[] readOrderFrom(ExcelReader e)
{
  orderSequenceWorksheet = e;
  currentOrder = new int[2];

  checkForComment();
  checkForLayer();
  currentOrder[0] = orderSequenceWorksheet.getIfInteger();
  orderSequenceWorksheet.getIfString();
  currentOrder[1] = orderSequenceWorksheet.getIfInteger();

  return currentOrder;
}

private void  checkForLayer()
{
  if(orderSequenceWorksheet.checkForLabelCell())
  {
    String isLayer = orderSequenceWorksheet.getStrValue();

    if(isLayer.equals(layer))
    {
      currentLayer = orderSequenceWorksheet.getIntValue();
```

```
      if(StartESFrame.verbose)
      System.out.println("OrderDistiller: Current Layer is
        #" + currentLayer);
    }
    else
    {
      orderSequenceWorksheet.goBack();
    }
  }
}

private void checkForComment()
{
  if(orderSequenceWorksheet.checkForLabelCell())
  {
    String isGate = orderSequenceWorksheet.getStrValue();
    if(isGate.equals(gate))
    {
        if(StartESFrame.verbose)
        {
          System.out.println("OrderDistiller:
                there is a comment ...");
          System.out.print("# ");
        }
        while(!orderSequenceWorksheet.eol())
        {
          String comment = orderSequenceWorksheet.getStrValue();
          if(StartESFrame.verbose)
          System.out.print(" " + comment + " |");
        }
        String comment = orderSequenceWorksheet.getStrValue();
        if(StartESFrame.verbose)
        System.out.println(" " + comment + " #");
```

```
            //Checking for other lines of comments
            checkForComment();
        }
        else
        {
            orderSequenceWorksheet.goBack();
        }
    }
 }

} // OrderDistiller
```