Capitolo 5
jES - Java Enterprise Simulator

Il progetto di simulazione d'impresa 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 P. 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 programma di simulazione che, grazie all'uso di simboli, in questo caso si parla di linguaggio di programmazione, riproduce i fenomeni che la teoria intende spiegare. jES (abbreviazione che sta per Java

Enterprise Simulator) è un modello, realizzato attraverso un programma e che poggia sulla libreria di funzioni di Swarm, che permette di riprodurre le realtà delle imprese[1].

Il modello jES può essere utilizzato per due scopi:

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

Il primo caso, quello maggiormente utilizzato, permette di mettere alla prova la conoscenza dell'impresa da parte del ricercatore che ricostruisce l'azienda per poterne simulare il comportamento. Infatti se nella ricostruzione si dovessero saltare dei passaggi fondamentali per il funzionamento dell'impresa la simulazione potrebbe non funzionare o dare risultati che si discostano dalla realtà.

Una volta ricostruita l'azienda si potrà utilizzare il simulatore per testare il comportamento dell'impresa e per provare ad immaginare dei cambiamenti all'interno del modello dell'impresa simulata o provare ad immaginare scenari differenti per la costruzione stessa dell'impresa. In qualunque caso se la simulazione precedentemente costruita è riuscita a simulare con realismo, la realtà che si vuole studiare, i cambiamenti saranno testati in modo credibile sull'impresa virtuale ed eventualmente, se i cambiamenti daranno buoni risultati, si potrà pensare di apportare delle modifiche all'interno dell'impresa reale.

Il modello jES sino ad ora è stato utilizzato per ricostruire la VIR Spa, che è un'impresa che produce valvole idrauliche e per la quale la simulazione ha suggerito un cambiamento che è stato apportato[2], e la

[1] Attualmente è in fase di sviluppo una nuova branca di jES ha l'ambizione di studiare la nascita e lo sviluppo delle imprese. Questo nuovo progetto prende il nome di jESEvol. (http://web.econ.unito.it/terna/)

[2] Il cambiamento suggerito dalla simulazione in realtà è stato apportato prima che la simulazione fosse pronta a suggerirlo ma successivamente la simulazione è stata in grado di apportare il cambiamento in questione esattamente come intuito dai Manager che vi lavorano all'interno.

BasicNet Spa, che è invece un'impresa non tradizionale che produce abbigliamento. Attualmente si è sviluppato con questa tesi e con altre tesi analoghe il primo modello di simulazione di un'organizzazione che nel caso specifico è il 118.

Il secondo aspetto del modello è improntato maggiormente su un'analisi teorica dei fenomeni che interessano temi come la creazione della conoscenza all'interno dell'impresa o l'innovazione imprenditoriale. In questo caso la simulazione aiuta a creare teorie sulla natura dell'impresa e come si può leggere in D. 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 sui concetti è tradotto interamente in forma di programmazione che permette al computer di eseguire la simulazione.

Su questa strada si sta sviluppando un progetto che prende il nome di jESEvol e che si propone di studiare come nascono e crescono, o "muoiono", le imprese all'interno di un mondo.

In ogni caso quel che è importante sottolineare è che ciò che la simulazione va a costruire è un modello di un'impresa reale o virtuale ma comunque sempre un modello.

Per quanto riguarda la discussione sul significato dei modelli rimando al capitolo tre di questa stessa tesi. Per ora mi limito ad inserire un passo tratto da Gilbert e Terna (2000, p.58) dove vengono illustrati i differenti modi con cui si possono spiegare i fenomeni sociali.

Da Gilbert e Terna:

"[…] 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."

5.1 La descrizione del mondo

Il primo approccio sull'uso di jES introduce l'esistenza di due modi indipendenti per descrivere e rappresentare il mondo in cui l'impresa opera.

Il cuore della simulazione è proprio la divisione che esiste tra i formalismi:

- What to Do (d'ora in poi indicato solo con WD),
- which is Doing What (d'ora in poi solo DW).

Quindi esistono due modi di descrivere il mondo in cui l'impresa opera.

Con la prima descrizione si analizza il *che Cosa fare* (WD, What to DO): il formalismo adottato permette la rappresentazione di tutti gli obiettivi che l'azienda intende perseguire o che deve affrontare.

Con la seconda descrizione si analizza il *Chi fa che cosa* (DW, which is Doing What): il formalismo adottato permette in questo caso la rappresentazione delle risorse che l'impresa ha a disposizione.

Quando la simulazione "gira" nel computer congiunge le due descrizioni facendo, attraverso l'interazione degli gli oggetti descritti nel lato WD e nel lato DW, emergere la realtà dell'impresa.

Per proseguire con la nostra analisi sul funzionamento del modello è molto importante definire da subito tre concetti che ricorreranno spesso nella descrizione del funzionamento del modello. Questi sono 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, potrebbe essere un macchinario, presente all'interno del mondo, e più specificamente all'interno dell'impresa o all'esterno della stessa, in grado di eseguire alcuni step necessari per portare a compimento un ordine.

Riassumendo possiamo dire che all'interno del nostro mondo simulato esistono degli ordini che devono essere presi in carico dall'impresa. Per poter portare a compimento l'ordine l'impresa, con le sue unità produttive, deve compiere dei passi. L'informazione relativa ai 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 quindi può essere visto come un modulo che riporta la storia dei passi già fatti e la sequenza di quelli da compiere.

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 eseguito solo quando l'unità si libera. Chiaramente tutte le unità produttive lavorano in modo autonomo le une dalle altre ed in parallelo.

Dal fatto che si è accennato alle code d'attesa della simulazione si deduce che la simulazione è in grado di simulare anche lo scorrere del tempo, e qualcuno sostiene che questo sia un terzo formalismo che descrive *Quando fare cosa* (WDW, When Doing What).

La simulazione quindi come già detto simula anche il trascorrere del tempo. Deciso il livello di dettaglio che si intende utilizzare, l'utente deve associare la più piccola unità di tempo reale presa in conderazione (un secondo, un minuto, dieci minuti, un ora, . . .) ad un tick della simulazione.

Le fasi lavorative sono rappresentate come gruppi di tick; una volta deciso che un tick rappresenta dieci secondi, perché questo è il livello di dettaglio che intendiamo utilizzare, e se, ad esempio, si vogliono simulare interventi di otto minuti, un intervento all'interno della simulazione sarà composto da 48 tick.

L'impresa simulata riceve gli ordini dal mercato se si tratta di un'impresa che lavora *just in time* oppure da un addetto alla programmazione se si tratta di un'impresa classica.

Il caso di simulazione di un processo di emergenza sanitaria assomiglia molto al caso dell'impresa che lavora *just in time* e che quindi riceve ordini dal mercato, solo che nel nostro caso il mercato è rappresentato dalle persone che richiedono aiuto.

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 secondo una sequenza le ricette preparate dall'utente;

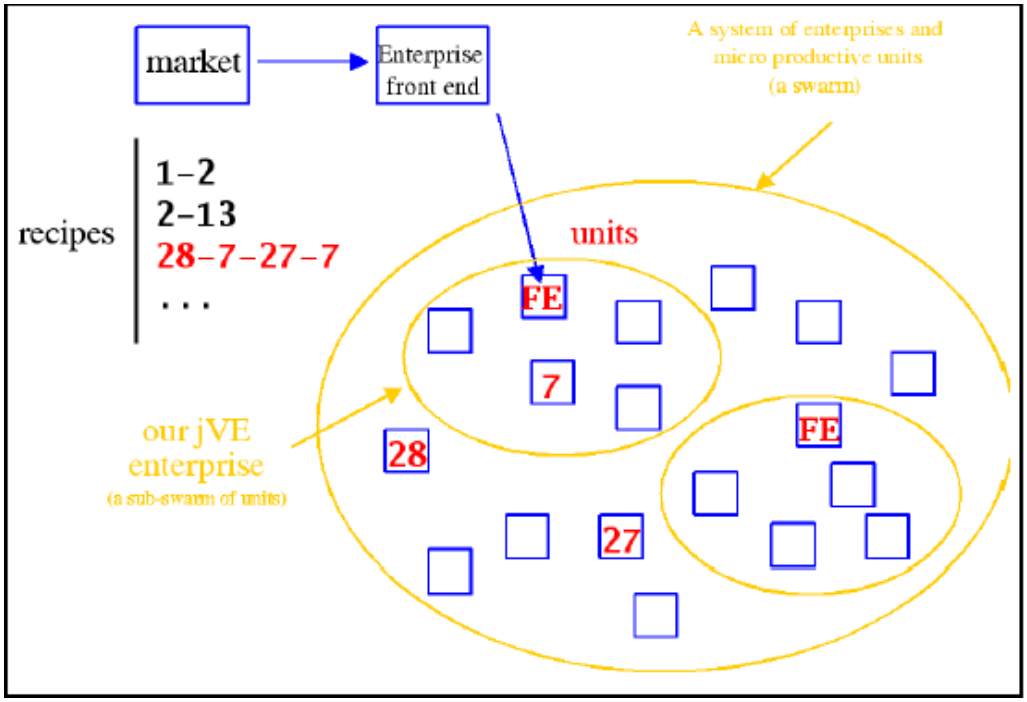


Figura 1- Schema generale del modello jES

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.

Quindi l'unità che ha in carico l'ordine, cioè l'unità che ha svolto l'ultimo passo della lavorazione, controlla il passo successivo della ricetta, individua l'unità che è in grado di svolgerlo e gli passa l'ordine a quest'ultima

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. Uno di questi implica che tra più unità disponibili a fare una certa lavorazione questa viene assegnata alla prima unità che compare nella lista. Un altro criterio dice di assegnare la lavorazione all'unità con la coda di attesa più corta e l'ultimo criterio dice di assegnare a caso la lavorazione 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 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 dislocate all'esterno nel mondo o in un'altra impresa presente nel mondo.

Il modello non crea alcun tipo di problema di rappresentazione.

Vediamo ora un semplice esempio introduttivo del mondo che fino ad ora è stato descritto.

In *figura* 2 possiamo 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 è il caso del nostro esempio.

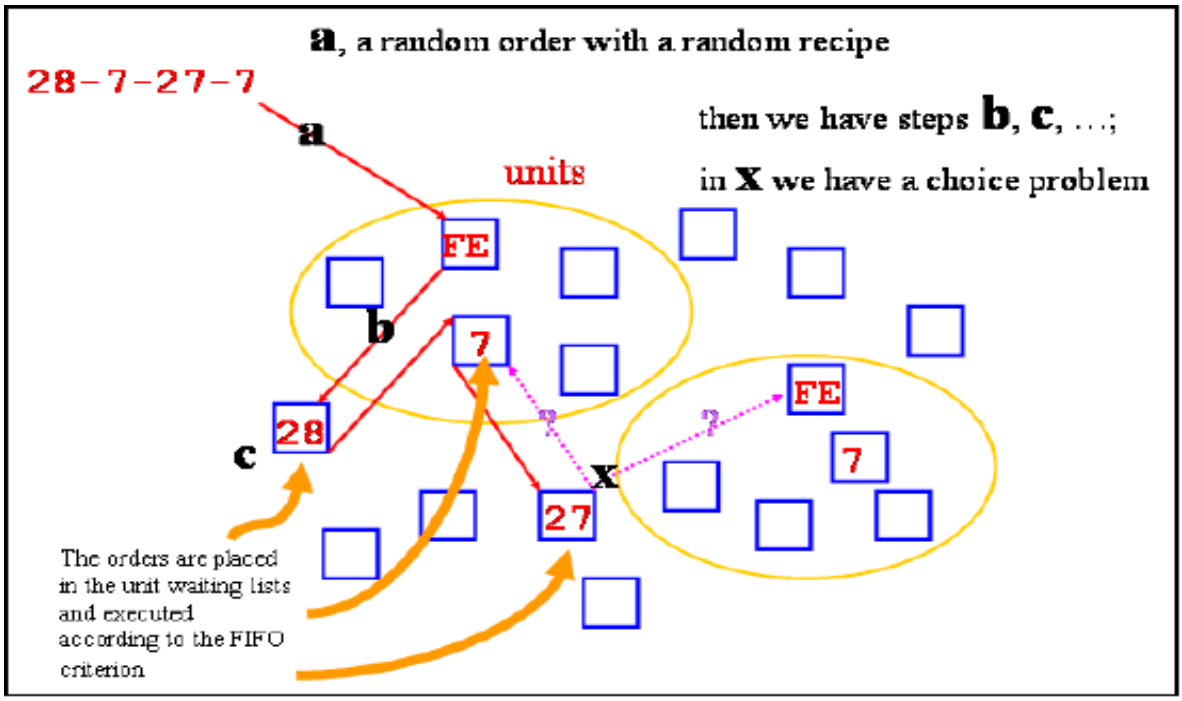


Figura 2- Le ricette

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

28	7	27	7

Tabella 1- Ricetta della figura

Il font end dell'impresa prende in carico la ricetta e la passa alla prima unità produttiva che è l'unità che può svolgere lo step 28. nel nostro esempio l'unità produttiva che prende in carico la ricetta è esterna all'impresa. Una volta che l'unità *28* 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 nostro esempio avremo che l'ordine passa dall'unità 28, all'unità 27 all'unità 7 della quale però esistono due "esemplari", o meglio 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 e che sono esattamente:

- la prima delle unità che possono eseguire la lavorazione presente nel file *unitBasicData.txt*;
- la scelta delle unità avviene in modo casuale;
- la scelta dell'unità avviene tra le unità che hanno la coda più corta.

Vedremo ora nel dettaglio i due formalismi che ci permettono di descrivere cosa deve fare l'impresa (aspetto WD) e chi è in grado di farlo, all'interno o all'esterno dell'impresa (aspetto DW). Faremo anche un breve cenno al terzo formalismo di cui si è accennato in precedenza che riguarda il tempo (aspetto WDW).

5.2 Che cosa fare (WD)

All'interno della simulazione del mondo abbiamo degli ordini da compiere, questi vengono descritti all'interno del formalismo delle ricette.

Le ricette sono formate da una sequenza di passi produttivi che rappresentano le lavorazioni necessarie per portare a termine una lavorazione.

La ricetta classica, ossia una lavorazione sequenziale per produrre un determinato bene o servizio si presenta così:

7 *s* 20	8 *m* 1	27 *h* 3	…

Tabella 2- Formalismo di una generica ricetta

Come si vede chiaramente dal semplice esempio riportato in tabella 1 le ricette, nella descrizione WD, vengono rappresentate attraverso un

formalismo numerico. I passi della ricetta sono espressi in modo univoco con una cifra che rappresenta la fase di produzione necessaria per la produzione del bene o del servizio dell' impresa (o organizzazione). Ad ogni fase di produzione viene affiancato il tempo necessario per il compimento dello stadio produttivo. Nell'esempio riportato in tabella è rappresentato un semplice processo produttivo che ha bisogno di tre fasi diverse per essere completato. La prima fase è compiuta da quelle unità che sono in grado di compiere il passo *7* e dura *20 secondi*, la seconda fase deve essere prodotta dalle unità che sono in grado di fare il processo *8* e tale processo durerà *1 minuto* e la terza ed ultima fase sarà compiuta da unità in grado di fare il processo *27* e tale processo durerà *3 ore.*

Terminata la ricetta l'ordine è stato completato ed è pronto ad "uscire" dall'impresa.

Particolarmente importante all'interno della ricetta è la specificazione di tempo necessaria per compiere i processi produttivi. Il tempo può essere espresso in:

1. **d**: giorni;
2. **h**: ore;
3. **m**: minuti;
4. **s**: secondi.

Ma le tipiche lavorazioni aziendali non sono solo sequenziali, per questo motivo all'interno del nostro modello sono stati introdotti alcuni passi speciali che permettono di descrivere:

- la ramificazione delle ricette;
- atti di approvvigionamento;
- lavorazioni per lotti.

Abbiamo detto in precedenza che le imprese possono produrre semilavorati utili alla produzione di una fase successiva di lavorazione o allo scambio con altre imprese. La produzione di questi prodotti sottintende

la presenza di un magazzino nel quale stoccare questi prodotti. Il magazzino è rappresentato all'interno del nostro modello come una *endUnit* e per stoccare la materia all'interno di questa unità particolare è necessario modificare leggermente il formalismo della ricetta, che diventa:

7 *s* 20	8 *m* 1	27 *h* 3	***e*** 40

Tabella 3 - Formalismo di una generica ricetta per un semilavorato

Nell'esempio abbiamo che il nostro ordine sarà stoccato in un'unità magazzino che ha codice *40*.

5.2.1 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 all'interno del modello proprio per affrontare il problema di cui sopra.

Un *sequential batch*, il quale viene espresso con il simbolo \ all'interno del formalismo delle ricette, deve essere associato ad una fase produttiva, si veda ad esempio la tabella.

7 *s* 20	8 *m* 1	27 *h* 3 \ 20

Tabella 4 - Sequential batch

In questo caso abbiamo che il passo 21 deve essere compiuto da un'unica unità produttiva contemporaneamente su 20 ordini. Naturalmente

dopo la lavorazione in *sequential batch* 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*.

5.2.2 Procurement

Abbiamo prima accennato a come i semilavorati possono essere stoccati in magazzino, ora si deve però capire come utilizzare questi semilavorati e quindi come recuperarli dal magazzino.

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 fatta:

7 *s* 20	*p* 2 30 27 8 *h* 3

Tabella 5 - Procurement

Dall'esempio riportato in tabella possiamo 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*.

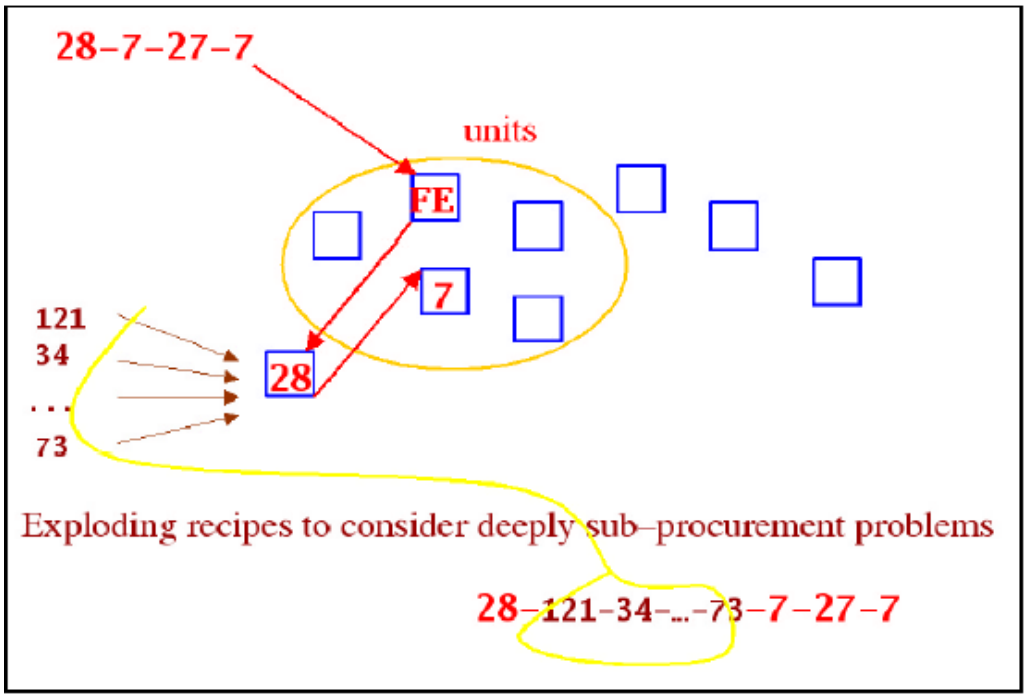


Figura 3- Processo di procurement

Volendo rappresentare graficamente il processo di procurement, in *figura 3* si può ben vedere come il passo 7 per èpoter essere concluso ha bisogno di alcuni semilavorati.

I casi di fornitura di componenti crea in ogni azienda problemi di sincronizzazione tra l'inizio delle diverse produzioni e le relative componenti

In questo ultimo caso l'unità che si occuperà della fornitura è considerata una scatola nera.

Grazie a questo formalismo per la creazione di sotto-ricette produttive è possibile esplodere a piacere ogni processo aziendale in più sub-processi in funzione del livello di dettaglio che si desidera utilizzare.

5.2.3 Or

Tipico di molte aziende è avere la possibilità di scegliere come produrre un determinato bene o come eseguire una determinata lavorazione.

La caratteristica di queste possibilità di scelta è che i prodotti finali risulterebbero uguali ma i processi seguiti dagli stessi, 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.

Riportiamo un esempio:

7 *s* 20	|| 2 *m* 4	|| 5 *s* 40

Tabella 6- Processo Or

Nell'esempio riportato vediamo che l'ordine può essere completato in tre modi alternativi, eseguendo il passo *7* o il passo 2 o il passo 5.

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;
- si esegua il secondo ramo;
- il ramo da eseguire sia scelto in modo casuale;
- si esegua il ramo che presenta il primo passo con la minore coda di produzione;
- il ramo sia scelto con l'ausilio di un passo computazionale di cui si parlerà in modo approfondito più avanti.

5.3 Chi fa che cosa (DW)

Passo ora ad illustrare il formalismo che serve ad indicare chi fa 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:

- unità produttive semplici;
- unità produttive complesse
- endUnit.

Cominciamo col descrivere le unità semplici per poi approfondire l'argomento e passare alla descrizione delle unità complesse e delle endUnit.

5.3.1 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 nel quale sono descritte le unità 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: concettualmente si potrebbe dire che l'unità produttiva fa semplicemente passare il tempo (simulato) necessario al completamento del passo.

Le differenti unità si contraddistinguono in base ad una cifra; ad agni unità produttiva sono associati uno o più passi che essa è in grado di svolgere.

Tutte le unità vengono descritte all'interno del file *unitBasicData.txt*.

Il file di cui sopra è fatto in questo modo:

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- unitBasicData.txt

La prima riga del file è obbligatoria ed indica:

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

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

il codice dell'unità: ad ogni unità presente nella imulazione è attribuito un numero univoco;

fase di produzione: indica quale passo l'unità produttiva è in grado di svolgere all'interno della simulazione[3];

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.

5.3.2 Unità complesse

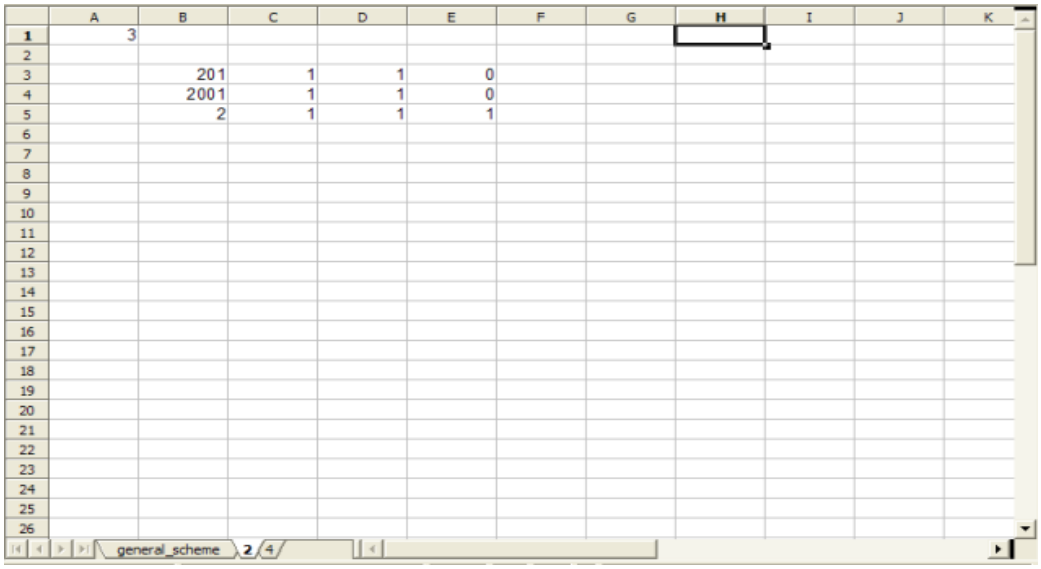
In questo caso le unità create ed utilizzate nella simulazione dovranno essere in grado di fare, oltre all'unica fase precedentemente descritta all'interno del file *unitBasicData* anche altre fasi produttive che verranno però descritte in un alto file chiamato *units.xls*.

Per creare unità complesse è necessario mettere all'interno del file *unitBasicData* nella posizione che indica la fase di produzione (*prod.phase*) uno zero e poi creare un file excel chiamato *unit.xls* in cui vengono indicate le fasi produttive che l'unità complessa e in grado di fare.

Riprendendo il file *unitBasicData* riportato *in figura 4* possiamo notare che la seconda e la quarta unità descritte nel file (e corrispondenti alle 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 fare ne foglio excel chiamato *units.xls*.

Il foglio excel si presenta così:

[3] Se la fase di produzione è segnata "0" allora bisogna andare a cercare le fasi produttive che l'unità è in grado di fare nel file delle unità complesse.



	A	B	C	D	E	F	G	H	I	J	K
1	3										
2											
3		201	1	1	0						
4		2001	1	1	0						
5		2	1	1	1						
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											

general_scheme | 2 | 4

Figura 5- Units.xls

In questo foglio è stato indicato:

1. il numero di fasi che l'unità è in grado di svolgere.
   Nell'esempio *l'unità 2* è in grado di realizzare tre fasi produttive. Questo aspetto è indicato nella prima casella del foglio excel che si trova in alto a sinistra;
2. nella colonna successiva sono indicati i codici di ogni fase produttiva che *l'unità 2* è in grado di fare;
3. i codici indicanti le fasi produttive sono seguiti (sulla medesima riga) dalle indicazioni di costi variabili e costi fissi;
4. L'informazione successiva a quella dei costi è quella che ci indica l'utilizzo o meno da parte dell'unità complessa degli *stand alone inventory*. L'indicazione sulla capacità o meno dell'unità di fare quest'operazione è indicata con 1 o 0 ;

Ogni foglio del file *unit.xls* corrisponde ad un'unica unità complessa e ad ogni foglio è assegnato il nome dell'unità complessa che è descritta al suo interno.

5.3.3 EndUnits

Come già detto in precedenza, i componenti della produzione possono essere stoccati per essere poi riutilizzati da altre unità produttive.

Perché le materie stoccate siano oggetto di procurement da parte delle unità è necessario che tali componenti, al termine delle loro lavorazioni, vengono 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* e questo file ha questo aspetto:

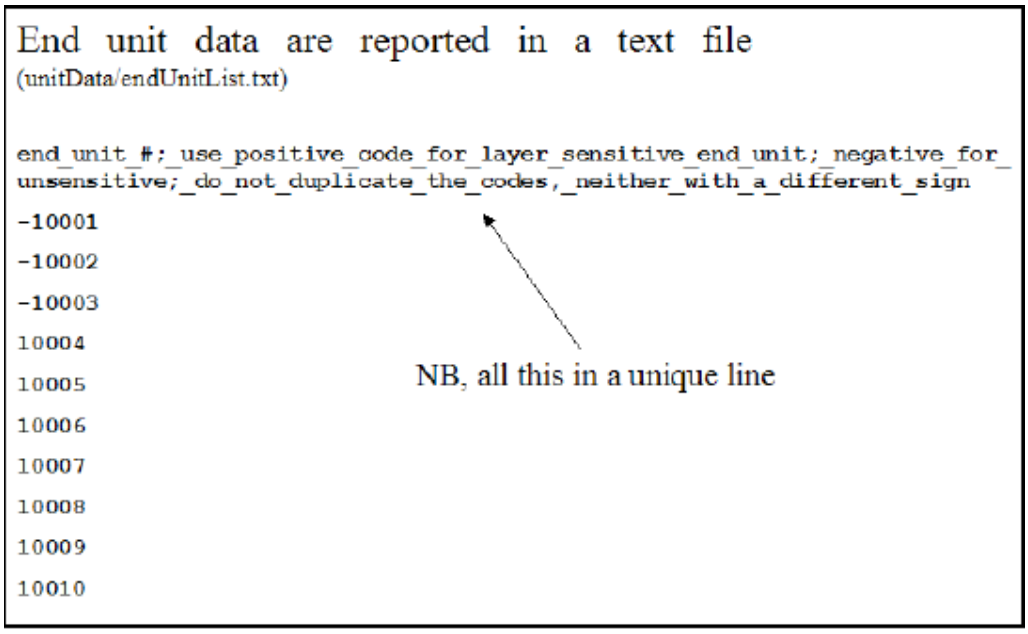


Figura 6- EndUnitList.txt

La prima riga del file è obbligatoria.

Il codice di queste unità può essere di due tipi, un tipo di codice è sensibile al livello[4] ed è identificato con un codice positivo, un altro tipo

[4] Di cui si è parlato in questo capitolo nel paragrafo precedente.

di codice che identifica le *end units* è insensibile ai livelli ed è identificato con un codice negativo.

5.4 Quando fare cosa (WDW)

La simulazione è attivata da degli ordini che contengono delle ricette, nelle quali sono descritti gli step necessari a portare a termine la produzione. Per far iniziare la simulazione, dobbiamo prima di tutto definire i parametri di tempo.

I primi paramenti che vengono stabiliti sono sostanzialmente due:

1. *ticksInTimeUnit*, il quale descrive quanti ticks, nell'orologio della simulazione, sono necessari per concludere un unità di tempo;
2. *timeToFinish* che descrive il numero di ticks dopo i quali il programma stoppa la simulazione.

Gli ordini all'interno della simulazione possono essere generati sia in modo casuale che tramite l'immissione di ricette che partono in un determinato momento. Nel primi caso gli ordini sono generati, all'interno della simulazione, dall'*orderGenerator* nel secondo caso invece si sceglie una precisa sequenza temporale usando l'*oderDistiller*.

È solo con l'uso dell'orderDistiller che noi introduciamo il terzo importante formalismo di jES, ossia il *When Doing What* (WDW).

Questo formalismo è molto utile per creare un repertorio di ricette e per fare partire queste ricette ad un tempo stabilito.

Il repertorio di ricette è contenuto in un file chiamato *recipes.xls* ed il file è contenuto in una directory a cui è stato attribuito il nome *recipeData*. Le ricette devono essere contenute nel primo foglio di lavoro di excel e questo foglio deve avere il nome Recipe.

Il foglio di ricette è così costruito:

#	Recipes	;															
	RecipeA101		1	*s*	1		2	*s*	1		*p*	1	10	3	*s*	1	;
	RecipeB100		1	*s*	1		*e*	10	;								

Tabella 7 - Ricette

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 vengono letti. Ciò che è molto importante è unicamente l'ordine con cui vengono disposti gli elementi all'interno del foglio. Infatti la lettura di questo avviene da sinistra verso destra e dall'alto verso il basso, quindi gli elementi devono essere disposti nello stesso ordine.

All'interno del file delle ricette ci possono essere dei commenti, per indicare 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. Quanto detto ora si può vedere nella *tabella 6* per ciò che riguarda la prima riga.

La prima cosa che il simulatore deve leggere all'interno della ricetta è il nome attribuito alla ricetta stessa ed il numero identificativo associato alla ricetta. Naturalmente il nome della ricetta può essere uguale per ogni ricetta ma il numero identificativo della singola ricetta deve essere univoco. Dopi i primi due importantissimi elementi della ricetta avremo il corpo della ricetta.

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 del file, conclusa con il segno di "***;***" descrive gli eventi di un tick dell'orologio della simulazione. La riga parte con un numero che la identifica e successivamente contiene il numero identificativo della ricetta che momento 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.

Esemplificando avremo:

4			32	*	1	;
5	l	2	33	*	1	;

Tabella 8 - OrderSequences.xls

Come si vede nell'esempio abbiamo la prima ricetta, la numero 32, che parte al tick 4 ed è ripetuta una sola volta. Per eseguire questa ricetta il simulatore non deve utilizzare i livelli.

Nel secondo caso invece abbiamo la ricetta 33 che parte al tick 5, è ripetuta una sola volta e per essere effettuata il simulatore deve utilizzare il secondo 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.

Questi file possono essere utilizzati insieme o uno alla volta, se nella simulazione abbiamo bisogno di utilizzare uno solo di questi file non possiamo eliminare l'altro ma dobbiamo compilarlo riempiendo la prima casella del file con uno zero seguito dal punto e virgola.

5.5 Un semplice esempio

Qui di seguito si riporta un esempio di funzionamento del modello per un'ipotetica impresa simulata molto semplice.

L'esempio aiuterà a compredendere come avviene all'interno di jES la gestione della produzione, ma soprattutto mostrerà quanto può essere utile lo strumento della simulazione utilizzato come macchina per automatizzare gli esperimenti mentali. Da D. Parisi (2000):

> *"[…] le simulazioni possono funzionare anche come macchine per fare esperimenti mentali, cioè per verificare con la simulazione le predizioni generate dallo scienziato con la sua testa. Gli esperimenti mentali non sono un metodo molto usato nella scienza, almeno non sono attività effettuate dagli scienziati in modo cosciente, sistematico, con risultati riportati in modo esplicito e pubblico. La ragione è che le limitazioni della mente umana limitano molto l'utilità di questo metodo di ricerca. […]Le cose stanno diversamente quando entrano in gioco le simulazioni. Le simulazioni possono essere usate come un nuovo metodo, più oggettivo e più sistematico per fare esperimenti mentali ."*

Il semplice esempio che riportiamo è fatto con un'impresa che ha due unità produttive con codice *1,2* e *3* ed una endUnit con codice *10.* il file *unitBasicData.txt* riporta queste informazioni:

Codice unità	***Passo associato***	***Usa magazzino***	***Costi fissi***	***Costi variabili***
1	1	0	10	1
2	2	0	10	1
3	3	0	10	1

Tabella 9- Descrizione unità

Ora descriviamo gli ordini da portare a termine. La prima ricetta va a descrivere un prodotto semilavorato mentre la seconda ricetta descrive un prodotto finito che avrà però bisogno dell'approvvigionamento del primo prodotto. Quindi in questo semplice esempio verranno usati i procurement.

RecipeA101	1 *s* 1	2 *s* 1	*p* 1 10 2 *s* 1	3 *s* 1 ;
RecipeB100	1 *s* 3	\ 3 2 *s* 1	3 *s* 3	\ 2 *e* 10 ;

Tabella 10- Ricette dell'esempio

Nell'esempio avremo che ogni tick della simulazione vale un secondo e che ogni secondo della simulazione è il corrispondente di un ciclo della simulazione stessa.

Nel nostro esempio verranno lanciati 4 ordini in un unico cicl. La sequenza dei lanci è quella riportata nella *tabella 10* che segue:

Ciclo produttivo	***Ordini lanciati***
0	;
1	100 * 3 101 * 1 ;

Tabella 11- Sequenza ordini dell'esempio

Quando il nostro semplice esempio sarà fatto partire avremo che *l'unità 1* incomincia a produrre un passo della ricetta 100, ed essendo chiamato subito un sequential batch avremo che produce tre pazzi della ricetta 100. con il lancio della simulazione si aprono i grafici inerenti all'andamento della produzione (primo grafico) ed al magazzino.

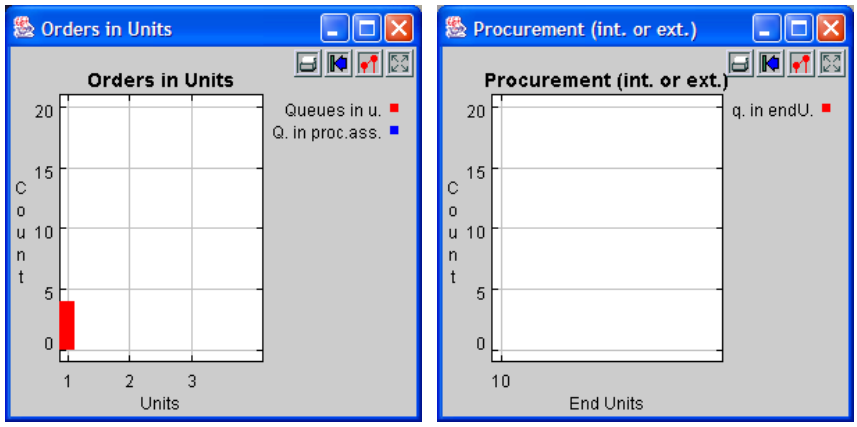


Figura 7- Andamento della produzione e magazzino

Quanto riportato in figura accade al tick 1, se ora facciamo un salto temporale e passiamo al tick 3 avremo che il grafico che riguarda la produzione è cambiato, infatti l'unità uno ha concluso la produzione del primo step e dei tre batch e quindi ha passato la produzione alla seconda unità. Al terzo tick parte anche la *ricetta 101*.

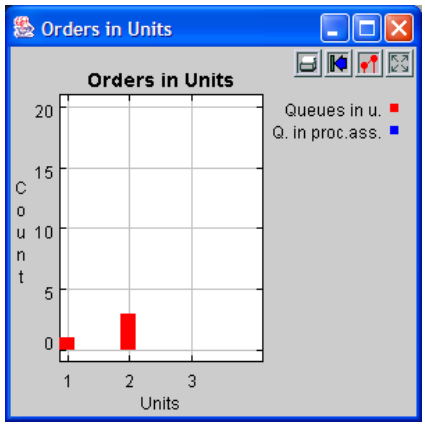


Figura 8- produzione dopo tre tick

È nel quarto tick che entra in gioco l'approvvigionamento da magazzino, infatti la *ricetta 101* ha bisogno di un semilavorato che sta nella endUnit 10 per poter proseguire. Quello che nel nostro esmpio capita è la mancanza del semilavorato, infatti questo semilavorato non è ancora pronto quindi la produzione di questa *ricetta 101* si fermerà fino a che la seconda ricetta (la numero 100) non avrà concluso il suo percorso e non avrà depositato il semilavorato in magazzino.

5.6 Le funzionalità del simulatore

Il simulatore ha al suo interno alcune funzionalità molto particolari che spesso vengono utilizzate all'interno della simulazione anche se non è necessariamente così. Queste funzionalità sono state create e sviluppato mano amano che si presentavano problematiche tipiche della realtà che si stava studiando. Per portare un 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 vengono chiamati *passi computazionali*, i quali erano stati creati ma non ancora utilizzati in alcuna simulazione.

Procediamo con ordine ed approfondiamo tutti gli aspetti cruciali del funzionamento del simulatore. Questi aspetti sono:

- gestione dei magazzini;
- gestione della conoscenza;
- contabilità;
- layer;
- passi computazionali.

5.6.1 Gestione dei magazzini

La gestione dei magazzini è un classico esempio di complessità nell'organizzazione aziendale. Infatti perché i magazzini funzionino ve è bisogno di regole che dicano quando questi devono riempirsi o svuotarsi.

Tali regole di "comportamento" 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 e gli chiedono se vi sono scorte

da produrre e in quale quantità. Il *RuleMaster* permette anche la gestione delle quantità di semilavorati che possono essere detenute nei magazzini, definendone in questo modo la dimensione.

Comunque il funzionamento del magazzino è ben descritto dalla *figura 9*.

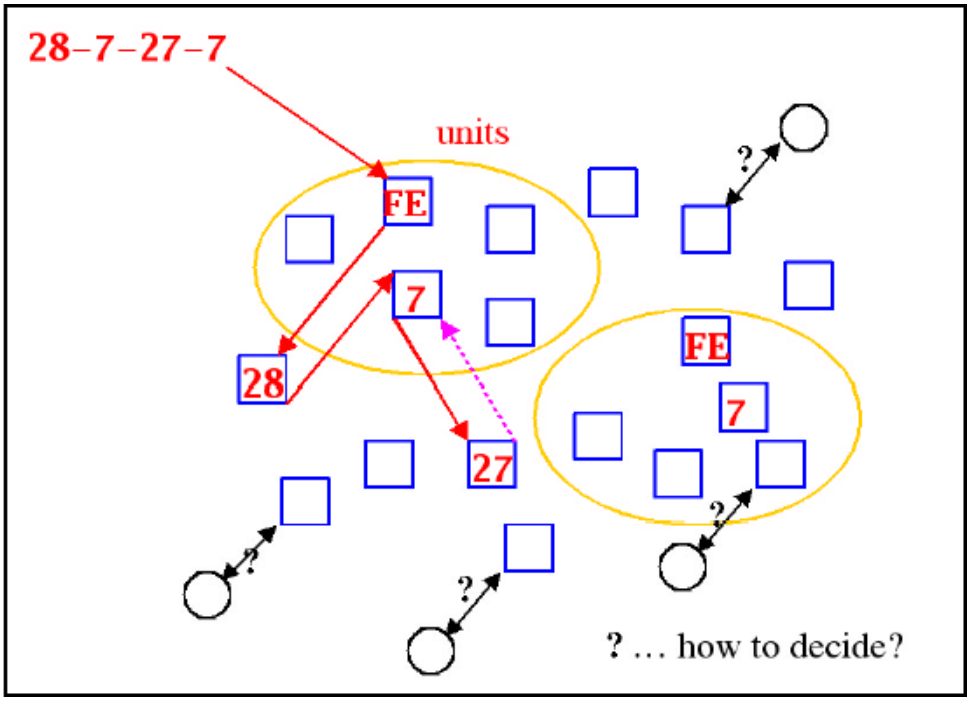


Figura 9 - I magazzini

5.6.2 Gestione della conoscenza

In ogni azienda il personale che vi lavora ha una certa conoscenza dei processi che caratterizzano l'organizzazione di questa impresa. Risulta quindi di notevole interesse simulare e studiare la circolazione delle informazioni all'interno dell'azienda.

Il funzionamento della circolazione di informazioni all'interno dell'azienda simulata è ben rappresentato dalla *figura 10*.

In verità il modello jES prevede una rappresentazione della conoscenza aziendale decentrata, rispettando a pieno le caratteristiche

delle simulazioni ad agenti e facendo emergere molto chiaramente la complessità tipica di queste situazioni. Purtroppo questa rappresentazione della circolazione della conoscenza non rispecchia le tipiche realtà aziendali dove la conoscenza è detenuta interamente da pochi soggetti.

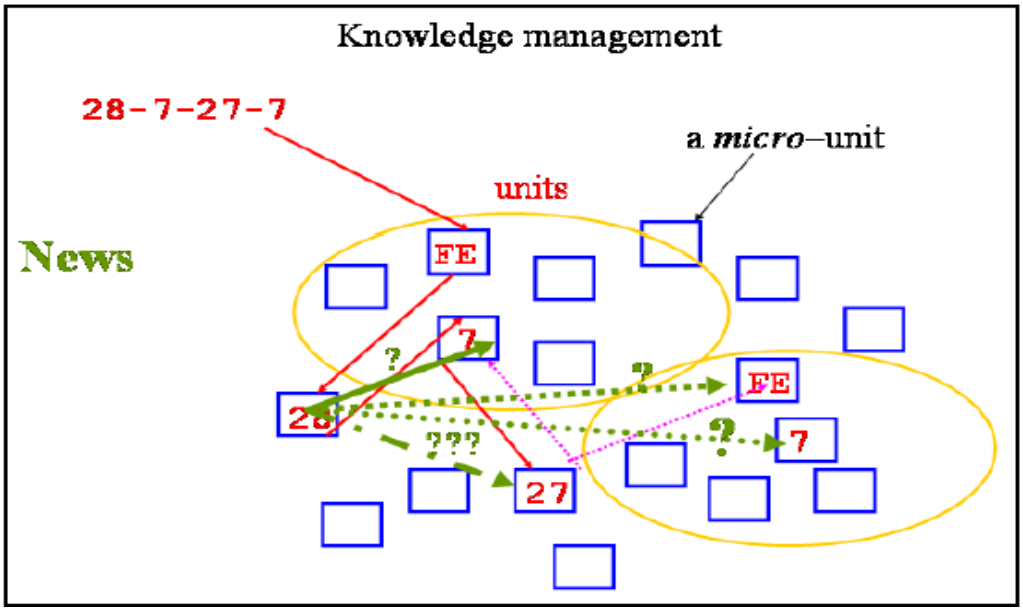


Figura 10- L'informazione

Nel modello jES le informazioni sono racchiuse all'interno di appositi oggetti informatici definiti News. Le unità produttive ad ogni ciclo decidono se inviare o meno informazioni ad altre unità produttive.

Attualmente, il programma 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 destinazione di risorse scarse condivise tra più unità produttive.

Esattamente come nella gestione dei magazzini, le News richiedono regole e contenuti tramite un RuleMaster ed un RuleMaker.

5.6.3 Contabilità

jES prevede all'interno del modello due punti di vista differenti per fare contabilità, ossia la contabilità può essere fatta per unità produttive e per prodotto. In ogni caso questa viene aggiornata in tempo reale durante il processo simulativo.

In *figura 11* si vede meglio come funziona questa caratteristica del simulatore d'impresa.

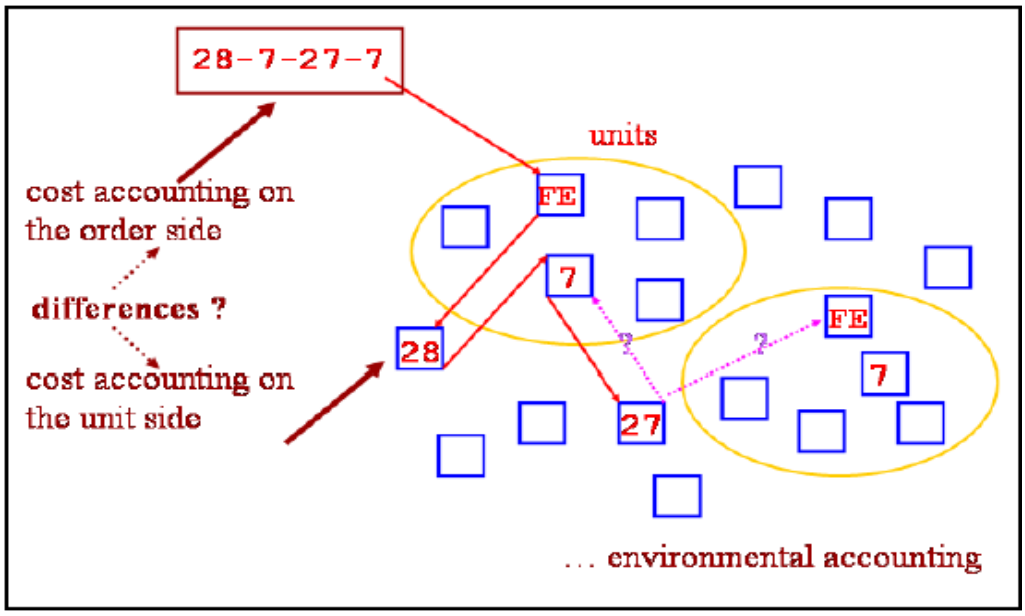


Figura 11- La contabilità

Il primo tipo di contabilità, e quindi la contabilità per unità produttiva, è caratterizzata dai costi fissi e variabili che, alla conclusione di ogni ciclo l'impresa simulata registra.

Nella seconda visione, il prodotto in ogni fase di lavorazione 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.

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

Bisogna notare che anche le unità che producono per i magazzini registrano costi fissi e variabili ed in più i materiali che giacciono in magazzino verranno caricati di un costo finanziario pari al tesso di interesse stabilitp dall'utilizzatore del simulatore.

5.6.4 Layer

Questo strumento della simulazione nasce dall'osservazione che, ad ordini uguali, possono essere associati beni che per le proprie caratteristiche fisiche sono identici ma che hanno caratteristiche qualitative differenti. Si pensi ad esempio al modello di simulazione sviluppato per la BasicNet[5], nella quale era necessario sviluppare collezioni diverse che dal punto di vista strettamente produttivo non creavano alcun tipo di problema ma i prodotti, sebbene identici, dovevano essere trattati in modo differente perché facevano parte di collezioni d'abbigliamento diverse.

Quindi all'interno della simulazione potremmo avere che ordini con ricette uguali potrebbero avere caratteristiche qualitative o quantitative differenti che, in qualche modo, devono essere rilevate durante la simulazione.

I layer consentono di gestire questa differenziazione.

L'assegnazione del layer avviene in fase di lancio degli ordini da parte dell'*orderDistiller*. Questa diversificazione degli ordini potrà avere o no, in base alle decisioni dell'utente, effetto 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.

[5] Ditta che produce abbigliamento

5.6.5 I passi computazionali

jES ha capacità computazionali che possono essere associate ad ogni passo della ricetta. In tabella si vede bene qual è la notazione necessaria perché si possa associare al passo della ricetta un passo computazionale:

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

Tabella 12- Passi computazionali

La fase di produzione 1 viene eseguita per un tic, dopodiché la fase di produzione 2 richiama il passo computazionale numero 1999, il richiamo del passo è 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.

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 *mamoryMatrixes.txt* ed il quale contiene alcune informazioni che sono:

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

Il numero che indica il nome della matrice potrà essere anche negativo, indicando così l'insensibilità di quella matrice ai livelli che vengono cerati automaticamente. Nel nostro semplice esempio il file delle *memoryMatrixes* contiene le seguenti informazioni:

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

Tabella 13- MemoryMatrixes

Come si vede molto chiaramente la seconda matrice del nostro

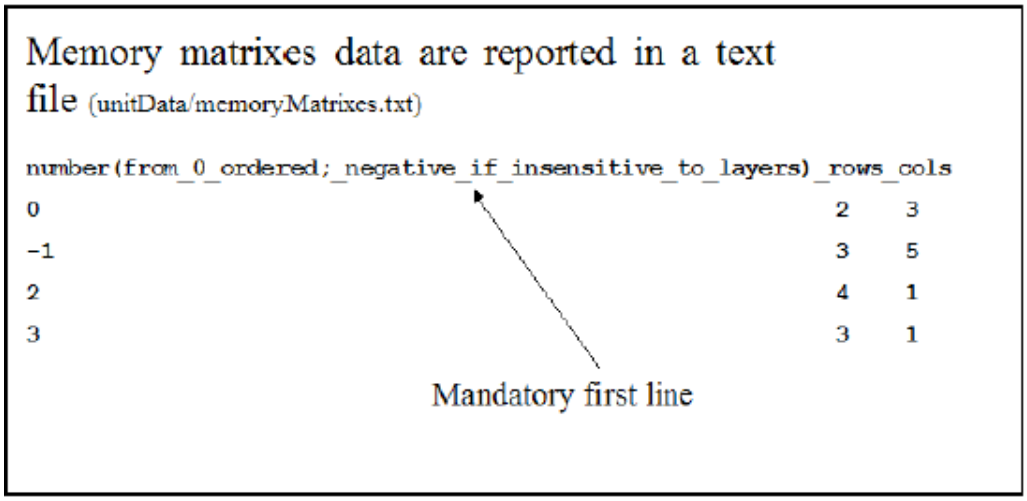


Figura 12- memory Matrixes.txt

esempio è insensibile ai livelli. Il file di *mamoryMatrixes* ha questo aspetto:

Quindi il passo computazionale viene utilizzato per attivare una sottoricetta che inizia e prosegue fino a conclusione indipendentemente dalla ricetta che l'ha lanciato.

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