Introduzione a jES

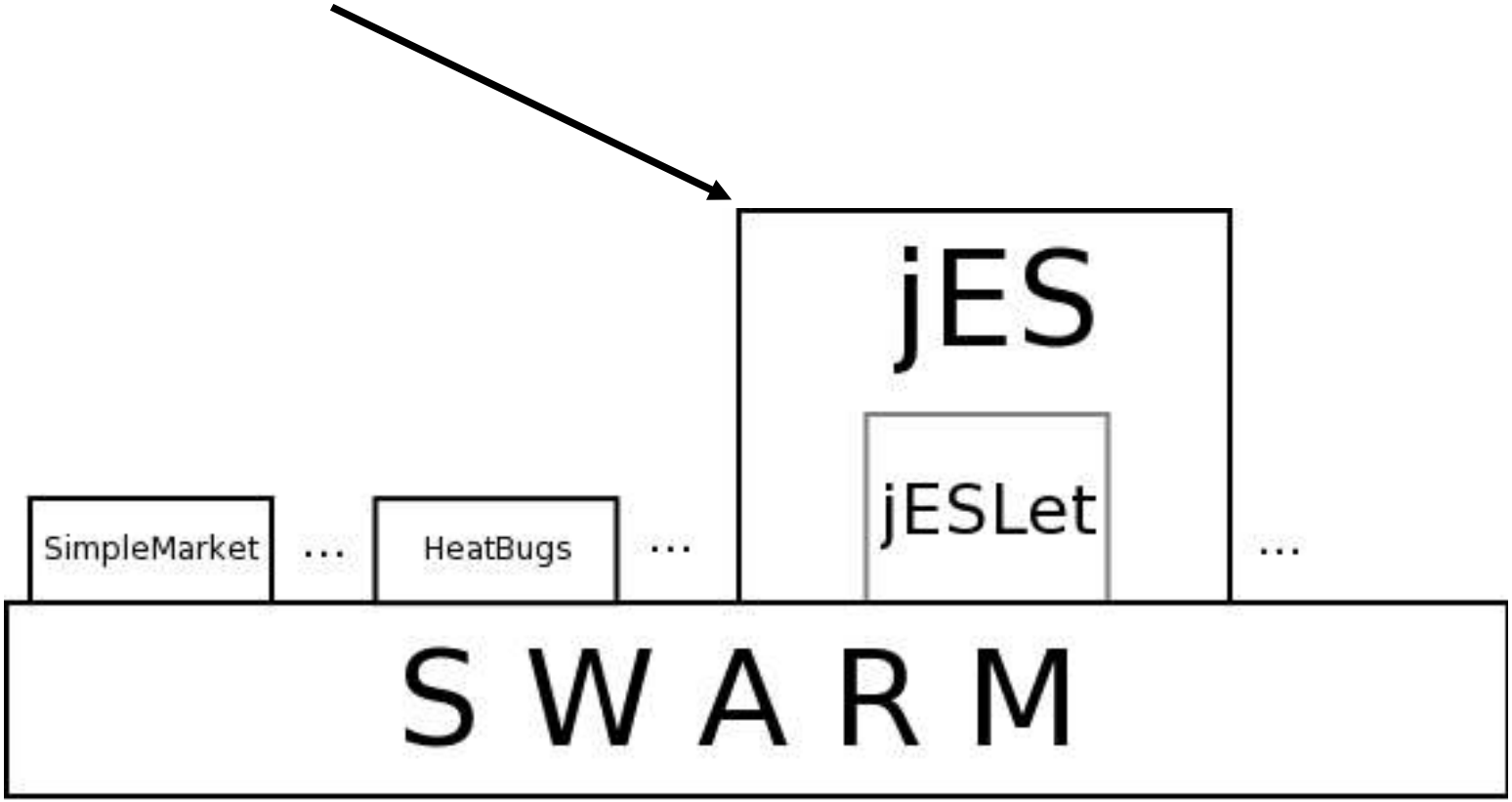
Università degli Studi di Bologna
Facoltà di Scienze MM. FF. NN.
Corso di Laurea in Scienze di Internet
Anno Accademico 2004-2005

Laboratorio di Sistemi e Processi Organizzativi

jES

- jES (Java Enterprise Simulator) è una piattaforma per sviluppare modelli di simulazione di impresa
- È basato sulla versione java di Swarm (http://www.swarm.org)
  - Per simulare le attività di un'impresa reale e analizzarne quindi i risultati
  - Per costruire imprese virtuali ed ipotetiche
- Nel primo caso si può usare il simulatore per testare il comportamento di una impresa simulata relativamente alle modifiche che in essa possono essere apportate
- Nel secondo ci si interessa nell'analisi teorica della creazione, del comportamento e delle interazioni delle imprese.

jES



jES

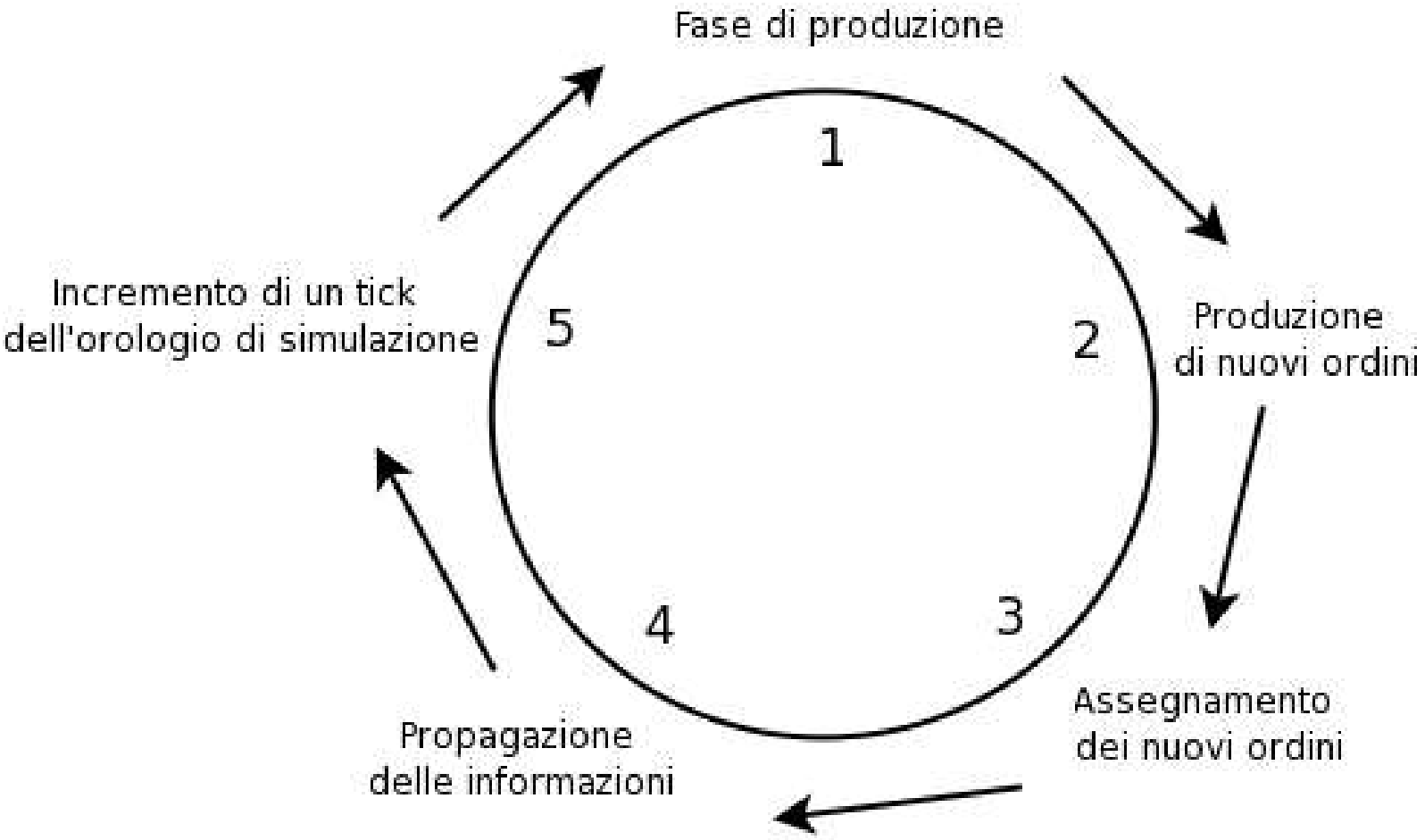
- L'impresa simulata
  - Realizza ordini
  - Ha unità produttive che eseguono i diversi passi del processo di produzione.
- Il primo aspetto rappresenta il Cosa fare (**WD** *What to Do*),
  - in esso gli ordini vengono rappresentati come ricette;
- Il secondo rappresenta il lato Cosa fa Cosa (**DW** w*hich is Doing What)*
- Un Terzo formalismo è quello che rappresenta la sequenza di eventi ed il loro ordine di esecuzione:
  - rappresenta il quando fare cosa (**WDW** *When Doing What).*

jES: comportamento del simulatore

- Possiamo descrivere il comportamento del simulatore supponendo che il processo di simulazione sia già incominciato e sia quindi in grado di elaborare ordini:

1. Fase di produzione,
2. Produzione di nuovi ordini,
3. Assegnamento dei nuovi ordini,
4. Propagazione delle informazioni,
5. Incremento del tick dell'orologio di simulazione.

jES: comportamento del simulatore



1. Fase di produzione (1/4)

- le unità di produzione operano sugli ordini presenti nelle proprie liste di attesa, se ce ne sono;
- Ogni ordine viene elaborato per ogni tick dell'orologio di simulazione.
  - se l'unità è inattiva (idle), non avendo alcun ordine nella sua waiting list, può produrre scorte in modo stand alone
    - tali scorte stand alone sono prodotte solo se il parametro useWarehouses è impostato a true;
  - Una volta processati gli ordini vengono messi in una lista di produzione eseguita per essere quindi diffusi ad altre unità di produzione;

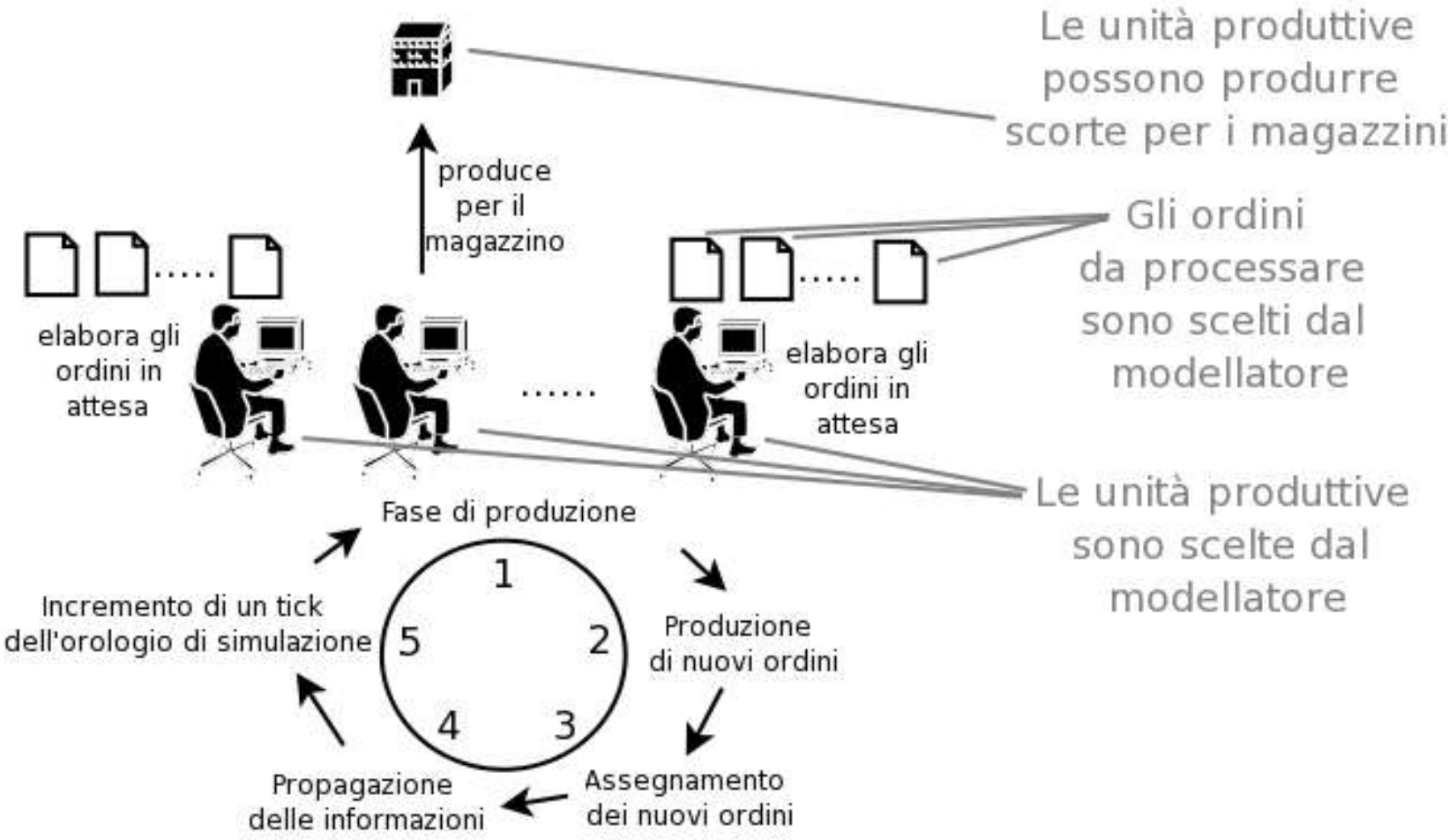
1. Fase di produzione (2/4)

- una ricetta d'ordine può includere il codice identificativo di una **endUnit**:
  - in questo caso l'ordine si riferisce ad un componente prodotto esternamente o internamente: dopo la sua produzione (o il suo approvigionamento) questo tipo di ordine è tenuto in un magazzino reale o virtuale rappresentato da una *endUnit*;
  - è bene osservare la differenza tra questo tipo di produzione e quella stand alone per le scorte di ogni specifica unità produttiva;

1. Fase di produzione (3/4)

- se un'operazione richiede più di un tick dell'orologio per essere conclusa, l'ordine è mantenuto nell'unità produttiva fino a quando tutto il tempo è trascorso, con un riassegnamento diretto ed immediato dell'ordine all'unità stessa;
- la produzione può essere rimpiazzata dall'uso di scorte relative ad ogni specifico passo di produzione:
  - ovviamente, se tali scorte stand alone esistono;
  - in questo caso, più di un ordine può essere completato in un singolo tick, se vi sono le necessarie scorte;
- la produzione può richiedere un tempo per l'organizzazione, con relativi costi e tempo speso (non ancora implementato).

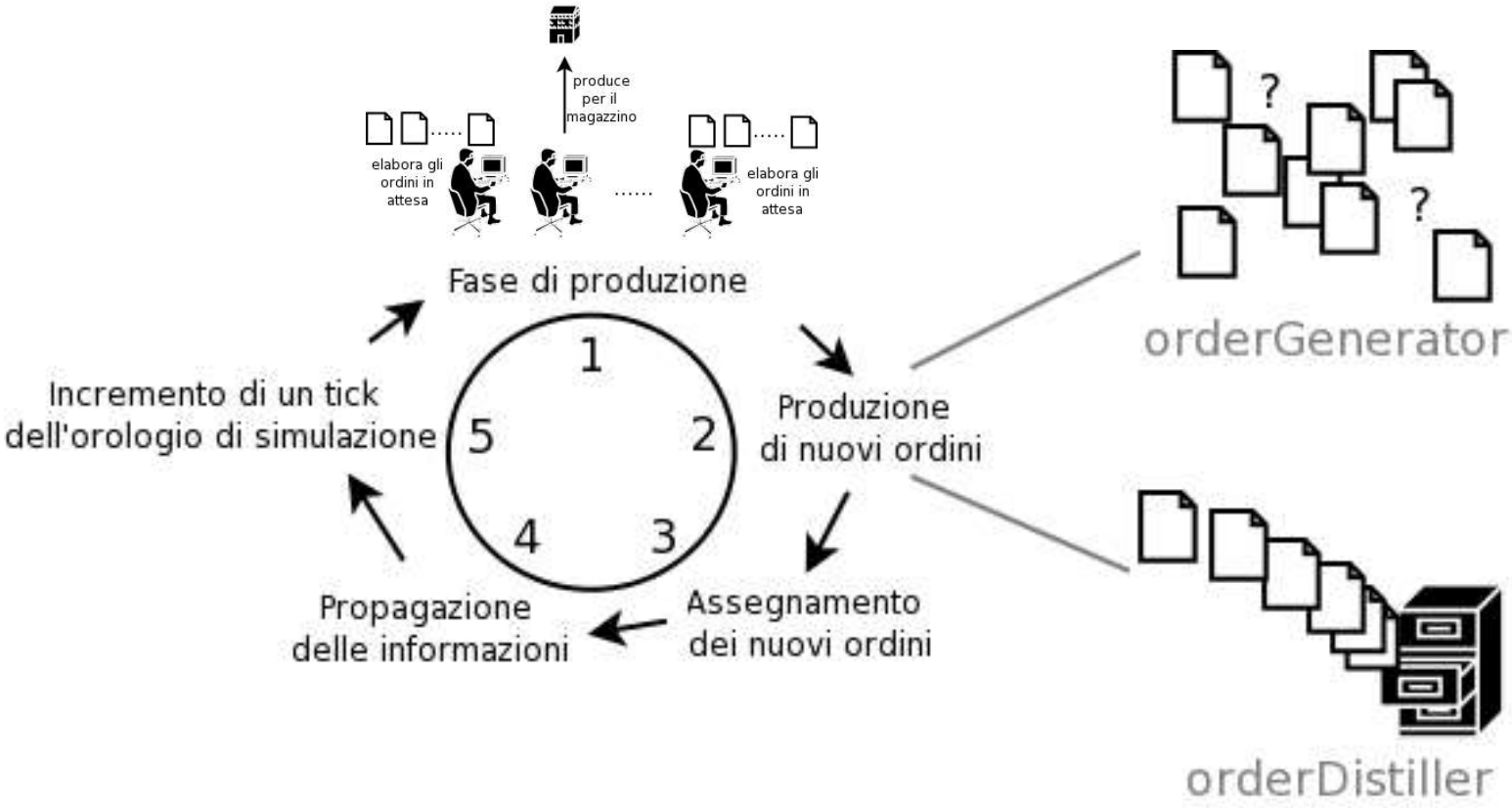
1. Fase di produzione (4/4)



2. produzione di nuovi ordini (1/2)

- i nuovi ordini sono messi in produzione:
  - seguendo uno script che descrive la sequenza degli eventi che devono essere simulati (usando l'oggetto ***orderDistiller***);
  - generati casualmente dall'oggetto ***orderGenerator***;
- ogni ordine contiene la ricetta contenente i passi che devono essere eseguiti;
- i nuovi ordini sono assegnati alle *unità di produzione* come mostrato al punto 3;

2. produzione di nuovi ordini (2/2)



3. Assegnamento dei nuovi ordini (1/4)

- ogni ordine contenuto nella ***produzione eseguita*** (*made production list*) delle *unità di produzione* o appena creato (come nel punto 2) compie una ricerca
  - sfruttando l'oggetto chiamato ***assigning tool***,
  - per scoprire se una o più *unità di produzione* è in grado di eseguire i passi rimanenti affinché la ricetta (e quindi l'ordine) sia completata.

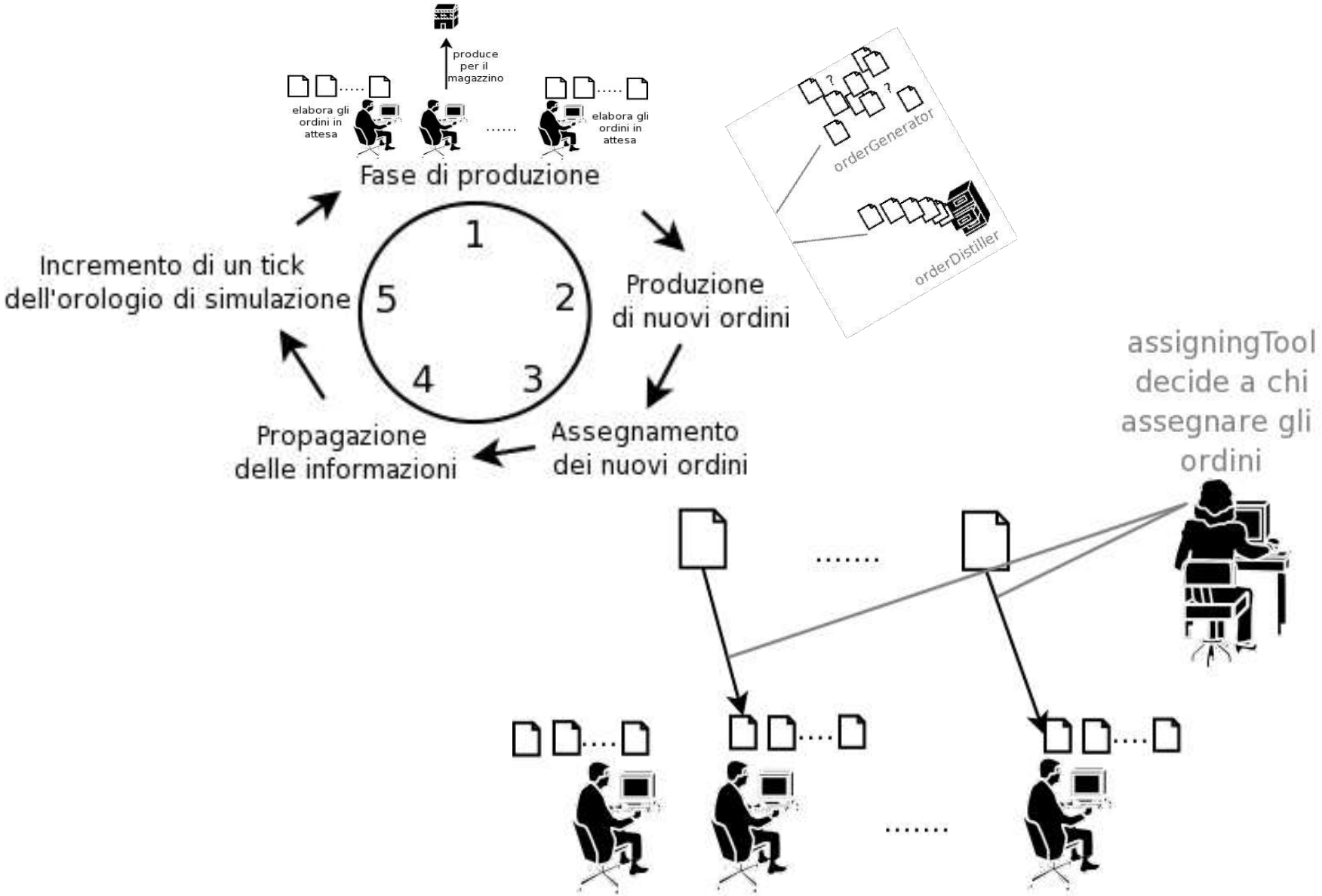
3. Assegnamento dei nuovi ordini (2/4)

- **assegnamenti**:
  - se solo una *unità di produzione* dà una risposta affermativa, allora l'ordine viene assegnato alla coda d'attesa di quella *unità di produzione*;
  - se più di una *unità di produzione* è in grado di eseguire il passo richiesto, si rende necessario scegliere una delle unità candidate:
    - la scelta può essere presa seguendo diversi criteri, come mostrato in ***unitCriterion***;
    - in futuro questo sarà un punto fondamentale di ***jES*** (non ancora implementato) permettendo l'intervento umano nella simulazione.

3. Assegnamento dei nuovi ordini (3/4)

- una volta che gli ordini sono assegnati alla coda d'attesa delle unità di produzione scelte, essi vi rimangono secondo il criterio FIFO, fino a quando i loro specifici passi di produzione sono compiuti;
  - gli ordini possono essere eseguiti secondo una politica LIFO in specifiche circostanze tecniche;
  - come visto ci può essere il caso in cui un passo per essere eseguito richieda più di un tick nella simulazione;
  - la sequenza degli ordini nella waiting list può essere gestita in modo da migliorare la performance dell'impresa (non ancora implementato);
- un ordine viene scartato quando l'ultimo suo passo viene compiuto.

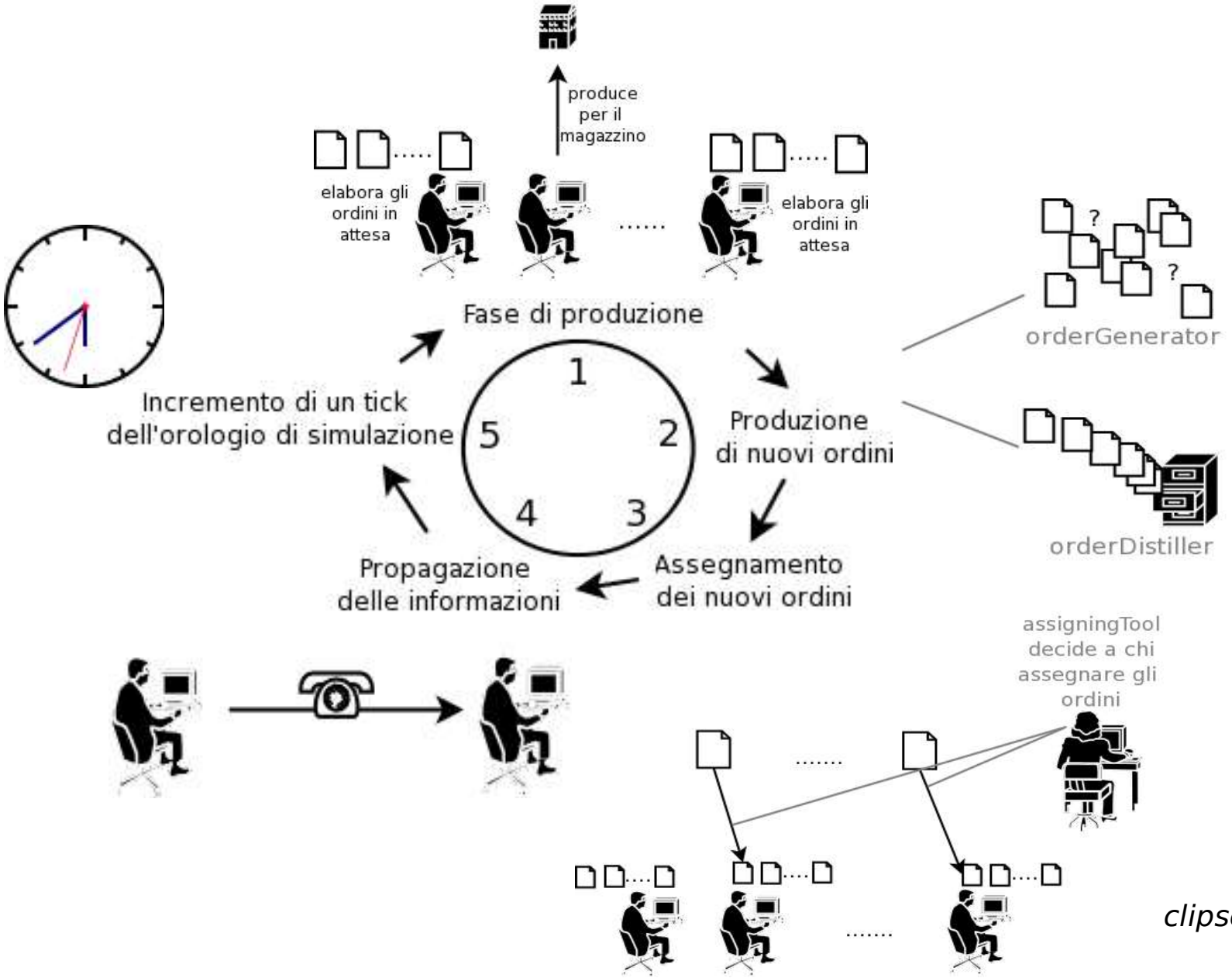
3. Assegnamento dei nuovi ordini (4/4)



Punti 4 e 5

- Punto 4 (***Propagazione delle informazioni***):
  - se il parametro ***useNewses*** è impostato a *true*, ogni *unità di produzione* propaga informazioni sugli ordini che spedirà ad altre *unità* in futuro;
- Punto 5 (***Incremento del tick dell'orologio di simulazione***):
  - nel successivo *tick* dell'orologio di simulazione la sequenza riprende dal punto 1.

Riassumendo



Come funziona il modello?

- Ad ogni tick dell'orologio di simulazione, tutte le unità di produzione compiono le azioni descritte al punto 1 indipendentemente (la reale sequenza non conta);
- Quindi, sempre nello stesso tick di simulazione, viene eseguito il punto 2;
- Quando tutte le operazioni al punto 2 sono terminate, gli ordini compiono quanto descritto al punto 3, in modo indipendente e sempre nello stesso tick di simulazione;
- Infine le unità di produzione eseguono il punto 4.

Che cosa fare (WD)

- Il **che cosa fare** (**WD**) viene rappresentato con il formalismo delle **ricette**.
- Le **ricette** sono composte da una successione di passi produttivi che rappresentano le *lavorazioni*.
- Poiché le tipiche attività lavorative di un'impresa non sono solo sequenziali, sono stati introdotti nel formalismo passi speciali che permettono la descrizione di
  - ramificazioni delle ricette,
  - atti di approvvigionamento,
  - lavorazioni per lotti.

Le ricette (1/3)

- Durante la descrizione WD si rappresentano le ricette attraverso un semplice formalismo numerico.
- Ogni passo della ricetta è espresso con una cifra col fine di rappresentare in modo univoco l'atto produttivo o organizzativo che deve essere compiuto.
- Ad ogni passo produttivo si affianca il tempo necessario per il suo compimento espresso con una quantità ed una unità di tempo (giorni, minuti, secondi, ...).

Le ricette (2/3): un esempio

- Esempio di *ricetta*:

{7; s; 20; 8; m; 1}

- il passo 7 richiede 20 secondi per essere completato ed il passo 8 un minuto.
- Terminata la *ricetta* l'ordine è stato completato ed è pronto ad uscire dall'impresa (ad esempio per essere venduto).

Le ricette (3/3)

- Il modo in cui le ricette degli ordini sono scritte è triplice:
  - Esterno,
  - Intermedio,
  - Interno.
- L'utente scriverà sempre le ricette in formato esterno e può creare un repertorio di ricette nel file *recipeData\recipe.xls* posto nella cartella principale di ***jES***.

Prodotto non finito

- Un ordine può anche rappresentare un ***prodotto non finito*** (un ***semi-lavorato***) utile per portare a termine altri ordini.
- Un prodotto non finito deve essere depositato in una unità speciale detta **endUnit**
- Ad esempio

{3; s; 2; 2; s; 10; e; 100}

- In questo esempio l'ordine andrà depositato in una *unità magazzino* (indicata con la lettera e) di codice 100.

I Lotti

- Ci sono situazioni aziendali in cui non risulta realistico descrivere dei processi produttivi prendendo in considerazione singoli oggetti perché i tempi unitari risulterebbero troppo piccoli ed ininfluenti all'interno della simulazione.
- Diventa naturale in questi casi ragionare per **lotti di prodotti**,
- Ad esempio, durante un'operazone di etichettatura di articoli o per una fornitura di bulloni.
- In **jES** sono state inserite due tipologie di passi definite batch:
  - ***Sequential batch***
  - ***Stand alone batch***

Sequential Batch

- Il **sequential batch** (espresso con il simbolo \, ad esempio {*n; ts; m; \; b*}) in una ricetta ripete per *m* unità temporali la produzione dello stesso passo su *b* ordini.
- Il tempo necessario per la produzione di ciascun ordine è misurato da *m/b* .
- Ad esempio in

  {*n1; ts; m1; n2; ts; m2; n3; ts; m3; \; b3; n4; ts; m4* }
- il passo n3 deve essere compiuto contemporaneamente su b3 ordini e il tempo necessario per ciascun ordine sarà pari a m3/b3

Sequential Batch

{*n; ts; m; \; b*}

- L'unità produttiva che compie il passo necessario al lotto (batch) deve aspettare di avere b ordini nella sua waiting list per avviare il batch (quindi l'effettiva produzione del lotto);
- mentre è in attesa può processare altri lotti o altri singoli ordini.
- Quindi un sequential batch deve essere associato ad un passo produttivo.

Sequential Batch: esempio

- Ad esempio

{7; s; 20; 8; m; 1; \; 100}

- in questo modo si indica che il passo 8 deve essere compiuto da un'unica unità produttiva contemporaneamente su 100 ordini.
- Dopo la lavorazione sequential batch verranno nuovamente gestiti in modo separato i diversi singoli ordini.
- L'esempio riportato potrebbe essere utilizzato per descrivere un'operazione di tipo sequential batch per simulare l'etichettatura di 100 prodotti che avviene in un minuto.

Stand alone batch

- Un **stand alone batch** (espresso con il simbolo /) può essere inserito soltanto in ricette composte da un unico passo ed un'unità di destinazione (endUnit).

- Questo formalismo è tipicamente utilizzato per rappresentare
  - una fornitura di materie prime
  - semilavorati necessari per completare altri ordini.

Stand alone batch: esempio

- Dato l'esempio

{7; d; 1; /; 2000; e; 10}

- ipotizzando una fornitura di bulloni, compiuto il passo 7 di durata un giorno all'interno della ***endUnit*** 10 saranno presenti 2000 bulloni utili al completamento di ordini futuri (valvole, motori, . . .).

Procurement

- Affinché un componente possa essere recuperato da una ***endUnit***, è previsto un passo di tipo **procurement** (espresso con il simbolo **p**);
- vediamo di seguito che il passo 4 per essere portato a termine necessita la fornitura di due *prodotti non finiti* con codice 10 e 20 (recuperabili rispettivamente nelle ***endUnit*** 10 e 20):

{3; s; 5; p; 2; 10; 20; 4; s; 5}

Esempi (1/2)

- Si consideri l'esempio seguente:

{10; s; 3; 11; s; 300; \; 100; e; 121}

- Il componente 121 è prodotto probabilmente internamente poiché ne viene descritto in dettaglio la produzione (un passo di tipo 10 della durata di 3 secondi, seguito da un lotto sequenziale di tipo 11 eseguito su 100 componenti contemporaneamente in 300 secondi);

Esempi (2/2)

{20; s; 2000; /; 1000; e; 34}

- il componente 34 è probabilmente prodotto esternamente essendo descritto come fosse una scatola nera con un lotto stand alone che produce 1000 oggetti di tipo 20 in 2000 secondi;

{p; 3; 31; 32; 33; 44; h; 2; e; 73}

- il componente 73 vede come unico passo il passo 44 che impiega 2 ore richiedendo la fornitura dei componenti 31, 32 e 33.

Passi OR

- È tipico di molte aziende la possibilità di realizzare un prodotto seguendo “strade” alternative;
- Ad esempio:
  - Il taglio di un materiale potrebbe, ad esempio, essere effettuato utilizzando una lama elettrica o un dispositivo laser.
  - I due prodotti finiti risulterebbero uguali, ma i passi compiuti per realizzarli sono stati differenti.
- Un passo di tipo or (espresso con il simbolo ||) permette di ramificare la ricetta produttiva su due o più rami alternativi.

Passi OR: esempio

{10; m; 1; ||; 1; 100; s; 30; ||; 2; 200; s; 45; ||; 0}

- Nell'esempio riportato sopra vediamo che l'ordine può essere completato in due modi alternativi: o eseguendo il passo 100 o il passo 200.
- Si noti che dopo il simbolo || è indicato un numero identificativo del ramo **or** in considerazione,
- l'intera sequenza di rami è conclusa da ||; 0.

Chi fa che cosa?

- L'aspetto **DW** (*Which is doing what*) è legato alla descrizione delle risorse produttive disponibili all'interno dell'impresa, o presenti esternamente ma in qualche modo legate alla sua realtà.
  - Unità
  - Unità complesse
  - EndUnit
  - Magazzini

Le unità

- 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.

Le unità semplici

- Tutte le unità semplici, che sono cioè in grado di gestire un unico passo di produzione, vengono descritte all'interno del file unitData\unitBasicData.txt
- la descrizione comprende le seguente informazioni:
  - Codice unità: numero univoco di ogni unità nella simulazione;
  - Passo associato: si indica quale passo produttivo è svolto dalll'unità;
  - Utilizzo dei magazzini: si indica se l'unità presenta un magazzino ad essa associato (con un 1 o uno 0);
  - Costi fissi: si indicano i costi fissi che l'impresa deve registrare ad ogni ciclo della simulazione;
  - Costi variabili: si indicano i costi variabili associati all'attività produttiva.

orderGenerator e orderDistiller

- L'oggetto **orderGenerator** è usato in fase di test del codice
  - Esso genera tutte le ricette internamente usando lo stesso intervallo base di tempo
- L'oggetto **orderDistiller** è usato per seguire una sequenza di ordini data
  - Si gestiscono ricette che usano intervalli di tempo eterogenei,
  - **orderDistiller** deve convertire internamente tutte le misure temporali alla più piccola (non ancora implementato)
- **orderDistiller** sfrutta la tabella definita nel file

unitData\unitBasicData.txt

Criteri di scelta

- Ogni ordine compie una ricerca per scoprire se una o più unità di produzione è in grado di eseguire il suo primo passo non compiuto;
- se più di un'unità è in grado di eseguire tale passo si rende necessaria la scelta di un'unità fra quelle valide.
- Sono definiti tre criteri di scelta impostati con il parametro **unitCriterion**:
  - 1. viene scelta la prima unità valida elencata nel file *unitData\unitBasicData.txt*;
  - 2. viene scelta casualmente una fra le unità valide;
  - 3. viene scelta l'unità valida con la minore coda d'attesa

Unità complesse

- È possibile inserire nel modello unità complesse in grado di svolgere più passi produttivi.
- Ogni unità registra costi fissi e variabili differenti in funzione dell'attività che sta svolgendo,
- analogamente anche l'utilizzo del magazzino potrebbe risultare differente.
- Per descrivere le unità complesse si associa al precedente formalismo una matrice che contiene i singoli passi, i costi fissi, i costi variabili e l'utilizzo o meno del magazzino.

endUnit

- Come abbiamo visto in precedenza, i componenti, per poter essere soggetti a procurement, al termine delle loro lavorazioni vengono depositati in unità magazzino dette ***endUnit***.
- Tale ***endUnit***
  - non è un nodo di produzione nell'impresa simulata,
  - ma un nodo che rappresenta una scorta reale o virtuale (quando stiamo producendo o ci stiamo procurando cose immateriali, come i servizi), in cui riponiamo, realmente o metaforicamente, il risultato della ricetta.

Capacita computazionali

- Il modello prevede l'inserimento di oggetti informatici in grado di eseguire algortimi informatici specificati dall'utente.
- I risultati delle computazioni sono archiviate, durante lo scorrere del tempo (simulato), in matrici di memoria predisposte dal programma.
- Questi oggetti con capacità computazionali, con le relative matrici di memoria, possono essere richiamati dalle ricette produttive utilizzando il simbolo c

Capacita computazionali: esempio

{6; s; 12; c; 1998; 2; 0; 1; 8; m; 2}

- Nell'esempio riportato al passo 8 è associata una computazione (simbolo c) con codice 1998 che richiede l'utilizzo delle due matrici 0 e 1.

Capacita computazionali

- lo scopo delle **capacità computazionali** è consentire all'impresa simulata di
  - eseguire operazioni immateriali,
  - fare previsioni,
  - gestire aste,
  - Indirizzare procurement,
  - archiviare informazioni in database.

News (1/3)

- Ogni azienda è caratterizzata da un certo grado di conoscenza che i singoli soggetti che la compongono hanno rispetto all'organizzazazione interna.
- I processi produttivi non sono, realisticamente, sempre svolti in modo decentrato come prevede la versione “pura” del modello.
- Risulta di notevole interesse quindi studiare e simulare la **circolazione di informazioni** che avviene all'interno dell'azienda simulata.

News (2/3)

- 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.
- La news risulta, quindi, una metafora
  - del messaggio,
  - Dell'e-mail,
  - della lettera,
  - del dispaccio aziendale,
  - della cartella consegnata ad un dato settore.

News (3/3)

- Attualmente, il programma **jES** permette
  - di simulare una circolazione di informazioni molto semplificata,
  - caratterizzata da una serie di messaggi che le diverse unità, logicamente connesse, si scambiano per facilitare le dicisioni relative alla produzione di scorte
- L'intero processo di propagazione delle informazioni è attivo se il parametro ***useNewses*** è impostato a *true*

Layer (1/2)

- Le ricette produttive caratterizzano i diversi prodotti che l'impresa simulata è in grado di realizzare.
- Ogni ordine è associato ad una ricetta, ma ordini con ricette uguali potrebbero avere caratteristiche qualitative o quantitative differenti che, in qualche modo, devono essere rilevate durante la simulazione.
- Ad esempio: la ricetta produttiva per la realizzazione di penne a sfera è la stessa, ma potremmo avere la necessità di distinguere le penne per il mercato asiatico da quelle per il mercato europeo.

Layer (2/2)

- Nel modello si è aggiunta la possibilità di associare ad ogni ordine un **layer**.
- I **layer** consentono di gestire in modo distinto prodotti uguali, ma riferiti a periodi di produzione o, più in generale, ad ambiti differenti;
- L'assegnazione del **layer** avviene in fase di lancio degli ordini
- Questa diversificazione degli ordini avrà effetto sulle operazioni interne alla simulazione, come *sequential batch, stand alone batch, procurement* e *capacità computazionali*.

Ancora su orderGenerator e orderDistiller

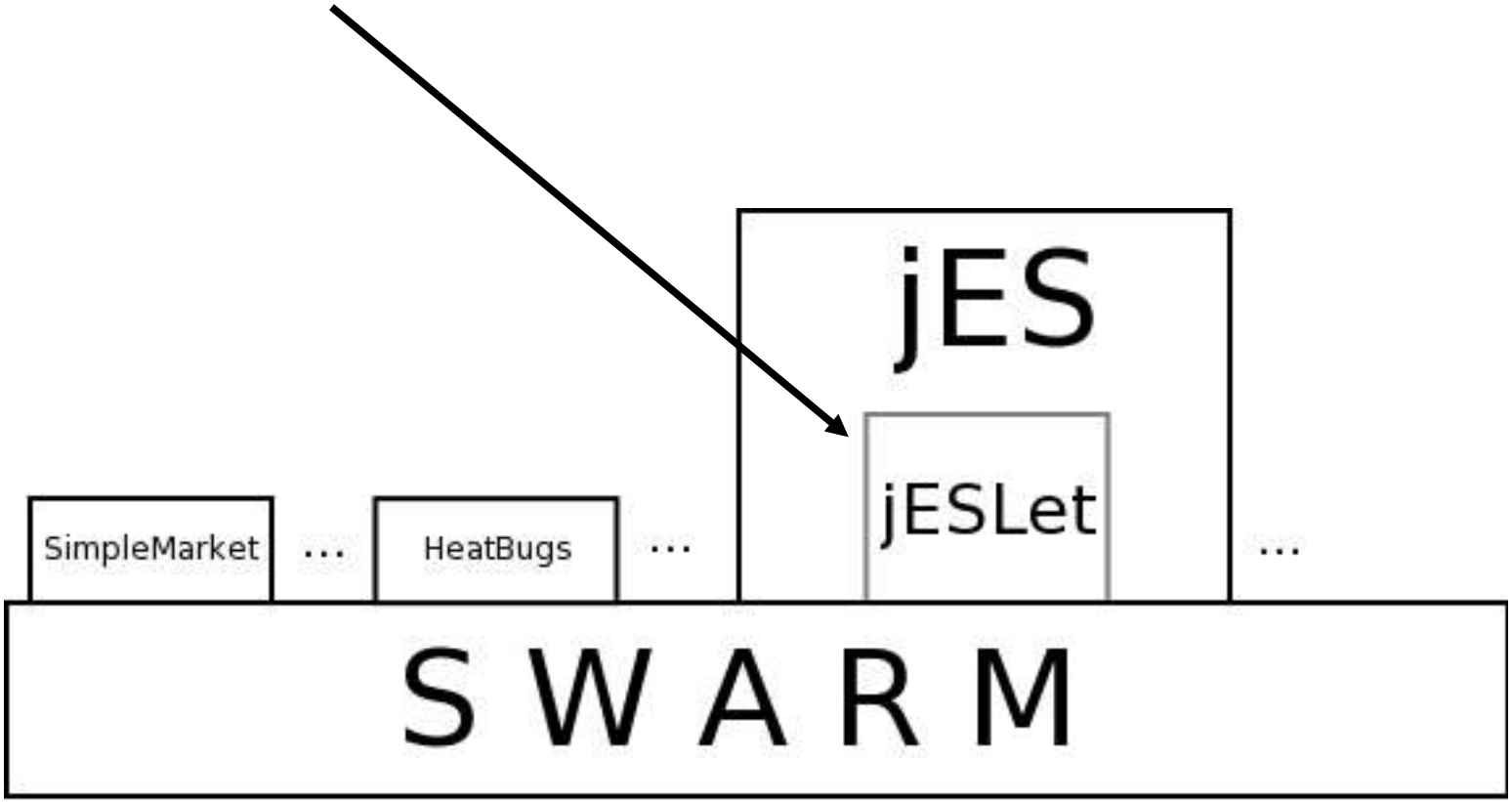
- Gli ordini sono
  - generati casualmente dall'istanza orderGenerator della classe OrderGenerator,
  - oppure ricavati da un repertorio di ricette, seguendo uno schedule temporale, dall'istanza orderDistiller della classe OrderDistiller
    - Questo è il caso comune di applicazione di jES per le situazioni reali
    - In questo caso, introduciamo il formalismo ***quando fare cosa***, *When Doing What* (**WDW**)

When Doing What

- Il formalismo **WDW** è un insieme di convenzioni utile per creare un repertorio di ricette e programmarle nel tempo.
- Questo repertorio di ricette è contenuto nel file *recipeData\recipes.xls;*
- Lo schedule (programma temporale) dell'esecuzione degli ordini è contenuto nel primo foglio di lavoro del file *recipeData\orderSequence.xls*
- Ogni blocco del file (concluso da un “;”) descrive gli eventi che accadono in un tick della simulazione;

jESLet

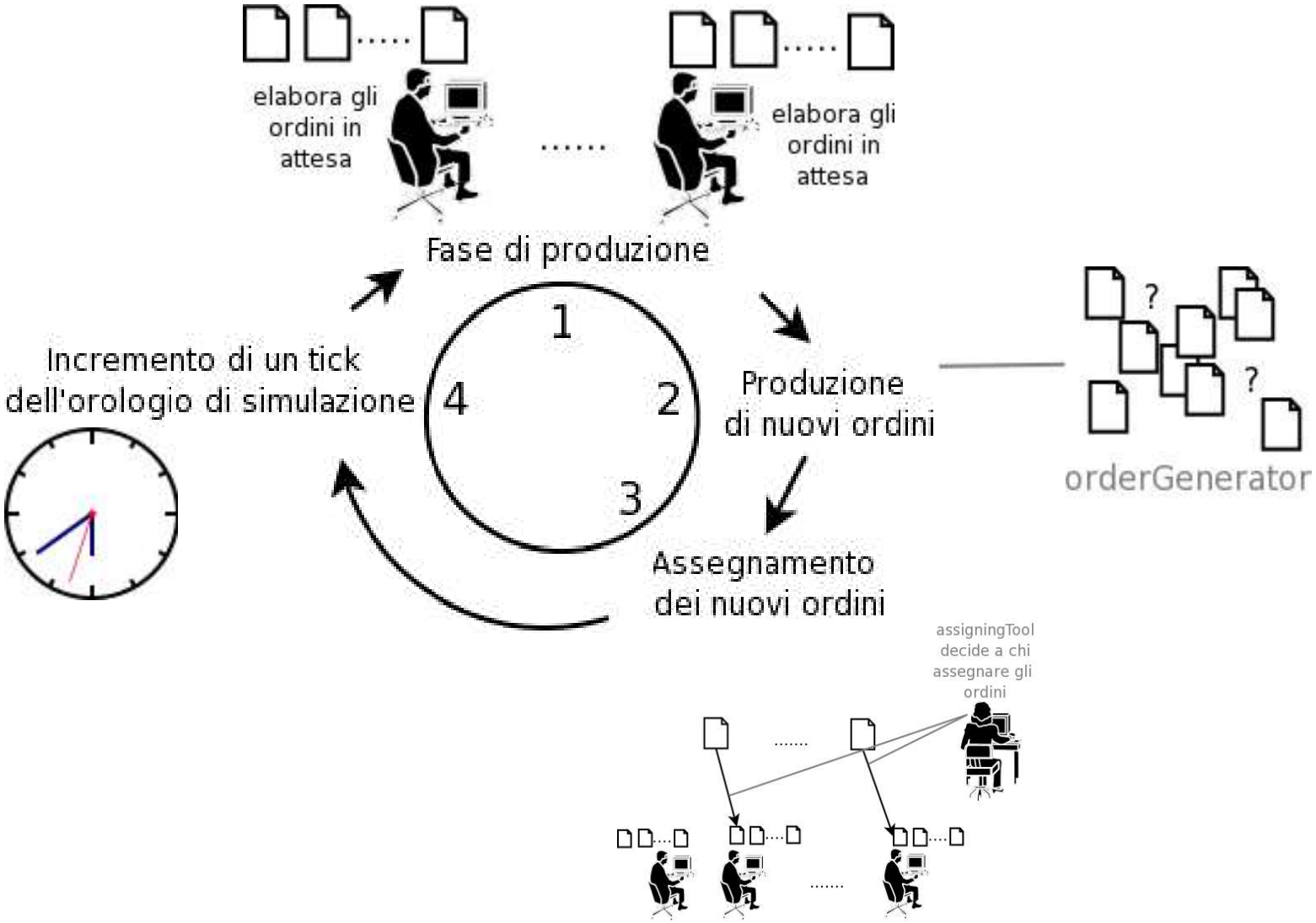
jESLet



jESLet

- È il modello light di jES
- Ossia è la versione didattica di jES
  - La maggior parte delle caratteristiche di jES è rimossa
  - Non può essere usata per simulare e modellare veri e propri casi di studio
  - Serve per capire il funzionamento della piataforma di simulazione jES
- Costituisce il nucleo centrale dell'architettura di jES

jESLet


elabora gli ordini in attesa
elabora gli ordini in attesa
Fase di produzione
1
2
3
4
Incremento di un tick dell'orologio di simulazione
Produzione di nuovi ordini
Assegnamento dei nuovi ordini
?
?
orderGenerator
assigningTool decide a chi assegnare gli ordini

JESLet: comportamento del simulatore

- Possiamo descrivere il comportamento del simulatore supponendo che il processo di simulazione sia già incominciato e sia quindi in grado di elaborare ordini:

1. Fase di produzione,
2. Produzione di nuovi ordini,
3. Assegnamento casuale dei nuovi ordini,
4. Incremento del tick dell'orologio di simulazione.

1. Fase di produzione (1/1)

- Le unità di produzione operano sugli ordini presenti nelle proprie liste di attesa, se ce ne sono;
- Ogni ordine viene elaborato per ogni tick dell'orologio di simulazione.
- Una volta processati gli ordini vengono messi in una lista di produzione eseguita per essere quindi diffusi ad altre unità di produzione;

2. Produzione di nuovi ordini (1/1)

- i nuovi ordini sono messi in produzione
  - generati casualmente dall'oggetto ***orderGenerator***;
- ogni ordine contiene la ricetta contenente i passi che devono essere eseguiti;
- i nuovi ordini sono assegnati alle *unità di produzione* come mostrato al punto 3;

3. Assegnamento dei nuovi ordini (1/1)

- ogni ordine contenuto nella ***produzione eseguita*** (*made production list*) delle *unità di produzione* o appena creato (come nel punto 2) compie una ricerca
  - sfruttando l'oggetto chiamato ***assigning tool***,
  - per scoprire se una o più *unità di produzione* è in grado di eseguire i passi rimanenti affinché la ricetta (e quindi l'ordine) sia completata.
- l'ordine viene assegnato alla coda d'attesa della prima *unità di produzione* che dà risposta affermativa con politica FIFO.
- un ordine viene scartato quando l'ultimo suo passo viene compiuto

jESLet e jES
con
Eclipse

Importare jESLet in Eclipse

- Creare un nuovo progetto java (chiamato jESLet)
- Importare le librerie
  - swarm.jar (/usr/share/swarm/swarm.jar)
- Importare da file system i file .java presenti nella cartella
  - jeslet-1.0.0/src
- Importare il file jeslet-0.1/jeslet.scm nella root del progetto
- Nel progetto creare una cartella chiamata unitData e importare il file
  - jeslet-1.0.0/unitData/unitBasicData.txt

Importare jES in Eclipse (1/3)

- Creare un nuovo progetto Java, importando
  - la libreria

  '/usr/share/swarm/swarm.jar ' in ambiene unix, o

  'C:\Swarm-2.2\share\swarm\swarm.jar' in ambiente windows,
  - la libreria 'jesframe-9.../lib/xlrd.jar',
  - la libreria 'jesframe-9.../lib/plot.jar',
- Iimportare da 'file system' i file .java presenti nella cartella 'jesframe-9...\src'. (eliminate se volete i 'warnings')
- Nella cartella del progetto importare da 'file system' il file 'jesframe.scm' presente nella cartella 'jesframe-9...',

Importare jES in Eclipse (2/3)

- Nella cartella del progetto creare una nuova cartella chiamata 'unitData' e in essa importate i file presenti nella cartella 'jesframe-9...\unitData\', in particolare:
  - 'unitBasicData.txt'
  - 'informationFlowMatrix.txt'
  - 'units.xls'
  - 'memoryMatrixes.txt'
- Nella cartella del progetto creare una nuova cartella chiamata 'log' e in essa importate il file 'concludedOrderLog.txt' 'presente nella cartella 'jesframe-9...\log\',

Importare jES in Eclipse (3/3)

- Nella cartella del progetto creare una nuova cartella chiamata 'Revenues' e in essa importate tutti i file presenti nella cartella 'jesframe-9...\Revenues\',
- Nella cartella del progetto creare una nuova cartella chiamata 'Benefit' e in essa importate tutti i file presenti nella cartella 'jesframe-9...\Benefit\',
- Nella cartella del progetto creare una nuova cartella chiamata 'Costs' e in essa importate tutti i file presenti nella cartella 'jesframe-9...\Costs\',
- Aprite il file 'StartESFrame.java' e create un nuovo 'Run'. Quindi eseguite il run.

Riferimenti

- http://www.eclipse.org
- http://www.eclipseuml.com
- http://www.swarm.org
- http://web.econ.unibo.it/terna/jes/