Appendice xxx

Analisi del corretto funzionamento della versione jVEFrame-0.7.0.a

Per verificare l'esattezza della versione *jVEFrame-0.7.0.a* riportiamo l'output dell'esecuzione passo per passo effettuata con riferimento alla programmazione dei materiali e delle componenti basando l'analisi su un numero di turni ridotti, come riportato in tabella xxx esemplificativa del file *orderSequences.xls*.

1	100	*	2	200	*	3							;
2	100	*	2	200	*	3	300	*	1	400	*	5	;
3							300	*	3	400	*	1	;

Tabella 1: Programmazione dei materiali e delle componenti, esempio su tre turni.

Le sequenze produttive, con i relativi tempi di esecuzione, sono estratti dalla tabella xxx, anche questa esemplificativa del file *recipes.xls*.

#	comment	;																			
#	comment	;																			
	alfa	100	1	s	10	2	s	2	3	s	4	4	s	3	5	s	1	e	100	;	
	beta	200	6	s	5	2	s	1	7	s	1	8	m	1	4	s	2	e	200	;	
	teta	300	p	1	100	9	s	10													;
#	comment	;																			
	sigma	400	p	1	200	10	s	1													;

Tabella 2: Sequenze di fasi di produzione, esempio di quattro ricette.

per poter impostare gli oggetti rappresentativi delle singole unità di produzione la struttura della simulazione prevede, dopo aver creato le

sonde necessarie per impostare i parametri, la lettura di dati dai file riportati nelle tabelle che seguono.

unit_#
100
200

Tabella 3: Esempio del file endUnit.txt.

unit_#	useWarehouse	prod.phase_#	fixed_costs	variable_costs
1	1	1	1	1
2	1	2	1	1
3	1	3	1	1
4	1	0	0	0
5	1	5	1	1
6	1	6	1	1
7	1	7	1	1
8	1	8	1	1
9	1	0	0	0
10	1	0	0	0

Tabella 4: Esempio del file unitBasicData.txt.

3				
	4	1	1	1
	1	1	1	1
	8	1	1	1

Tabella 5: Esempio del file units.xls, unità complessa n°4.

2				
	9	1	1	1
	8	1	1	1

Tabella 6: Esempio del file units.xls, unità complessa n°9.

4				
	10	1	1	1
	7	1	1	1
	4	1	1	1
	8	1	1	1

Tabella 7: Esempio del file units.xls, unità complessa n°10.

La simulazione inizia con la creazione degli oggetti necessari al funzionamento all'interno del metodo buildObject() del VEFrameModelSwarm.java:

- Sono create le unità finali;

End unit number 100 has been created.

End unit number 200 has been created.

- Sono create le unità semplici;

Unit number 1 has been created.

Unit number 2 has been created.

Unit number 3 has been created.

Unit number 4 has been created.

- Come si può osservare nella tabella n° 4 la quarta unità, così come la nona e la decima, è una unità complessa…

Unit 4 is a special one.

…in grado di effettuare tre fasi di lavorazione differenti…

special unit 4 nOfPhasesToDealWith 3

…ognuna delle quali con propri costi fissi e variabili.

special unit 4 fixed costs 1.0 variable costs 1.0

special unit 4 fixed costs 1.0 variable costs 1.0

special unit 4 fixed costs 1.0 variable costs 1.0

- Tutte le unità sono state create

Unit number 5 has been created…

… … … … …

…Unit 10 is a special one.

- Come spiegato in precedenza (vedi capitolo xxx) non sono stati utilizzati i magazzini e le informazioni;

No warehouses generated

- Si effettua il richiamo al metodo, presente nella classe OrderDistiller.java, necessario per definire il dizionario delle possibili fasi di lavorazione. Avviene la determinazione della lunghezza e l'archiviazione dei nomi delle fasi.

Length of the dictionary (# of codes identifying steps in recipes): 18

0 100

1 200

2 1

3 2

4 3

5 4

6 1

7 8

8 5

9 6

10 7

11 8

12 9

13 8

14 10

15 7

16 4

17 8

- Lo stesso richiamo al metodo setDictionary() avvia anche la lettura del file recipes (vedi tabella 1). Se la riga contiene un commento questo è segnalato…

The row # 1 contains a comment

The row # 2 contains a comment

…altrimenti se la riga contiene una ricetta si determina la lunghezza della stessa ed è creato un esemplare della classe Recipe.java.

The row # 3 contains a recipe named alfa

OrderDistiller: aRecipe with code 100was born

The row # 4 contains a recipe named beta

OrderDistiller: aRecipe with code 200was born

The row # 5 contains a recipe named teta

OrderDistiller: aRecipe with code 300was born

The row # 6 contains a comment

The row # 7 contains a recipe named sigma

OrderDistiller: aRecipe with code 400was born

The row # 1 contains a comment

The row # 2 contains a comment

The row # 3 contains a comment

- Una volta che lo schedule è avviato sono richiamate tutte le funzioni della simulazione. Anzitutto al primo *tick* del turno vengono azzerati i costi della contabilità e poi è eseguita la produzione assegnando ad ogni unità il primo passo della ricetta che è in grado di fare;

Step 1 in unit 1; waitingList count 0

Step 1 in unit 2; waitingList count 0

Step 1 in unit 3; waitingList count 0

Step 1 in unit 4; waitingList count 0

Step 1 in unit 5; waitingList count 0

Step 1 in unit 6; waitingList count 0

Step 1 in unit 7; waitingList count 0

Step 1 in unit 8; waitingList count 0

Step 1 in unit 9; waitingList count 0

Step 1 in unit 10; waitingList count 0

- Dopo che il primo passo della ricetta di produzione è stato assegnato, gli ordini sono assegnati ad una lista di produzione giornaliera;

Step 2 in unit 1 dailyProductionList count 0

Step 2 in unit 2 dailyProductionList count 0

Step 2 in unit 3 dailyProductionList count 0

Step 2 in unit 4 dailyProductionList count 0

Step 2 in unit 5 dailyProductionList count 0

Step 2 in unit 6 dailyProductionList count 0

Step 2 in unit 7 dailyProductionList count 0

Step 2 in unit 8 dailyProductionList count 0

Step 2 in unit 9 dailyProductionList count 0

Step 2 in unit 10 dailyProductionList count 0

Lo schedule richiama il metodo distill() della classe OrderDistiller.java, il quale avvia la lettura del file orderSequences (visto in tabella 1) e si avvia così il turno relativo…

The shift #1has begun

…di conseguenza tutti gli ordini di produzione che devono essere evasi nel turno vengono instanziati e si comunica la loro presenza alla console;

New order (#1)

- Il programma Order,java crea due vettori che definiscono: lo stato di completamento delle fasi di lavorazione…

orderState vector 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

…il contenuto del vettore delle fasi di lavorazione;

orderRecipe vector 1 1 1 1 1 1 1 1 1 1 2 2 3 3 3 3 4 4 4 5 100

- Gli ordini sono poi propagati verso le unità che rispondono se sono in grado di eseguire una determinata lavorazione;

The unit 1 has received the order # 1

New order (#2)

orderState vector 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

orderRecipe vector 1 1 1 1 1 1 1 1 1 1 2 2 3 3 3 3 4 4 4 5 100

The unit 1 has received the order # 2

New order (#3)

*orderState vector 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

*orderRecipe vector 6 6 6 6 6 2 7 8 8 8

Step 2 in unit 6 dailyProductionList count 1

The unit 6 has received the order # 3

Step 2 in unit 7 dailyProductionList count 0

Step 2 in unit 8 dailyProductionList count 0

Step 2 in unit 9 dailyProductionList count 0

Step 2 in unit 10 dailyProductionList count 0

- Terzo tick…

Step 1 in unit 1; waitingList count 2…

…Step 1 in unit 10; waitingList count 0

Step 2 in unit 1 dailyProductionList count 1

The unit 1 has received the order # 2

Step 2 in unit 2 dailyProductionList count 0

Step 2 in unit 3 dailyProductionList count 0

Step 2 in unit 4 dailyProductionList count 0

Step 2 in unit 5 dailyProductionList count 0

Step 2 in unit 6 dailyProductionList count 1

The unit 6 has received the order # 4

Step 2 in unit 7 dailyProductionList count 0

Step 2 in unit 8 dailyProductionList count 0

Step 2 in unit 9 dailyProductionList count 0

Step 2 in unit 10 dailyProductionList count 0

- Al sesto *tick* l'ordine numero tre passa alla unità numero due per effettuare la lavorazione numero due. Stessa cosa succede poi agli altri ordini;

Step 1 in unit 1; waitingList count 2…

…Step 1 in unit 10; waitingList count 0

Step 2 in unit 1 dailyProductionList count 1

The unit 1 has received the order # 1

Step 2 in unit 2 dailyProductionList count 0

Step 2 in unit 3 dailyProductionList count 0

Step 2 in unit 4 dailyProductionList count 0

Step 2 in unit 5 dailyProductionList count 0

Step 2 in unit 6 dailyProductionList count 1

The unit 2 has received the order # 3

Step 2 in unit 7 dailyProductionList count 0

Step 2 in unit 8 dailyProductionList count 0

Step 2 in unit 9 dailyProductionList count 0

Step 2 in unit 10 dailyProductionList count 0

- All'ottavo *tick* l'ordine numero tre passa alla unità numero quattro per effettuare la lavorazione numero otto. L'unità numero quattro è complessa ed è perciò in grado di effettuare più tipi di lavorazioni. La scelta tra l'unità quattro e la otto avviene considerando la prima delle due che risponde. Stessa cosa succede poi agli altri ordini;

Step 1 in unit 1; waitingList count 2…

…Step 1 in unit 10; waitingList count 0

Step 2 in unit 1 dailyProductionList count 1

The unit 1 has received the order # 1

Step 2 in unit 2 dailyProductionList count 1

The unit 7 has received the order # 4

Step 2 in unit 3 dailyProductionList count 0

Step 2 in unit 4 dailyProductionList count 0

Step 2 in unit 5 dailyProductionList count 0

Step 2 in unit 6 dailyProductionList count 1

The unit 2 has received the order # 5

Step 2 in unit 7 dailyProductionList count 1

The unit 4 has received the order # 3

Step 2 in unit 8 dailyProductionList count 0

Step 2 in unit 9 dailyProductionList count 0

Step 2 in unit 10 dailyProductionList count 0

- All'inizio del novantunesimo *tick* parte il secondo turno;

The shift #2has begun

- Nuovi ordini vengono creati esattamente come visto in precedenza;

New order (#6)

orderState vector 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

orderRecipe vector 1 1 1 1 1 1 1 1 1 1 2 2 3 3 3 3 4 4 4 5 100

The unit 1 has received the order # 6

New order (#7)

orderState vector 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

orderRecipe vector 1 1 1 1 1 1 1 1 1 1 2 2 3 3 3 3 4 4 4 5 100

The unit 1 has received the order # 7

New order (#8)

*orderState vector 0 0 0

va che all'undicesimo ordine in posizione zero è presente un procurement, si necessita del semilavorato con codice 100;

Proc. spec. created (by order 11 at pos. 0; NB first pos. is 0)

item(s) to be procured 100

New order (#11)

- Il primo passo della ricetta che necessita di un procurement viene indicato con valore negativo;

orderState vector 0 0 0 0 0 0 0 0 0 0

orderRecipe vector -9 9 9 9 9 9 9 9 9 9

Proc. spec. created (by order 12 at pos. 0; NB first pos. is 0)

item(s) to be procured 200

New order (#12)

orderState vector 0

orderRecipe vector -10

Proc. spec. created (by order 13 at pos. 0; NB first pos. is 0)

item(s) to be procured 200

New order (#13)

orderState vector 0

orderRecipe vector -10

Proc. spec. created (by order 14 at pos. 0; NB first pos. is 0)

item(s) to be procured 200

New order (#14)

orderState vector 0

orderRecipe vector -10

Proc. spec. created (by order 15 at pos. 0; NB first pos. is 0)

item(s) to be procured 200

New order (#15)

orderState vector 0

orderRecipe vector -10

Proc. spec. created (by order 16 at pos. 0; NB first pos. is 0)

item(s) to be procured 200

New order (#16)

orderState vector 0

orderRecipe vector -10

- La produzione è quindi assegnata come già visto in precedenza;

Step 1 in unit 1; waitingList count 2…

…Step 1 in unit 10; waitingList count 0

Step 2 in unit 1 dailyProductionList count 1…

…Step 2 in unit 10 dailyProductionList count 0

-

eu 100 2 eu 200 0 as seen in procurementAssembler of unit 9

proc. spec. of order 11 at pos. 0 items 100

eu 100 1 eu 200 0 as residual in procurementAssembler of unit 9

completed

The end unit 100 has removed the order # 1

The unit 9 has received the order # 11

eu 100 1 eu 200 0 as seen in procurementAssembler of unit 10
proc. spec. of order 12 at pos. 0 items 200
eu 100 1 eu 200 -1 as residual in procurementAssembler of unit 10

eu 100 1 eu 200 0 as seen in procurementAssembler of unit 10
proc. spec. of order 13 at pos. 0 items 200
eu 100 1 eu 200 -1 as residual in procurementAssembler of unit 10

eu 100 1 eu 200 0 as seen in procurementAssembler of unit 10
proc. spec. of order 14 at pos. 0 items 200
eu 100 1 eu 200 -1 as residual in procurementAssembler of unit 10

eu 100 1 eu 200 0 as seen in procurementAssembler of unit 10
proc. spec. of order 15 at pos. 0 items 200
eu 100 1 eu 200 -1 as residual in procurementAssembler of unit 10

eu 100 1 eu 200 0 as seen in procurementAssembler of unit 10
proc. spec. of order 16 at pos. 0 items 200
eu 100 1 eu 200 -1 as residual in procurementAssembler of unit 10

Step 1 in unit 1; waitingList count 2…
…Step 1 in unit 10; waitingList count 0

Step 2 in unit 1 dailyProductionList count 1…
…Step 2 in unit 10 dailyProductionList count 0

The shift #3has begun

The simulation finishes here. It is not the right and normal way, but is only for explanation. Thank you for your attention and interest.