Università degli studi di Padova

Facoltà di Ingegneria

Dipartimento di tecnica e gestione dei sistemi industriali

Tesi di Laurea di Primo Livello

Assistenza perinatale: modello e codifica in linguaggio Arena

Relatore: Prof. Romanin-Jacur Giorgio

Laureando: Carraro Daniele

Anno accademico 2010–2011

INDICE

SOMMARIO

In questa tesi cercherò di esplicare i vari passaggi affrontati nella traduzione di un modello informatico da Micro Saint ad Arena.
Entrambi sono software utilizzati per le simulazioni di scenari reali complessi e trovano ampia applicazione soprattutto nell'ambito della gestione aziendale.
Sia per Micro Saint che per Arena sono state utilizzate le versioni Student, ovvero versioni di prova messe a disposizione dalle stesse software house e che tutti possono scaricare gratuitamente.
Il modello di partenza invece, una simulazione del sistema di assistenza perinatale, mi è stato gentilmente concesso dal professor Romanin-Jacur, uno dei creatori del modello.
Nel corso della trattazione verrà poi esposto in modo dettagliato come i vari elementi presenti nella simulazione in Micro Saint siano stati tradotti in linguaggio Arena.
Prima si analizzeranno i moduli di inizio e fine modello, i create ed i dispose; poi si passerà alla struttura delle task in Micro Saint e come queste siano state scomposte in diversi blocchi in Arena; verranno inoltre analizzati i vari tipi di variabili ed i modi e metodi per poterle creare e per poterci assegnare diversi valori. Si parlerà poi dei cicli *while* e delle tecniche utilizzate per poterli tradurre in Arena utilizzando dei decisions e l'operatore *if*. Infine esporrò come i diversi linguaggi implementano la gestione dello scorrere del tempo e successivamente le varie differenze tra i moduli decisionali presenti in Micro Saint e quelli presenti in Arena.
Nel successivo capitolo analizzerò in modo più specifico il modello in Arena, cercando sempre di metterlo in relazione con il modello di partenza.
È stato necessario suddividere il modello finale in sottoparti perché la complessità ed il numero di moduli di quest'ultimo è veramente molto maggiore rispetto al programma di partenza. In Micro Saint infatti le varie tasks possono contenere varie informazioni di diversa natura, ogni singola task infatti è simile ad un mini modello contenete istruzioni iniziali, di lancio, finali e con una propria gestione dei tempi.
In Arena invece i moduli sono molto più specializzati e simulano solo un singolo aspetto dell'intero sistema. Infatti per tradurre una task di Micro Saint si è a volte costretti ad utilizzare svariati moduli del linguaggio Arena.

INTRODUZIONE

Nei moderni scenari di mercato, in cui sempre più aziende lottano per ottenere la leadership nel proprio settore, l'ottimizzazione e la ricerca della massima efficienza sono diventati aspetti chiave e fattori critici di successo.

Ogni buon decisore deve quindi saper prevedere tutte le possibili situazioni future nel modo più corretto e completo possibile ed avere inoltre pieno controllo delle operations aziendali individuando inefficienze e potendo successivamente intervenire per apportare le opportune correzioni.

Alcuni tra gli strumenti più utilizzati per raggiungere tali scopi sono i cosiddetti software di 'simulation and modeling'. Questi programmi permettono la costruzione di specifici modelli attraverso l'utilizzo dei moduli, ovvero box di varie forme rappresentativi delle varie fasi di processo.

L'utilizzo di questi software non è però ristretto al solo ambito industriale. È possibile difatti ricreare qualsiasi sequenza di attività, purché se ne conoscano tutti i vari passaggi e i meccanismi logici che li legano tra loro.

Un esempio della duttilità di tali piattaforme ci viene offerto da Facchin Paola, Ferrante Anna, Rizzato Elena, Romanin-Jacur Giorgio e Salmaso Laura che, grazie a Micro Saint, hanno realizzato un modello rappresentativo del network di assistenza perinatale.

Le regioni italiane sono infatti divise in distretti sanitari ognuno dei quali ha vari reparti di ostetricia. Non tutti però sono in grado di fornire le cure necessarie ai bambini nati prematuramente oppure in gravi condizioni. E' quindi di fondamentale importanza riuscire ad individuare il numero dei futuri neonati e l'effettiva capacità di ogni cento medico per poter indirizzare le madri con i loro bambini nel centro più vicino e più idoneo alle loro esigenze.

Questa simulazione è stato poi applicata alla regione Veneto portando benefici sia ai vari distretti sanitari, sia alle neomamme che ai loro figli.

Lo scopo di questa trattazione sarà proprio la traduzione di questo modello da MicroSaint ad Arena, altro comune software di 'simulation and modeling'.

Si metteranno inoltre in luce le varie analogie tra i due linguaggi e si cercherà di spiegarne le differenze terminando poi con una precisa descrizione delle varie parti del modello tradotto in Arena.

CAPITOLO 1: Il modello ed i software utilizzati

1.1.Micro Saint

Figura 1.1.Il logo di Micro Saint

Micro Saint è un software di simulazione di eventi discreti, con i suoi modelli è possibile ottenere importanti informazioni sui processi che potrebbero essere troppo costosi in termini di tempo e risorse da testare nel mondo reale.

Il software è basato su quattro elementi base: ellissi, detti tasks, rappresentano le attività con il loro uso di risorse e di tempo; frecce, rappresentano la sequenza con cui si succedono le varie attività; rombi, rappresentano le decisioni; piccoli rettangoli sovrapposti, rappresentano le code.

Figura 1.2.Barra di creazione oggetti in Micro Saint

Fu inizialmente sviluppato dalla *MA&D*, azienda che nel 2006 è stata acquisita dalla *Alion Science and Technology*.

La versione utilizzata per il modello di partenza è stata Micro Saint Student.

1.2.Arena

Figura1.3.Il logo di Arena

Arena, altro software di simulazione, fu inizialmente sviluppato da *Systems Modeling* e successivamente acquisito da *Rockwell Automation* nel 2000. È interamente basato sul linguaggio denominato SIMAN.
Rispetto a Micro Saint presenta un maggior numero di moduli. In figura 1.4 vengono appunto illustrati i blocchi presenti in Arena.

Figura 1.4.I diversi tipi di moduli

I moduli che saranno utilizzati per questa trattazione saranno:

- Create, la task iniziale;
- Assign, utilizzato per creare nuove variabili o per assegnarle nuovi valori;
- Process, utilizzato per simulare ritardi e giacenze;
- Decide, utilizzato se siamo in presenza di delle scelte opzionali o probabilistiche;
- Separate, utilizzato per dividere il programma in più sottorami;
- Dispose, la task finale.

Anche di questo software avrò a disposizione la Student Edition.

1.3.Il problema reale alla base del modello

Solitamente in Europa e negli USA l'assistenza perinatale è suddivisa in tre livelli:

1°. assistenza base;

2°. assistenza intermedia per problemi neonatali;

3°. assistenza intensiva.

Il primo livello viene applicato ai casi di gravidanza normali, senza complicazioni ed è caratterizzato da una assistenza continua per madre e figlio; il secondo livello si applica ai casi a basso rischio o nei parti prematuri con gestazioni fino a 30 settimane ed è caratterizzato da un continuo monitoraggio da parte di specialisti ed anche dall'uso di incubatrici; il terzo livello viene applicato nei casi ad alto rischio o quando la gestazione è stata inferiore alle 28 settimane ed è caratterizzata dalla presenza costante di specialisti pronti ad intervenire nelle situazioni più critiche e dall'utilizzo di incubatrici ventilate artificialmente.

Bisogna inoltre stabilire il numero di nascite per anno, la permanenza media degli utenti e considerare che non tutti i centri medici riescono a garantire tutti i livelli di assistenza.

Durante la gravidanza potrebbero presentarsi varie complicazioni, molte sono diagnosticabili in anticipo, altre però sono inaspettate e quindi richiedono un pronto intervento ed un elevato livello di assistenza.

Generalmente una donna incinta dovrebbe essere ammessa in un centro che presenti un adeguato livello di assistenza. Il nascituro però potrebbe richiedere un livello di assistenza superiore a causa di complicazioni inaspettate e un successivo trasferimento urgente in un centro che possa garantire un livello superiore.

Conseguentemente i centri con i più bassi livelli, uniti alle richieste provenienti direttamente dai centri urbani, potrebbero diventare una fonte di future richieste per i centri maggiormente specializzati.

1.4.Il modello in Micro Saint

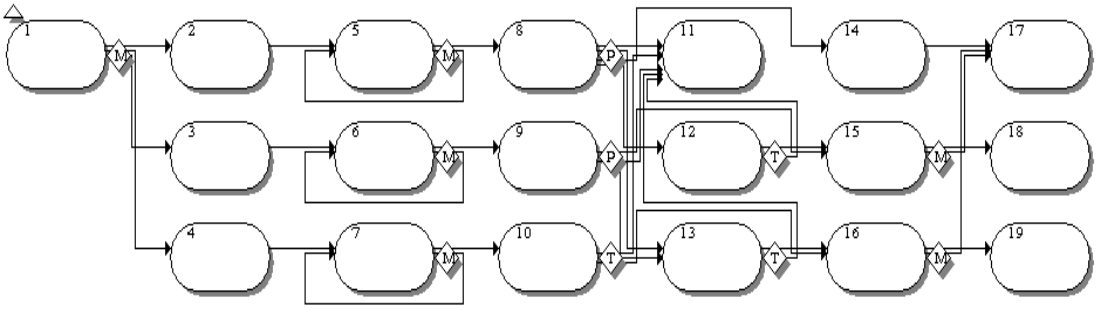

Figura 1.5.Il modello in MicroSaint

Il modello, rappresentato in Figura 1.5, è composto complessivamente da 19 ellissi o tasks.
L'ellisse 1 rappresenta la situazione iniziale, gli ellissi 2-4 vengono utilizzati per inserire tutte le variabili e i parametri utilizzati dal modello, ovvero il tasso di domanda per ogni livello di assistenza, la capacità e distanza di ogni centro. Negli ellissi 5-7 viene generata la domanda per tutti e tre i livelli, in 8-10 vengono decise le destinazioni di mamme e neonati e i possibili trasferimenti per i neonati. Nell'ellisse 11 vengono convogliate le mamme ed i neonati ammessi nei centri fuori regione, gli ellissi 12-13 gestiscono il trasferimento dei neonati, gli ellissi 14-16 sono tasks con scopi funzionali al modello.
Infine l'ellisse 17 rappresenta l'ammissione delle neo mamme e gli ellissi 18-19 l'ammissione dei neonati al secondo e terzo livello di assistenza.
I parametri di distribuzione iniziali sono stati ottenuti elaborando i dati storici dei vari distretti sanitari.
Lo scopo del modello è quello di evidenziare la saturazione di tutti i distretti sanitari ed il tasso di occupazione dei loro posti letto.

CAPITOLO 2: La traduzione

Pur essendo entrambi software di simulazione, Micro Saint ed Arena presentano numerose differenze tra i loro linguaggi.

Se infatti il modello originario conta 30 moduli, 19 tasks più 11 elementi decisionali, il corrispettivo modello in Arena ne conta circa 90.

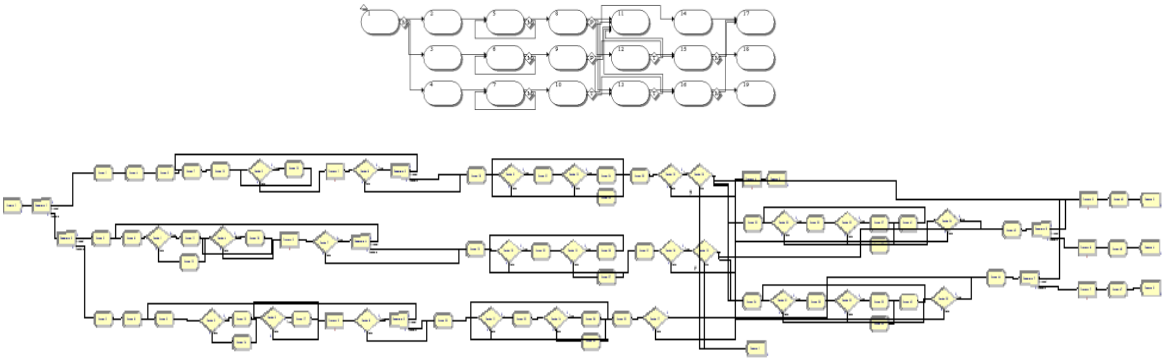

Figura 2.1.I modelli a confronto. In alto Micro Saint, sotto Arena

È quindi utile definire i vari passaggi della traduzione e come i diversi elementi siano stati trasformati durante la codifica.

2.1.Create e dispose

Se in Micro Saint è necessario inserire solo lo ‘start’, un piccolo triangolino posizionato poco sopra ed a sinistra del blocco di partenza, figura 2.1, in Arena invece ogni modello deve cominciare con un modulo denominato ‘create’ e terminare con un altro modulo denominato ‘dispose’.

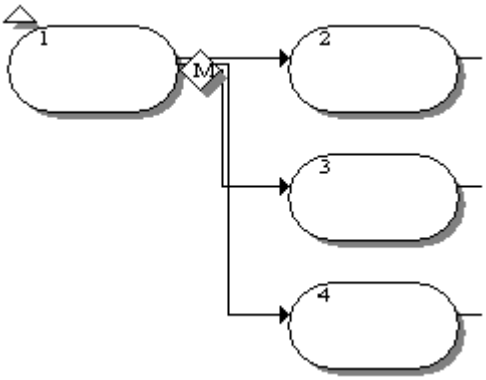

Figura 2.2.Inizio del modello in Micro Saint

Nei moduli create si possono impostare l'entità e la frequenza degli arrivi che andranno poi a percorrere l'intero modello, è inoltre possibile definirne il numero massimo e il momento esatto in cui comincerà la loro creazione.
Nei moduli dispose invece abbiamo la possibilità di salvare l'entità delle statistiche.
Come mostrato in figura 2.3, ogni sottoramo del modello deve terminare con un dispose.

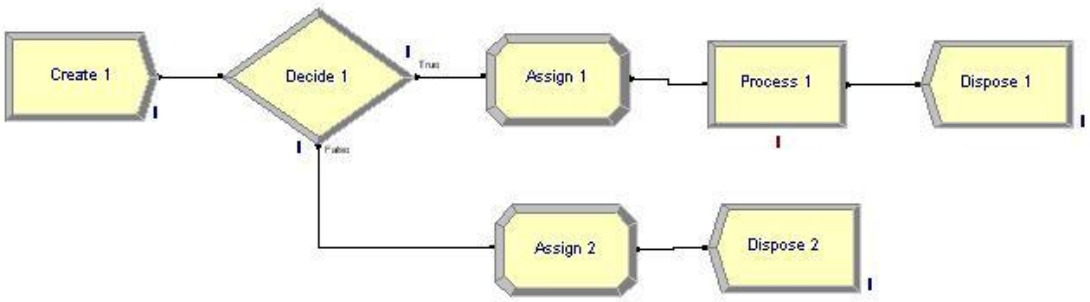

Figura 2.3.Esempio con create e dispose in Arena

2.2.Struttura delle tasks

Le tasks, ovvero i moduli ellittici in Micro Saint, sono elementi molto complessi e talmente ricchi di informazioni da risultare quasi dei sottomodelli.
Al loro interno infatti è possibile impostare:

- le informazioni relative al tempo impiegato (distribuzione, media, deviazione standard);
- Release condition, ovvero le condizioni d'entrata;
- Beginning effect, ovvero gli effetti iniziali;
- Launch effect, ovvero gli effetti di lancio;
- Ending effect, ovvero gli effetti finali della nostra task.

L'ordine con cui vengono eseguite le varie istruzioni è il seguente:

1 Release effect
2 Beginning effect
3 Mean time
4 Standard deviation
5 Launch effect
6 Ending effect.

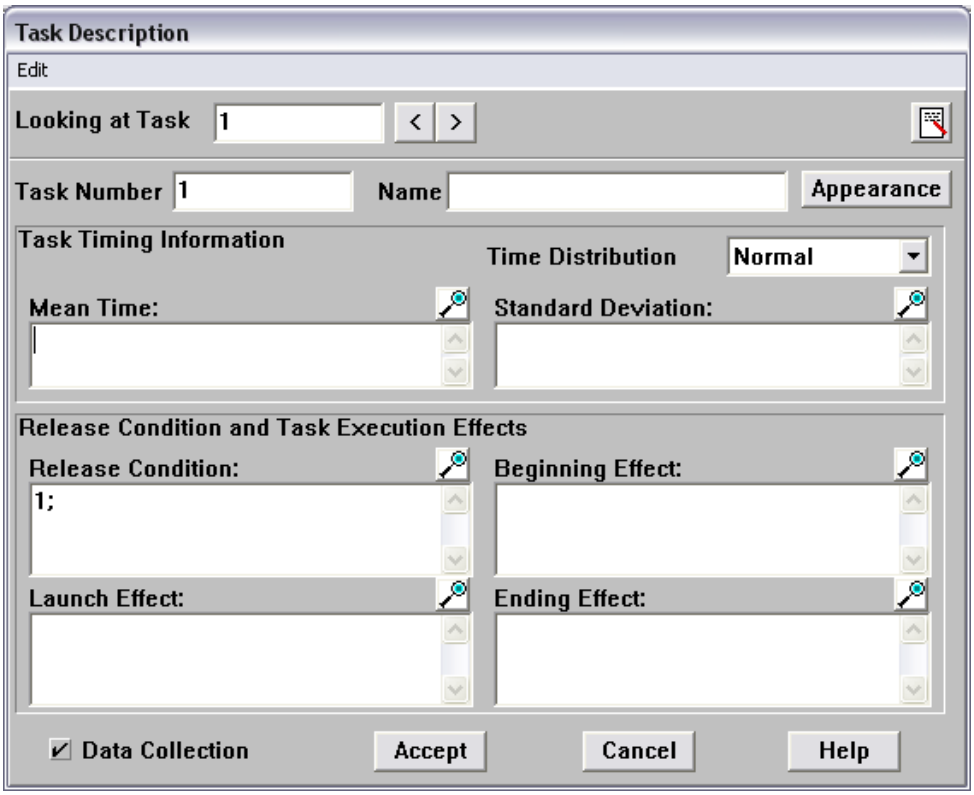

Figura 2.4.Struttura interna di una task

In Arena invece non abbiamo a disposizione moduli così completi, si devono quindi scindere le varie istruzioni in moduli minori, aumentando così la corposità del modello stesso.

Infatti se dovessimo tradurre una task che presenti istruzioni sia nel beginning, sia nel launch, sia nell'ending effect, sarebbero necessari almeno 3 moduli del linguaggio Arena. A volte però, in presenza di istruzioni complesse, i moduli potrebbero essere molti di più.

2.3.Creazione ed assegnazione di valori alle variabili

In Micro Saint la creazione e l'assegnazione di valori alle variabili avviene all'interno delle task.

Le istruzioni vanno semplicemente inserite nel riquadro relativo al beginning effect oppure all'ending effect.

Per creare variabili in più dimensioni è necessario indicare tra parentesi quadre le coordinate di nostro interesse. Ad esempio:

- variabile semplice: a:=3
- variabile monodimensionale: a[2]:=0.35
- variabile bidimensionale: a[4,1]:=2.87

In Arena invece i relativi moduli hanno sempre forma ellittica ma vengono chiamati ‘assign’.
Per poter creare ed assegnare valori alle variabili è necessario però utilizzare l’apposito strumento mostrato in figura 2.5.

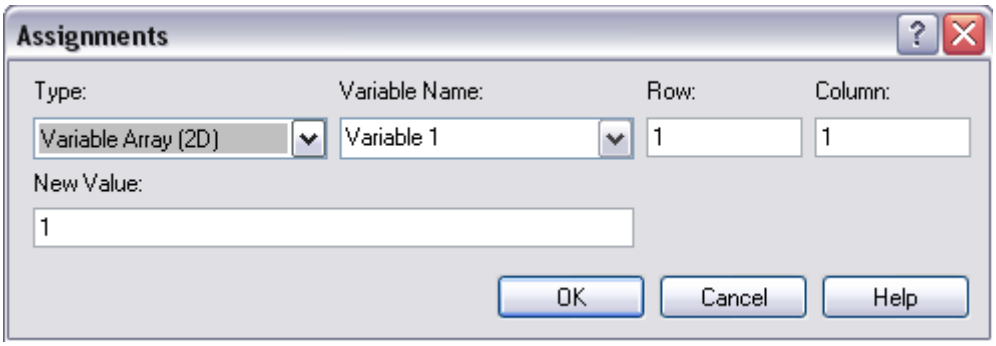

Figura 2.5.Assignments in Arena

2.3.1.Variabili con valori casuali

Nel modello in Micro Saint è stato fatto uso di una variabile monodimensionale, *xx*, alla quale sono stati associati valori casuali secondo una distribuzione uniforme.
Per poter tradurre *xx* in Arena è stato necessario l’uso del comando *UNIF(a,b)*. Questo operatore infatti restituisce numeri casuali da una distribuzione uniforme che ha come valore minimo *a* e media *b*.
In figura 2.6 è possibile osservare la stringa di comando utilizzata.

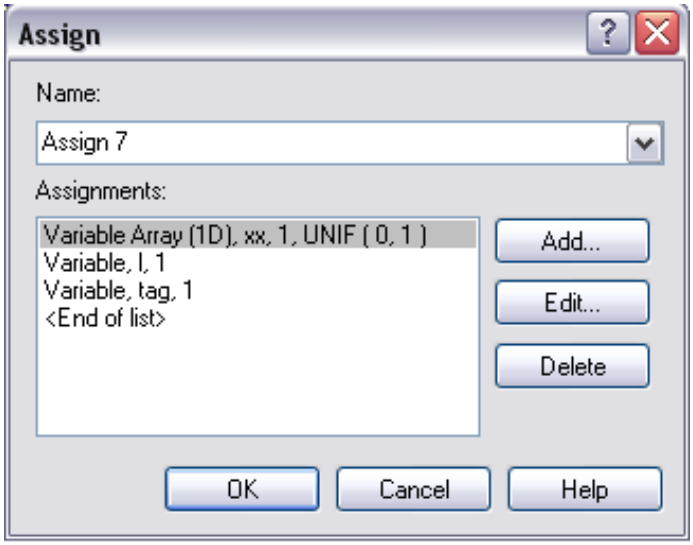

Figura 2.6.Linea di comando per la variabile *xx* in Arena

2.4.*While do*

In Micro Saint è possibile inserire le varie istruzioni per i cicli *while* direttamente all'interno delle tasks.

In Arena invece non è possibile, per ogni ciclo *while* è stato quindi necessario l'utilizzo dei decides e dell'operatore *if*.

Eccone un esempio:

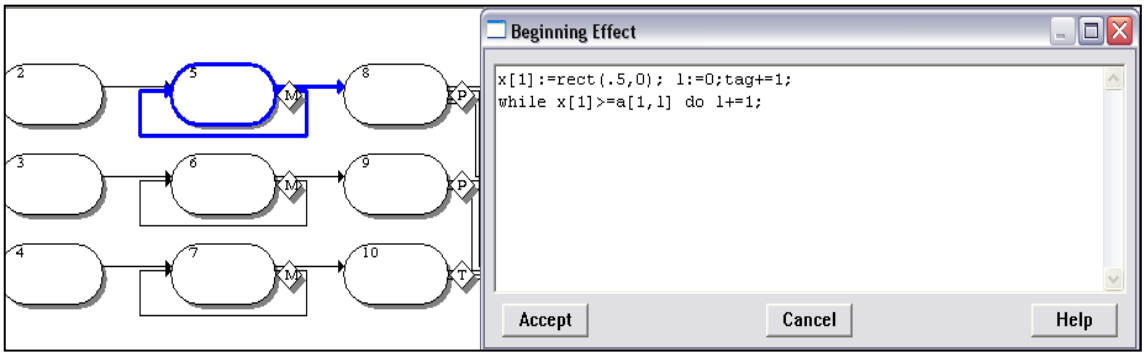

Figura 2.7.Ciclo *while* in Micro Saint

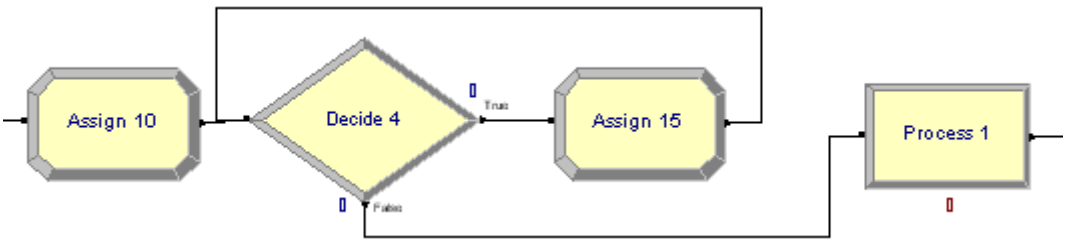

Figura 2.8.Lo stesso ciclo tradotto in Arena

Nel decide 4 infatti è stato inserito il comando *if xx(1) >= aaa(1, l)*, mentre nell'assign 15 è presente l'incremento della variabile *l*.

2.5.Gestione del tempo

Come già anticipato nei precedenti paragrafi, in Micro Saint la gestione dello scorrere del tempo viene attuata direttamente nelle varie task.

Figura 2.9.Sezione di task relativa alla gestione del tempo in Micro Saint

In Arena invece per poter implementare dei tempi di attesa o dei ritardi è necessario l'utilizzo dei blocchi specifici chiamati 'process'.

Come infatti è possibile vedere in figura 2.10 in ogni process è possibile definire:

- l'azione desiderata;
- il tipo di ritardo;
- unità di misura del tempo;
- il tipo di allocazione;
- l'espressione della funzione di densità di probabilità con i relativi parametri.

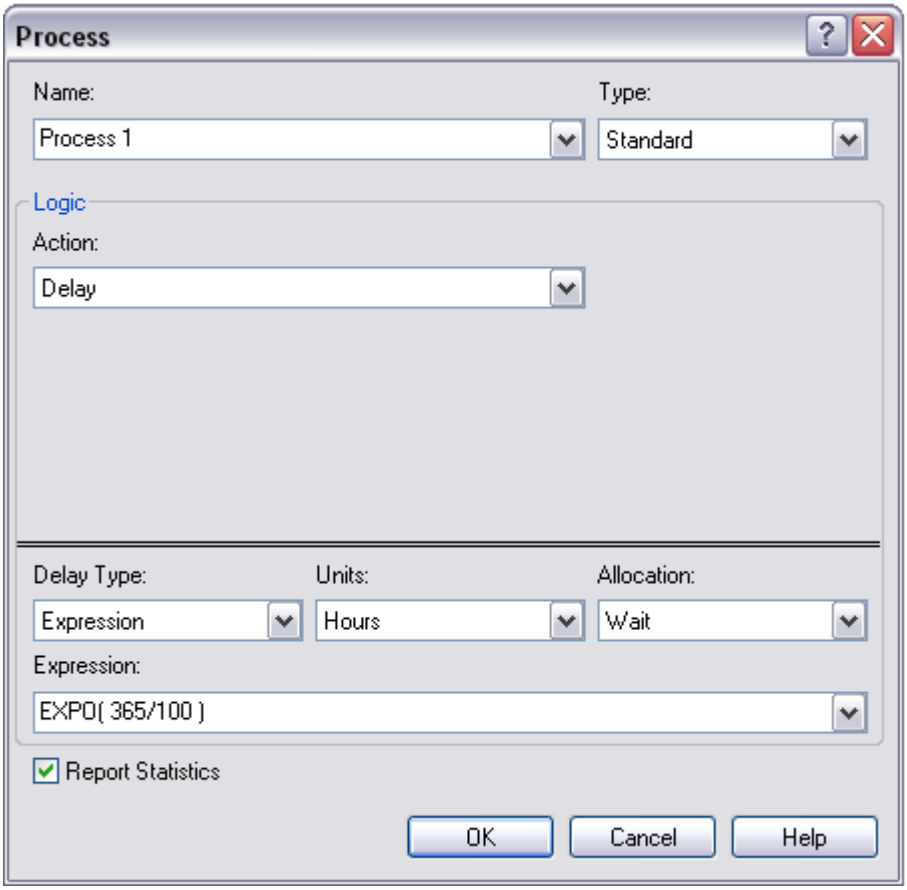

Figura 2.10.Struttura di un modulo process in Arena

2.5.1.Equivalenze per le diverse distribuzioni di probabilità

In Micro Saint i parametri utilizzati per definire le distruzioni di probabilità sono sempre:

- Mean time, ovvero la media;
- Standard deviation, ovvero la deviazione standard.

In Arena invece ogni distribuzione viene definita secondo particolari parametri. Per esempio la distribuzione gamma, utilizzata varie volte nel modello, utilizza due specifici parametri: alfa e beta.

È quindi necessario utilizzare un'equivalenza per trasformare media e deviazione standard nei corrispettivi valori alfa e beta. Le equazioni utilizzate sono state le seguenti:

- *media = alfa/beta*;
- *deviazione standard = alfa/(beta^2).*

Si è presentato un problema analogo anche per la distribuzione uniforme o rettangolare, infatti in Arena i parametri utilizzati sono:

- Minimum, valore minimo;
- Maximum, valore massimo.

2.6.Decisions

La traduzione dei moduli decisionali è abbastanza immediata.

I linguaggi in questo caso sono simili sia per quanto riguarda l'aspetto grafico, entrambi utilizzano i rombi, sia per quanto riguarda la struttura interna di compilazione delle istruzioni.

Solo in certi casi, per poter tradurre in modo corretto il modello originario, è stato necessario introdurre un modulo decide seguito da un separate.

Per esempio i comandi del modulo decisionale della quinta task del modello in Micro Saint riportano:

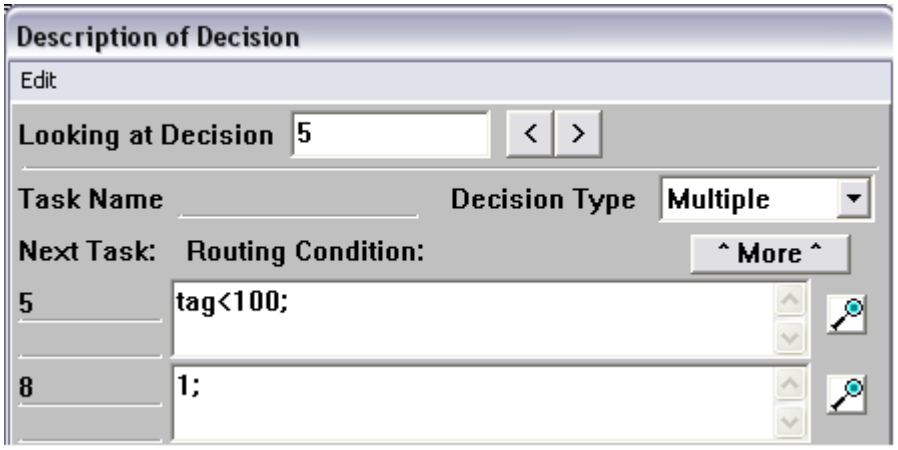

Figura 2.11.Istruzioni della quinta task del modello in Micro Saint

Invece in Arena la prima condizione è stata inserita nel primo modulo, decide 6, mentre la seconda nel modulo successivo, separate 3:

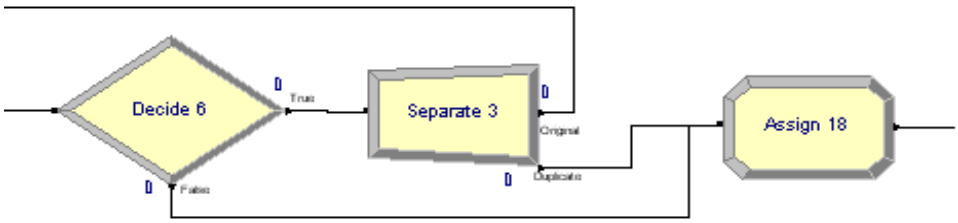

Figura 2.12.Traduzione in Arena del decide della quinta task

Nel decide 6 è stata inserita la condizione *tag < 100* mentre il separate 3 viene utilizzato per ramificare il modello in più parti. Ciò significa che, arrivati in quel punto, ci sarà una biforcazione e che il programma andrà sempre sia nel primo ramo, sia nel secondo.

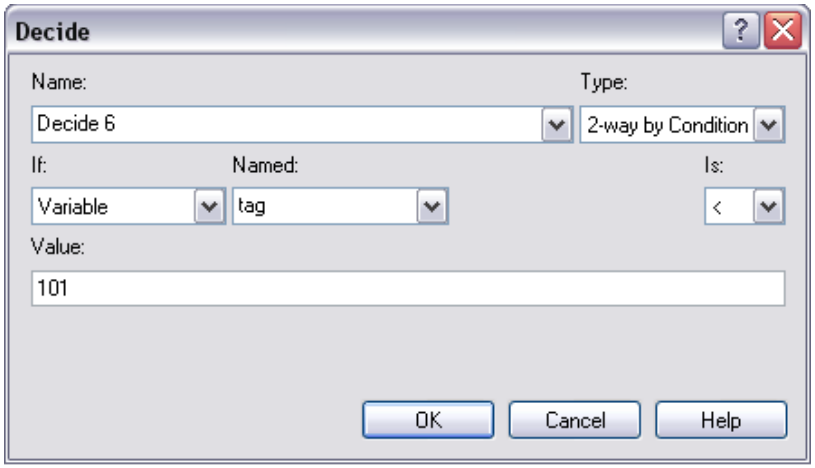

Figura 2.13.Istruzioni presenti nel decide 6, Arena

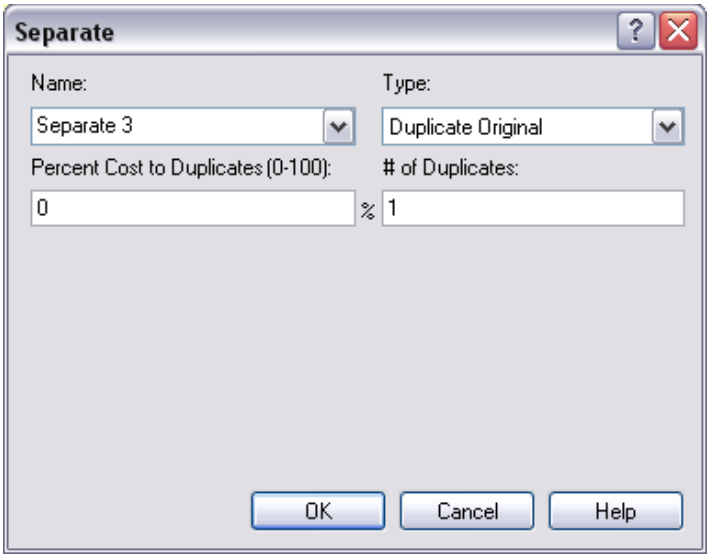

Figura 2.14.istruzioni presenti nel separate 3, Arena

CAPITOLO 3: Il modello in Arena

3.1.Inizio ed introduzione di tutti i parametri utili

Il modello inizia con una task create, necessaria se si vuole creare una nuova simulazione. Subito dopo sono stati inseriti due moduli separate, entrambi con '# of Duplicates' uguale a 1, per poter far confluire il modello in tre diversi rami. Ovvero, subito dopo aver superato quella task, il programma eseguirà in contemporanea le istruzioni che incontrerà nelle tre ramificazioni.

Successivamente troviamo gli assigns 1-6, queste task contengono i valori dei parametri e delle variabili che saranno poi utilizzati nel corso della simulazione.

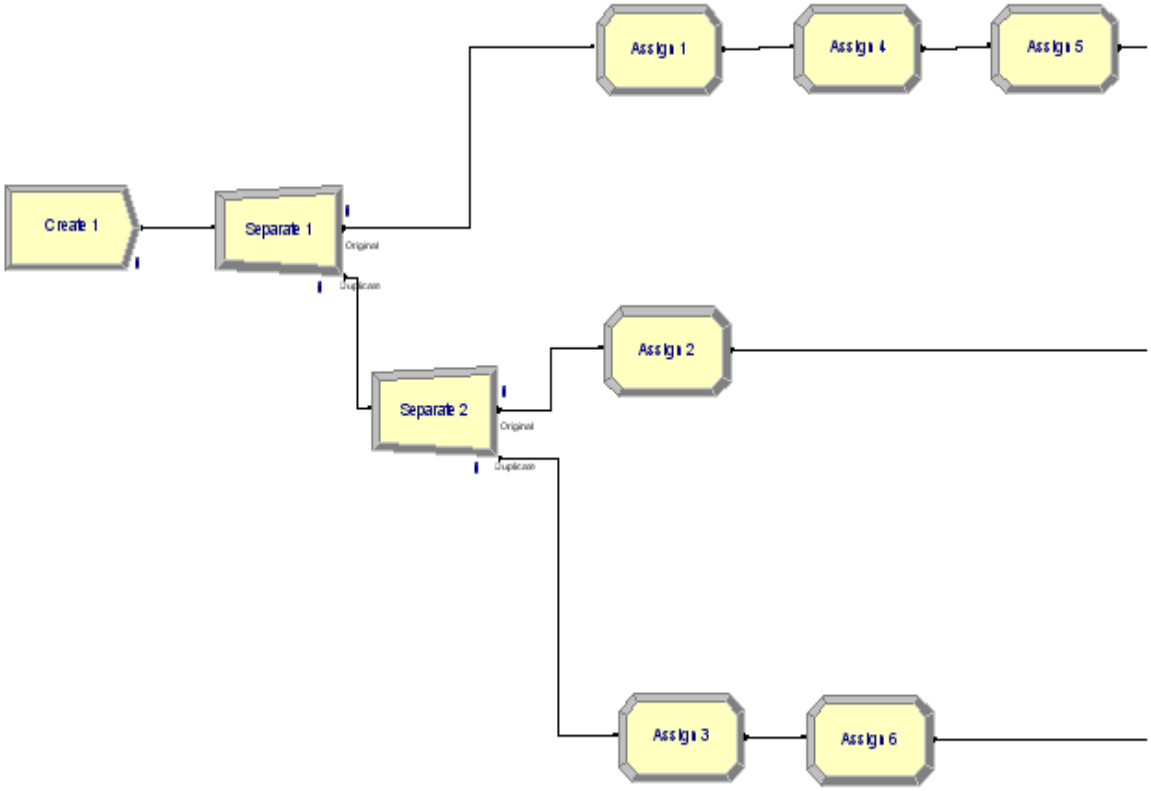

Figura 3.1.Inizio ed introduzione di tutti i parametri utili

3.2.Generatori di domanda per i vari livelli di servizio

La seconda parte del modello viene utilizzata per generare la domanda relativa ai vari livelli di assistenza.

Per poter fare ciò è stata creata negli assigns 7-9 una variabile vettoriale, *xx*, alla quale vengono assegnati valori casuali secondo una distribuzione uniforme.

Attraverso un ciclo *while*, riprodotto in Arena tramite l'utilizzo dei decides 1-3 e del comando *if*, è possibile determinare in quale categoria rientra il valore di *xx*. Grazie infatti ad una variabile semplice, *l*, siamo in grado di etichettare il valore di *xx* in modo da poterne gestire l'assegnazione nei vari centri.

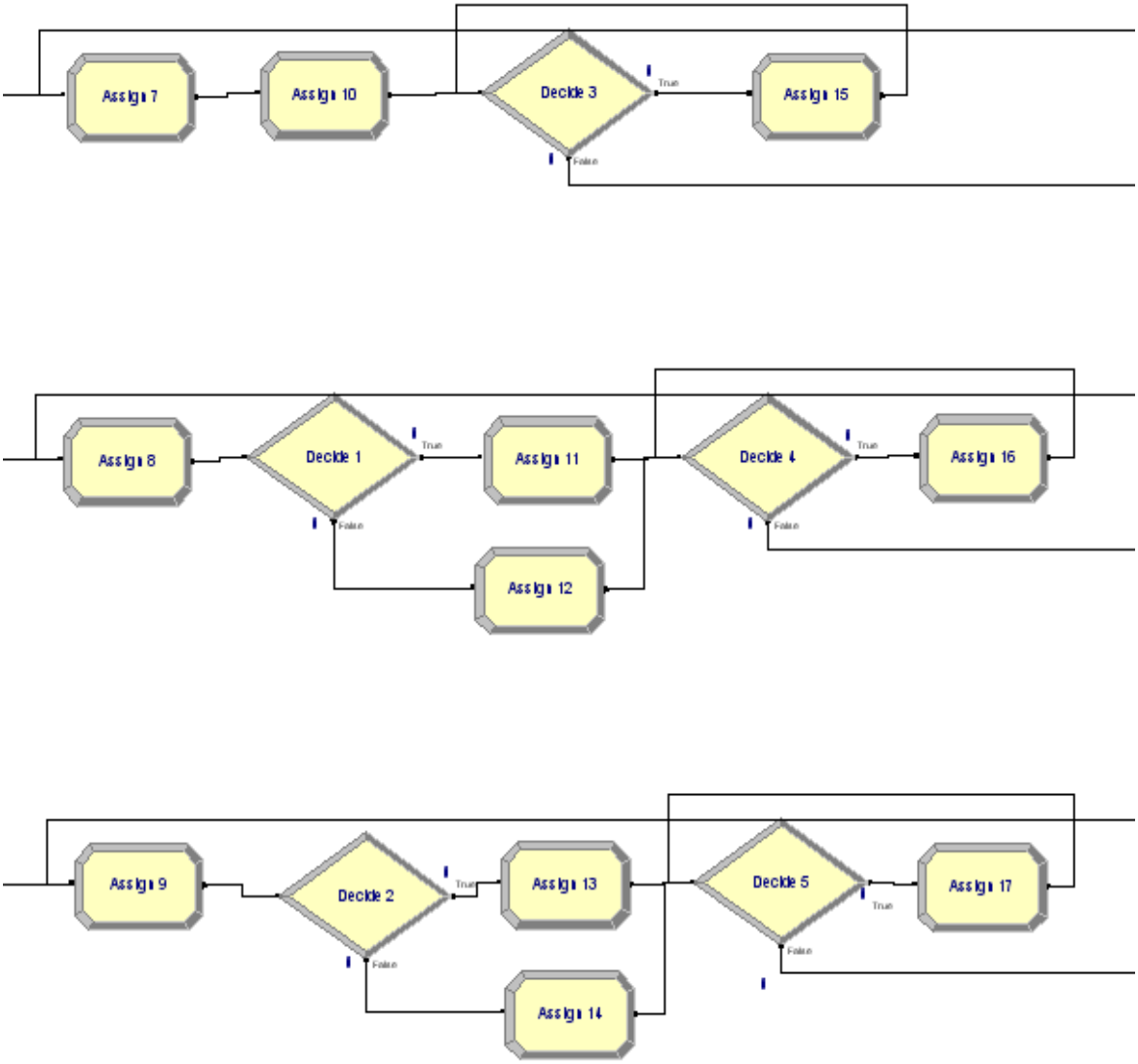

Figura 3.2.Generatori di domanda per i vari livelli di servizio, parte iniziale

I decides 6-8 e i separate 3-5 sono invece utilizzati per poter ripercorrere le precedenti operazioni in modo da continuare a generare numeri casuali e quindi poter simulare nuove richieste per i vari livelli di assistenza.

Sono stati inoltre introdotti dei moduli processes, i numeri 1-3, grazie ai quali possiamo controllare il tempo tra la fine di un ciclo e l'inizio del successivo.

Infine, negli assigns 18-20, è stata definita una nuova variabile semplice, *disp*. Con essa verrà indicata la disponibilità di ogni centro e si potrà quindi capire quando il centro sarà arrivato a saturazione.

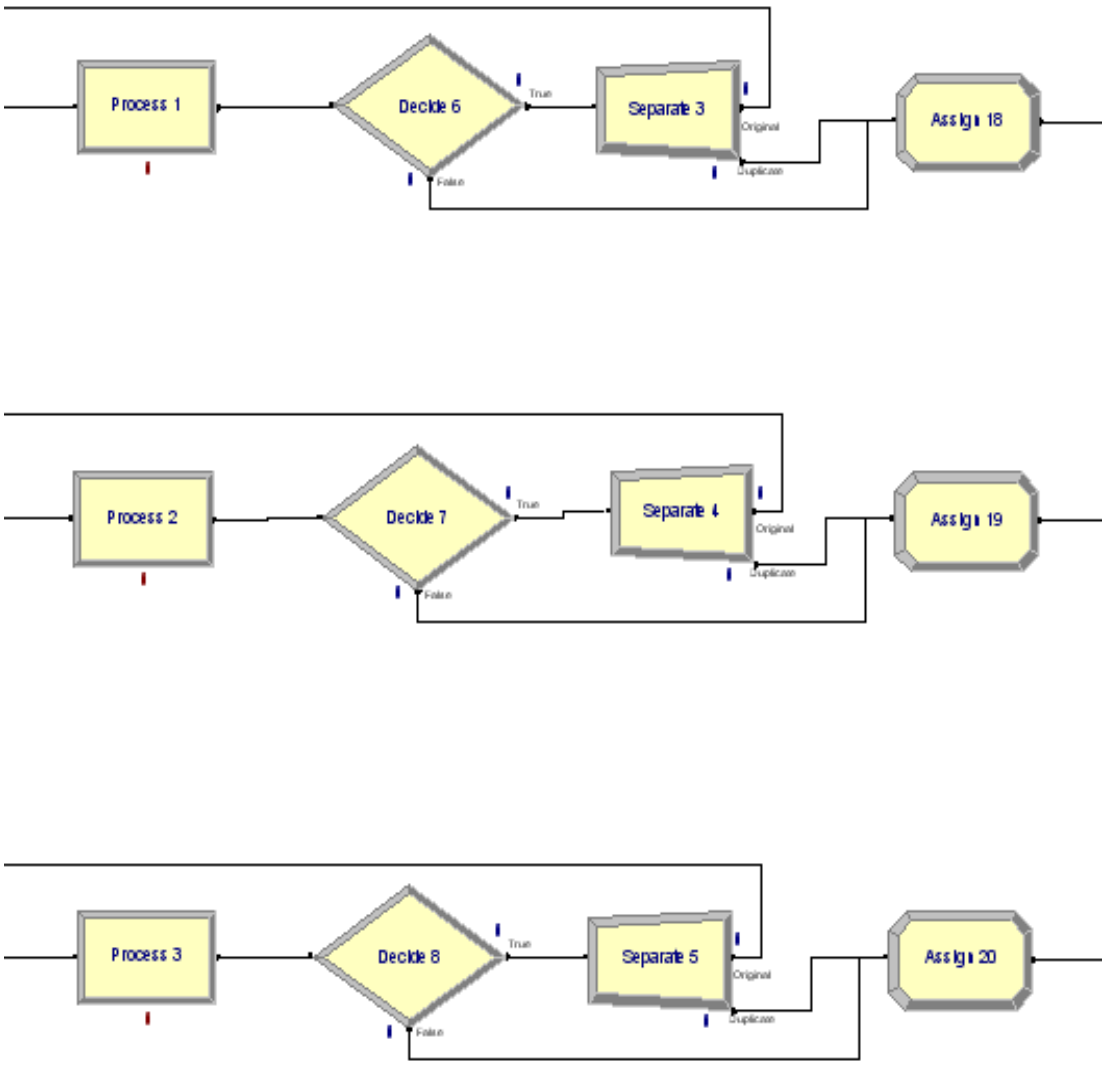

Figura 3.3. Generatori di domanda per i vari livelli di servizio, parte finale

3.3.Smistamento dei diversi casi nei centri più idonei

Inizialmente sono stati utilizzati i decides 9-11 per costruire il secondo ciclo *while* incontrato nella traduzione. Quest'ultimo è abbastanza complesso e per riprodurlo in Arena è stato necessario l'utilizzo di ben 5 moduli. Al suo interno infatti è anche presente un costrutto *if*, decides 12-14 .

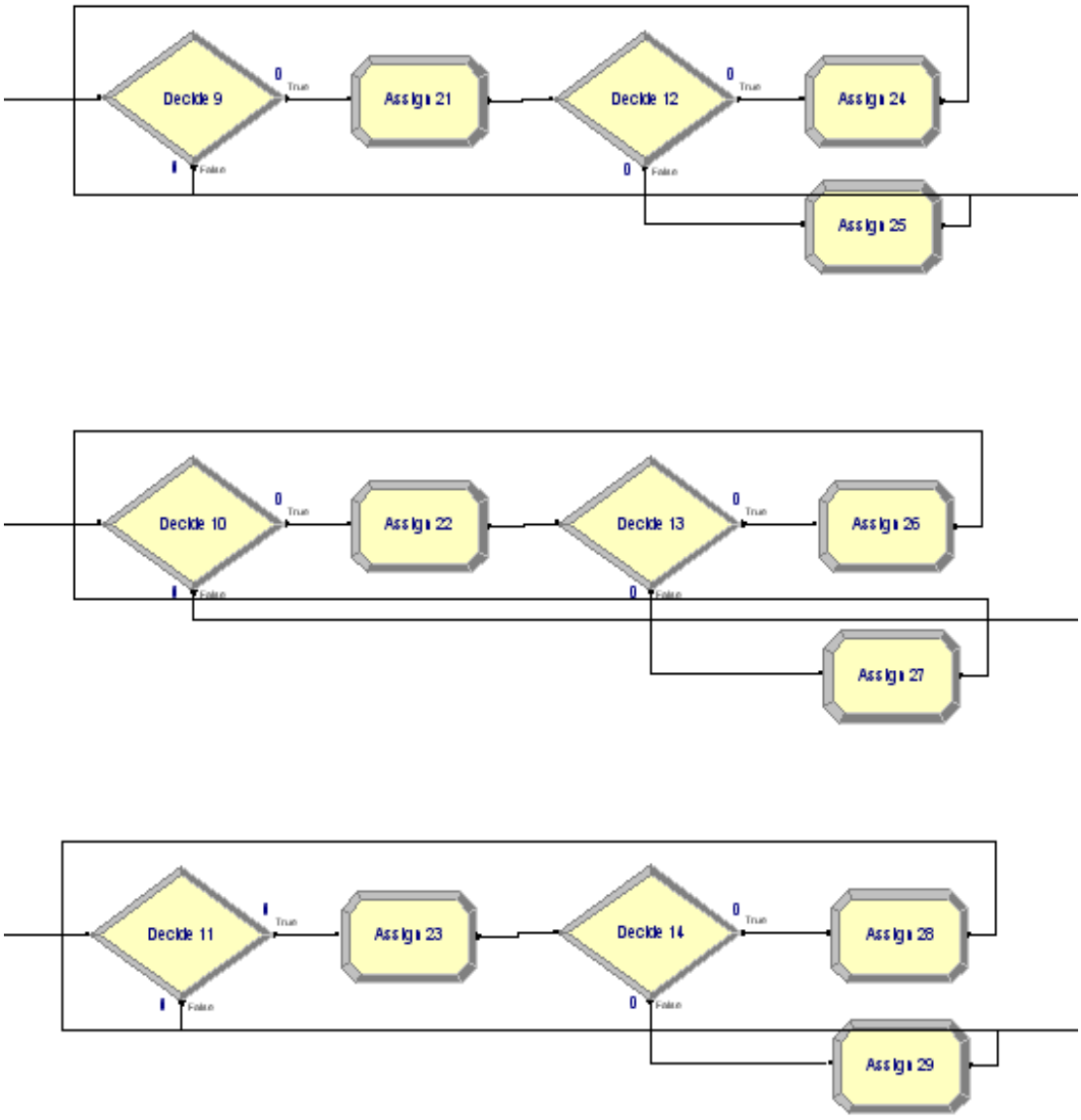

Figura 3.4.Smistamento dei diversi casi nei centri più idonei, parte iniziale

Infine sono stati inseriti due moduli decides per ogni sottoramo, per l'ultimo ne è bastato uno, per poter smistare neo mamme e figli nelle diverse tipologie di centri assistenza.
In questa parte è stato necessario l'utilizzo di un dispose per l'uscita relativa all'else.

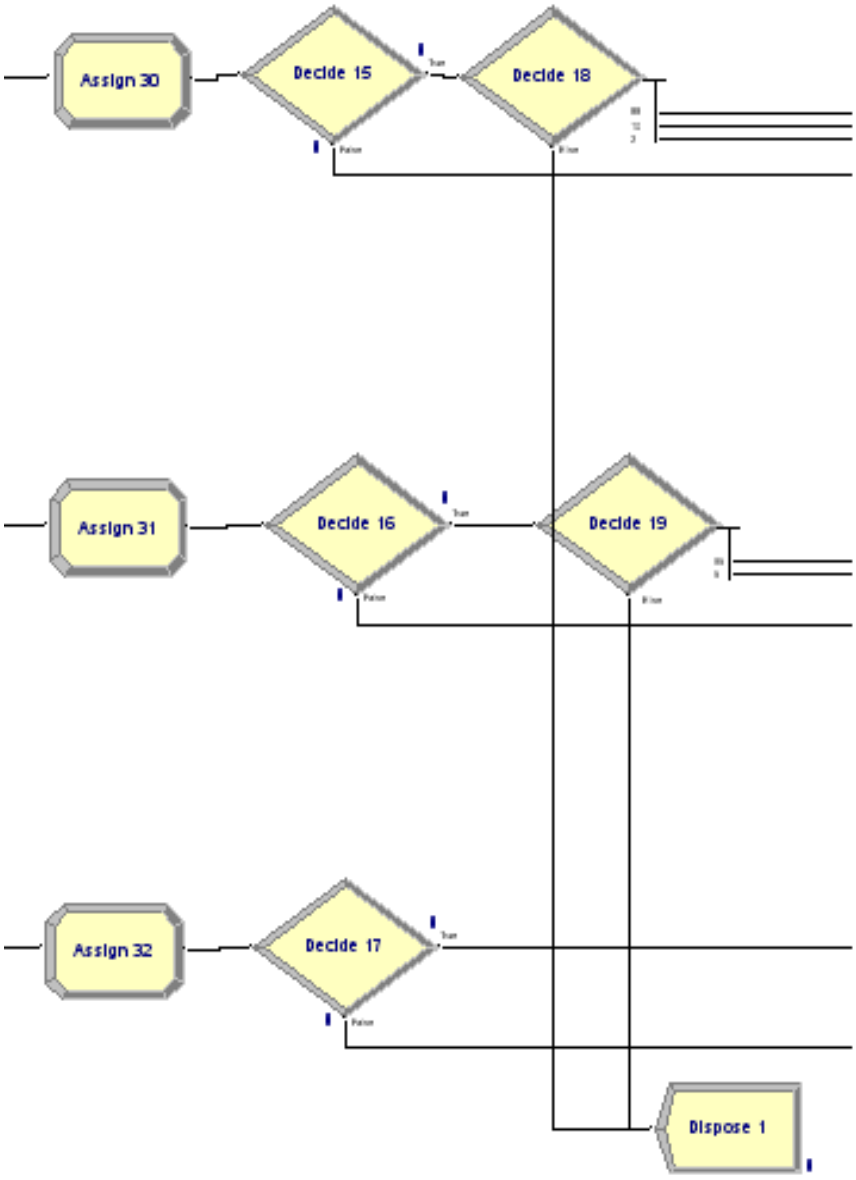

Figura 3.5. Smistamento dei diversi casi nei centri più idonei, parte finale

3.4.Gestione del trasferimento dei neonati

Nel ramo superiore, process 4 e dispose 2, vengono raccolti i neonati e le mamme ammessi nei centri fuori regione.

I rami inferiori invece presentano lo stesso ciclo *while*, già descritto nel paragrafo precedente: condizione principale nei decides 20-21, mentre il costrutto *if* è stato inserito nei decides 22-23.

Alla fine di questa parte di programma i due moduli decisionali, 24-25, possono o far proseguire l'avanzamento verso l'ultima sezione del modello, oppure ricondurre al ramo superiore, process 4 e dispose 2.

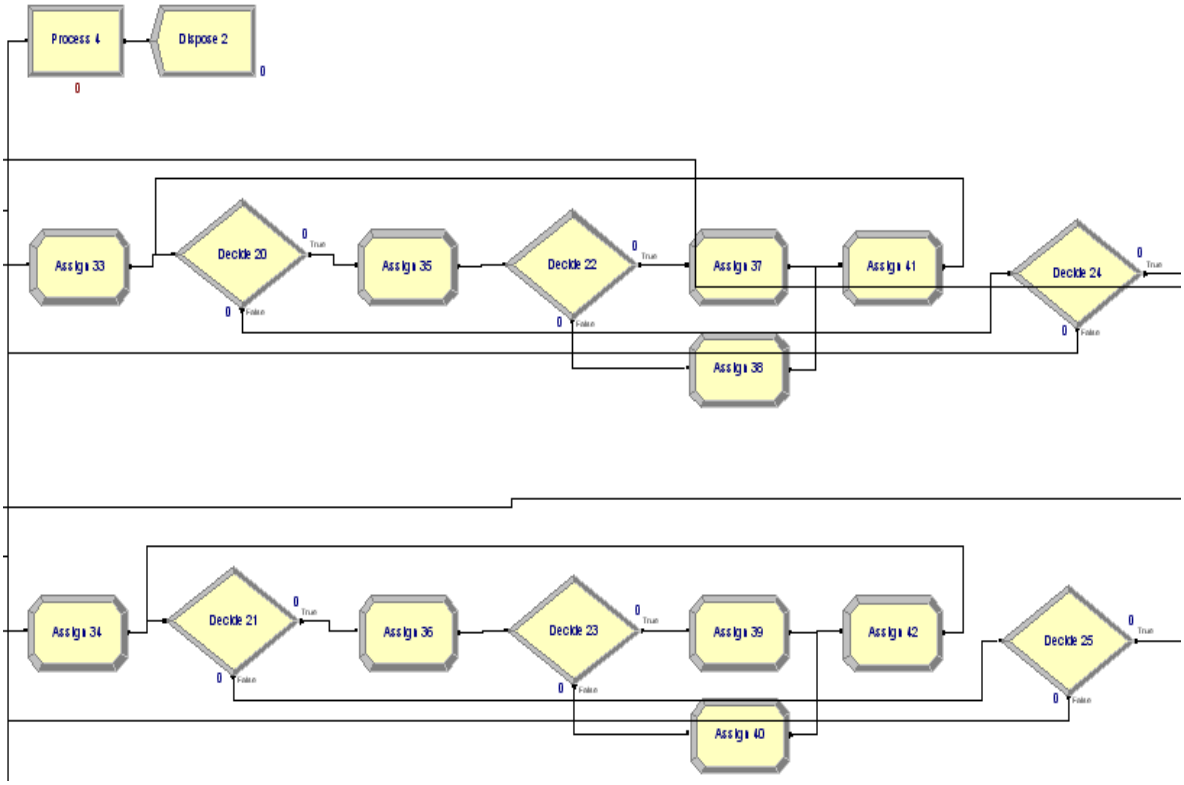

Figura 3.6.Gestione del trasferimento dei neonati

3.5.Accettazione nei vari livelli di assistenza

I moduli iniziali, assigns 43-44, sono funzionali al modello e vengono utilizzati solo per assegnare nuovi valori alle variabili *k* e *car*.

Invece i separates 6-7 hanno lo scopo di ramificare il modello in due parti, entrambi con una probabilità di percorrenza del 100%. In questo modo è possibile simulare sia i posti letto occupati dalle madri, sia quelli occupati dai loro figli.

Le neomamme infatti vengono indirizzate al ramo superiore, moduli 5, 45 e 3; mentre i neonati, a seconda del loro precedente percorso, vengono convogliati o nel ramo centrale, moduli 6, 46 e 4, oppure in quello inferiore, moduli 7, 47 e 5.

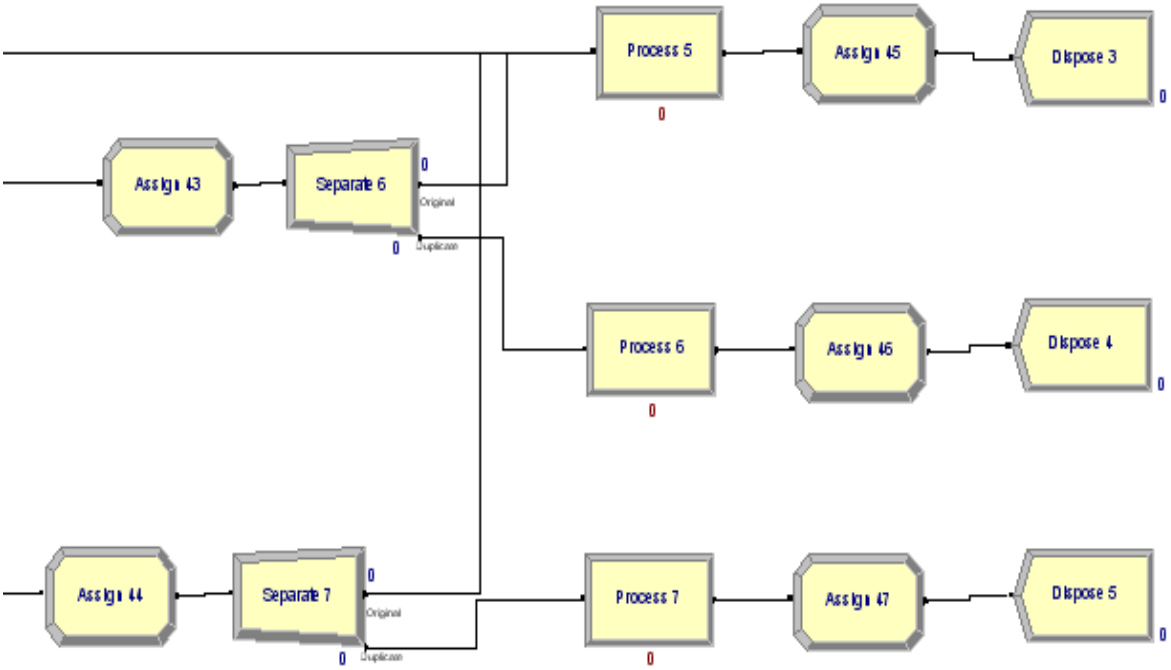

Figura 3.7.Accettazione nei vari livelli di assistenza

CONCLUSIONI

Il modello in Micro Saint, seppure con qualche difficoltà iniziale, è stato totalmente tradotto in ogni sua parte.

Infatti, dopo aver appreso le caratteristiche principali dei due linguaggi, è stato possibile carpirne le differenze e le similitudini. Ogni task è stata analizzata in tutte le sue parti, sono state poi raccolte tutte le istruzioni ed infine, utilizzando gli strumenti messi a disposizione dal linguaggio Arena, è stata completata la traduzione.

Il modello finale non è però utilizzabile in uno scenario reale infatti, già dopo meno di un minuto dal lancio del programma, compare un messaggio d'errore.

La versione Student è infatti limitata e non consente di elaborare un numero elevato di cicli.

BIBLIOGRAFIA

- Facchn P., Ferrante A., Rizzato E., Romanin-Jacur G., Salmaso L., 2010, "Perinatal assistance network planning via simulation", *Journal of Applied Quantitative Methods*, vol.5, n.3, pp.479-485.

- Sito ufficiale Micro Saint (http://www.maad.com/index.pl/micro_saint) 17/08/2011

- Sito ufficiale Arena (http://www.arenasimulation.com/) 18/08/2011

- http://www.alionscience.com/en/Top-Menu-Items/News-Room/Press-Releases/2003-06/AlionAcquiresMicroAnalysisDesignInc?p=1 18/08/2011

- http://phx.corporate-ir.net/phoenix.zhtml?c=196186&p=irol-newsArticle&ID=825787&highlight= 18/08/2011

- http://www.itl.nist.gov/div898/handbook/eda/section3/eda366b.htm 26/08/2011

- Park, Sung Y., Bera, Anil K., 2009, "Maximum entropy autoregressive conditional heteroskedasticity model", *Journal of Econometrics* (Elsevier), pp. 219-230

- http://dimgruppi.ing.unibs.it/tecnologia/esame/PCPB/simulazione%20di%20processi.pdf 18/08/2011

- Tayfur Altiok, Benjamin Melamed, 2007, *Simulation Modeling and Analysis with ARENA*, Academic Press