ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 ΠΡΟΣΟΜΟΙΩΣΗ ΣΥΣΤΗΜΑΤΩΝ

Η προσομοίωση περιλαμβάνει μία μεγάλη συλλογή μεθόδων και εφαρμογών μίμησης της συμπεριφοράς πραγματικών συστημάτων, συνήθως, με χρήση ηλεκτρονικού υπολογιστή και κατάλληλου λογισμικού. Πρόκειται για έναν ιδιαίτερα γενικό όρο επειδή έχει πολλά πεδία εφαρμογών. Σήμερα η προσομοίωση είναι πιο δημοφιλής από ποτέ, σε αναλογία με την ραγδαία ανάπτυξη της τεχνολογίας των υπολογιστών.

Η προσομοίωση, όπως τα περισσότερα εργαλεία ανάλυσης, ασχολείται με συστήματα και μοντέλα τους (πρότυπα). Σύστημα είναι μία διαδικασία που έχει υλοποιηθεί και λειτουργεί ή υπό σχεδίαση, όπως:

- Μία εγκατάσταση παραγωγής με εργαζόμενους, μηχανές, πρώτες ύλες, ενδιάμεσα και έτοιμα προϊόντα, μεταφορικές ταινίες, οχήματα και αποθηκευτικούς χώρους.
- Μία υπηρεσία με πελάτες, εξυπηρετούντες και διακριτές, αυτοματοποιημένες ή μη, λειτουργίες, όπως μηχανές αυτόματης συναλλαγής (ATMs), ταμεία και γραφεία έκδοσης δανείων σε μία τράπεζα.
- Ένα δίκτυο διανομής προϊόντων.
- Ένα κυκλοφοριακό σύστημα, όπως ένα σύστημα εθνικής οδού με κίνηση οχημάτων, κόμβους και διόδια.
- Μία εγκατάσταση χημικής παραγωγής με δεξαμενές, δίκτυα σωληνώσεων και αντιδραστήρες.
- Ένα κατάστημα γρήγορου φαγητού (fast food) με εργαζόμενους διαφόρων κατηγοριών, πελάτες, εξοπλισμό και σύστημα προμήθειας πρώτων υλών.

Τα συστήματα μελετώνται για διάφορους λόγους. Συνήθως μελετώνται για να βελτιωθεί η λειτουργία και απόδοση τους καθώς και για την πρόβλεψη της συμπεριφοράς τους ήδη από την φάση της σχεδίασης. Άλλωστε, είναι συχνά επιθυμητή η ύπαρξη ενός μοντέλου μέσω του οποίου γίνεται μελέτη διαφόρων σεναρίων λειτουργίας, όπως η βλάβη μιας μηχανής σε ένα σύστημα παραγωγής, ή η διαφορετική διάταξη και κατανομή των εργαζομένων μιας υπηρεσίας στις διάφορες διαδικασίες. Η προσομοίωση συχνά βοηθά και στην κατανόηση της λειτουργίας πολύπλοκων συστημάτων, γεγονός που την καθιστά ιδιαίτερα χρήσιμη.

1.2 ΠΡΟΣΟΜΟΙΩΣΗ ΣΤΟΝ ΥΠΟΛΟΓΙΣΤΗ

Ο υπολογιστής είναι ένα αποτελεσματικό εργαλείο προσομοίωσης για μία μεγάλη ποικιλία συστημάτων χρησιμοποιώντας λογισμικό ειδικά σχεδιασμένο να μιμείται τις λειτουργίες είτε τα χαρακτηριστικά τους. Πρακτικά, σε κάθε περίπτωση, αναπτύσσεται ένα μοντέλο του πραγματικού ή υπό ανάπτυξη συστήματος. Έπειτα, εκτελείται ένα σύνολο προσομοιώσεων της λειτουργίας του συστήματος με διάφορα αριθμητικά σενάρια των παραμέτρων του με σκοπό την καλύτερη κατανόηση της συμπεριφοράς του και τον προσδιορισμό των βέλτιστων τιμών των παραμέτρων.

Παρ' ότι η προσομοίωση δεν είναι το μοναδικό εργαλείο ανάλυσης, επιλέγεται όλο και συχνότερα. Ένα μοντέλο προσομοίωσης μπορεί να γίνει όσο πολύπλοκο χρειάζεται, ώστε να αναπαριστά σωστά το σύστημα χωρίς να αναγκάζει τον αναλυτή να κάνει προσεγγίσεις ή απλοποιήσεις που θα αναιρούσαν την εγκυρότητα της μοντελοποίησης.

Η μέθοδος της προσομοίωσης παρουσιάζει και ορισμένα μειονεκτήματα. Το προφανέστερο από αυτά είναι το γεγονός ότι δίνει αποτελέσματα που αφορούν ένα σενάριο λειτουργίας του συστήματος για πεπερασμένο χρονικό ορίζοντα και όχι την κατά μέσον όρο συμπεριφορά του. Πολύ συχνά τα συστήματα έχουν στοχαστικές εισόδους, δηλαδή η συμπεριφορά τους εξαρτάται από διάφορους τυχαίους παράγοντες. Έτσι, κάθε φορά που προσομοιώνουμε ένα σύστημα είναι σαν να εκτελούμε ένα πείραμα τύχης. Αποτέλεσμα αυτού είναι να τίθενται υπό αμφισβήτηση τα αποτελέσματα της προσομοίωσης.

Ωστόσο, ένα καλό μοντέλο προσομοίωσης θα πρέπει να συμπεριφέρεται, κατά το δυνατόν, όπως το πραγματικό σύστημα, δηλαδή να παρουσιάζει ανάλογη τυχαιότητα. Το παραπάνω *μειονέκτημα* είναι λοιπόν αναγκαίο. Πως μπορεί, όμως, κανείς να είναι βέβαιος αν τα αποτελέσματα ενός πειράματος προσομοίωσης αντικατοπτρίζουν τη μέση ή αναμενόμενη συμπεριφορά ενός συστήματος; Σε πολλές περιπτώσεις, υπό τις ίδιες συνθήκες, όσο αυξάνει η διάρκεια της προσομοίωσης (ή ο αριθμός τους), τόσο τα αποτελέσματα συγκλίνουν σε κάποιες μέσες τιμές, παρουσιάζοντας μικρότερη μεταβλητότητα (διασπορά).

Όπως θα παρουσιαστεί και στην παρούσα εργασία μπορούμε να προσδιορίσουμε στατιστικά το πλήθος προσομοιώσεων που απαιτούνται, ή το μέγεθος μιας προσομοίωσης, ώστε να είμαστε πιο σίγουροι για τα αποτελέσματα. Όμως στην πράξη αυτό δεν είναι πάντα ρεαλιστικό, επειδή πολλά συστήματα δουλεύουν για συγκεκριμένο χρόνο και όχι επ' άπειρον, όπως μία τράπεζα, που ανοίγει τις πόρτες τις το πρωί και κλείνει μετά από επτά ώρες.

Συμπερασματικά, απαιτείται ιδιαίτερη προσοχή ως προς τον αριθμό, είτε την διάρκεια των προσομοιώσεων. Μπορεί κανείς να εξαλείψει την τυχαιότητα από ένα μοντέλο, να παίρνει σαφή αποτελέσματα χωρίς, όμως, να είναι βέβαιο ότι θα είναι και έγκυρα. Είναι προτιμότερο να απαντάμε προσεγγιστικά για το πραγματικό σύστημα, παρά να είμαστε ακριβείς αλλά για λάθος πρόβλημα. Για τον σκοπό αυτό, οι εκτιμήσεις που προκύπτουν από ένα ή περισσότερα πειράματα προσομοίωσης αναλύονται με στατιστικά εργαλεία και παρουσιάζονται με τη μορφή των λεγόμενων *διαστημάτων εμπιστοσύνης*.

1.3 ΕΦΑΡΜΟΓΕΣ

Η παρούσα εργασία πραγματεύεται την προσομοίωση συστημάτων παραγωγής και εξυπηρέτησης με την βοήθεια του προγράμματος Arena®. Μελετώνται συστήματα αναμονής που εξετάζονται στα εργαστήρια του μαθήματος «Προσομοίωση» που διδάσκεται στο τμήμα *Μηχανικών Παραγωγής και Διοίκησης* του Πολυτεχνείου Κρήτης, καθώς και πιο σύνθετα συστήματα. Βασικός σκοπός της εργασίας είναι το παρόν έντυπο να χρησιμοποιηθεί ως εγχειρίδιο εκμάθησης της Arena και διδασκαλίας των εργαστηριακών ασκήσεων.

Στην παρούσα εργασία εξετάζονται τα εξής προβλήματα:

- Προσομοίωση συστημάτων αναμονής (queuing systems) και σύγκριση των αποτελεσμάτων με την αντίστοιχη θεωρία.

- Στατιστικός προσδιορισμός του πλήθους προσομοιώσεων που απαιτούνται, ώστε το σχετικό σφάλμα εκτίμησης ενός μέτρου απόδοσης να είναι, με μεγάλη πιθανότητα, μικρότερο από ένα άνω όριο.

- Μελέτη γραμμών παραγωγής με ενδιάμεσες αποθήκες πεπερασμένης χωρητικότητας.
- Εφαρμογή της μεθόδου Ανάλυσης Διαταραχών (perturbation analysis) [4] για την βελτιστοποίηση των ρυθμών αφίξεων και εξυπηρέτησης σε ένα σύστημα αναμονής. Η μέθοδος Ανάλυσης Διαταραχών αναπτύχθηκε περί το 1980 και αποτελεί αντικείμενο έρευνας μέχρι σήμερα.
- Προσομοίωση συστημάτων συναρμολόγησης (assembly systems).

2. ΕΙΣΑΓΩΓΗ ΣΤΗΝ ARENA

2.1 ΓΕΝΙΚΑ

Η Arena είναι ένα περιβάλλον προσομοίωσης συστημάτων παραγωγής, στα οποία παρατηρείται κίνηση, αποθήκευση και κατεργασία κομματιών, συστημάτων εξυπηρέτησης, στα οποία οι εξυπηρετούμενοι σχηματίζουν ουρές μπροστά από σημεία εξυπηρέτησης, ή εν γένει συστημάτων ροής τα οποία διαχειρίζονται τηλεφωνικά σήματα, εντολές εκτέλεσης στο υπολογιστή, κ.λπ.. Αναπτύχθηκε από την αμερικάνικη εταιρία *Systems Modeling Corporation*[1] και λειτουργεί σε περιβάλλον Microsoft Windows. Τα μοντέλα (πρότυπα) στα οποία αναφέρεται η παρούσα εργασία αναπτύχθηκαν στην ακαδημαϊκή έκδοση Arena® 4.0.

Ο χρήστης έχει την δυνατότητα να προσομοιώνει συστήματα χωρίς να γράφει ο ίδιος κώδικα προγραμματισμού. Αντίθετα, καλείται να επιλέξει από έναν μεγάλο αριθμό μονάδων σχεδίασης (τα λεγόμενα **modules**, **elements** και **blocks)** τις κατάλληλες και να τις συνδέσει μεταξύ τους ώστε να αναπαριστούν το σύστημα Καθώς ο χρήστης συνθέτει το μοντέλο του συστήματος, η Arena το μετατρέπει αυτόματα σε γλώσσα προσομοίωσης SIMAN.

Στην Εικόνα 2.1.1 παρουσιάζεται το περιβάλλον της Arena. Το αριστερό τμήμα του παραθύρου ονομάζεται Project Bar και χωρίζεται σε δύο περιοχές:

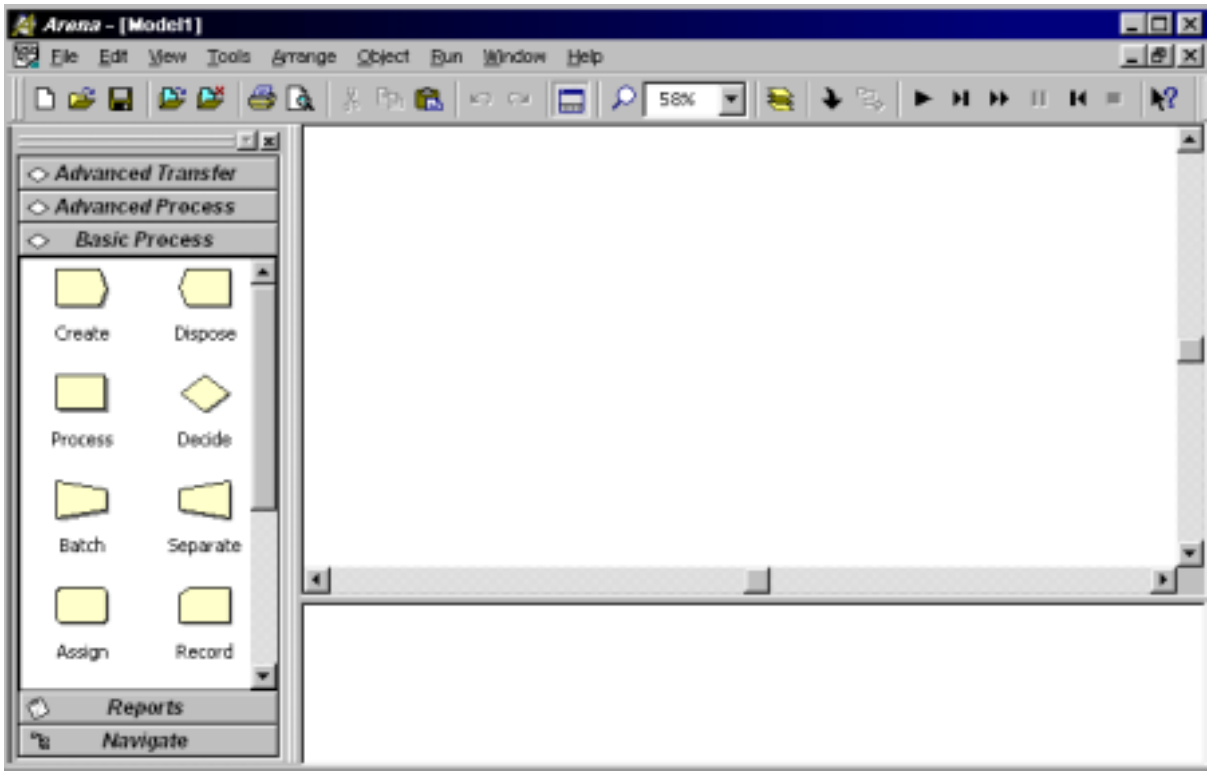
- Τις ομάδες εργαλείων (templates/ panels). Αυτές εισάγονται και εξάγονται με τα εικονίδια “Attach...” και “Detach Template”  αντίστοιχα. Υπάρχουν οι εξής ομάδες:

  - Basic Process (.tpo)
  - Advanced Process (.tpo)
  - Advanced Transfer (.tpo)
  - Blocks (.tpo)
  - Elements (.tpo)
  - UltArena (.tpo)

  Κατά την ανάπτυξη ενός μοντέλου, δεν είναι απαραίτητο να χρησιμοποιούνται ταυτόχρονα και οι έξι ομάδες εργαλείων.

- Τα πλαίσια *Reports* και *Navigate*. Από το πλαίσιο Reports μπορεί κανείς να έχει πρόσβαση στα αποτελέσματα μιας προσομοίωσης, ενώ στο πλαίσιο *Navigate* παρουσιάζονται οι διάφορες όψεις ενός ομοιώματος (βλ. Παράρτ. 2).

[1] Η εταιρία έχει εξαγορασθεί από την *Rockwell Software* με διεύθυνση στο Internet: http://www.arenasimulation.com/



Εικόνα 2.1.1 Το περιβάλλον της Arena

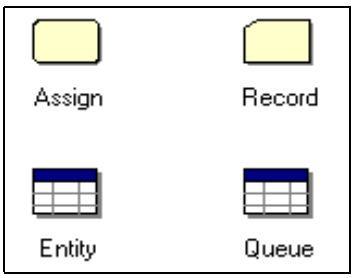
2.1.1 Σχεδίαση με modules

Όπως προαναφέρθηκε, κάθε μοντέλο αναπαρίσταται από ένα σύνολο μονάδων σχεδίασης (modules, blocks και elements), τα οποία εκτελούν επί μέρους, αλλά αλληλένδετες λειτουργίες. Στο εξής, για λόγους συντομίας, τα blocks και elements συχνά θα αναφέρονται ως modules.

Τα modules είναι ομαδοποιημένα σύμφωνα με τη λειτουργία και την πολυπλοκότητα τους στις έξι ομάδες εργαλείων που προαναφέρθηκαν. Υπάρχουν δύο ειδών modules· τα Flowchart(διαγράμματος ροής)[2] και τα Data(δεδομένων, βλ. Εικόνα 2.1.2). Εισάγοντας τα πρώτα στην περιοχή σχεδίασης, δημιουργούμε μία αναπαράσταση του μοντέλου που εξελίσσεται στο χρόνο. Στα Data modules ορίζονται οι μεταβλητές και οι παράμετροι του μοντέλου με τις αρχικές τους τιμές.

Στην κατηγορία Basic Process ανήκουν modules που προσομοιώνουν βασικές λειτουργίες, όπως την άφιξη ενός πελάτη σε ένα σταθμό εξυπηρέτησης (βλ. Παρ.2.2). Στην δεύτερη κατηγορία (Advanced Process) ανήκουν modules που προσομοιώνουν πιο στοιχειώδεις λειτουργίες, όπως την επισκευή μιας μηχανής (αυξημένη λεπτομέρια σχεδίασης). Στην τρίτη κατηγορία (Advanced Transfer) ανήκουν modules που προσομοιώνουν μεταφορές μέσα σε συστήματα, όπως την μεταφορά κομματιών με μεταφορικές ταινίες σε ένα εργοστάσιο.

[2] Ο όρος *Διάγραμμα Ροής* χρησιμοποιείται για να περιγράψει την αλληλουχία των ενεργειών κατά την λειτουργία του μοντέλου (π.χ. άφιξη, εξυπηρέτηση, αναχώρηση πελάτη από μία τράπεζα).



Εικόνα 2.1.2 Οι δύο τύποι module. Flowchart (επάνω) και Data (κάτω)

Όσο λεπτομερέστερη πρέπει να γίνει η προσομοίωση, τόσο πιο εξειδικευμένα modules χρειάζονται. Τα ***blocks*** και ***elements*** δίνουν τη μεγαλύτερη δυνατή λεπτομέρεια σχεδίασης[3] καθώς εκτελούν τις στοιχειωδέστερες λειτουργίες που μπορεί να υπάρξουν. Στην επόμενη παράγραφο εξετάζονται ορισμένα βασικά modules κατά την ανάπτυξη ενός μοντέλου M/M/1.

2.2 ΤΟ ΣΥΣΤΗΜΑ M/M/1

Πρόβλημα

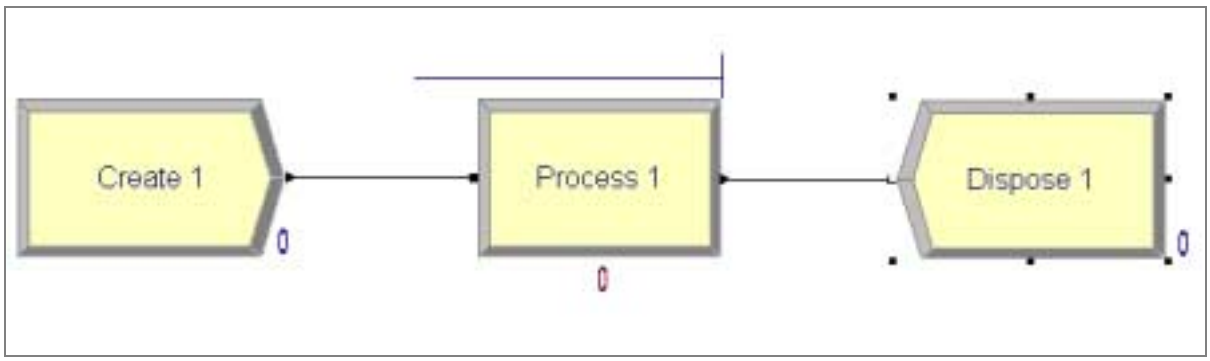
Πελάτες φθάνουν σε ένα γραφείο για να εξυπηρετηθούν. Στο γραφείο υπάρχει ένας εργαζόμενος (εξυπηρετών). Υποθέτουμε ότι το σύστημα μπορεί να δεχθεί άπειρους πελάτες στην ουρά αναμονής. Οι αφίξεις και εξυπηρετήσεις των πελατών γίνονται κατά Poisson (βλ. Παράρτ. 1). Το σύστημα αυτό είναι το απλούστερο της θεωρίας ουρών αναμονής και συμβολίζεται M/M/1. Ζητούμε το μέσο πλήθος πελατών στην ουρά αναμονής, το οποίο στο εξής θα συμβολίζουμε N_q.

2.2.1 Μοντελοποίηση

Για την ανάπτυξη ενός μοντέλου στην Arena αρχικά περιγράφουμε την ακολουθία των δραστηριοτήτων που απαιτούνται για την κίνηση μίας οντότητας (πελάτη, κομματιού κ.λπ.) μέσα από το σύστημα που εξετάζουμε. Για το σύστημα M/M/1 οι δραστηριότητες είναι οι εξής: άφιξη πελάτη, είσοδος πελάτη στην ουρά, δέσμευση του εξυπηρετούντος, τέλος εξυπηρέτησης πελάτη, αποδέσμευση εξυπηρετούντος και έξοδος. Έπειτα εισάγουμε τα κατάλληλα modules για την αναπαράσταση των δραστηριοτήτων (γεγονότα).

Ένα μοντέλο M/M/1 παρουσιάζεται στην Εικόνα 2.2.1. Όταν ένας πελάτης φθάσει στο σύστημα μπαίνει στην ουρά αναμονής. Μόλις έρθει η σειρά του, δεσμεύει τον υπάλληλο εξυπηρέτησης (Seize), εξυπηρετείται (Delay) αποδεσμεύει τον υπάλληλο και φεύγει. Οι αφίξεις πελατών προσομοιώνονται με ένα *Create* module. Η τοποθέτηση των πελατών στην ουρά, η δέσμευση του εργαζόμενου, η εξυπηρέτηση και η αποδέσμευση προσομοιώνονται από κοινού χρησιμοποιώντας ένα *Process* module. Οι πελάτες φεύγουν από το σύστημα από ένα *Dispose* module. Στην συνέχεια αναπτύσσονται βήμα-βήμα τα μέρη του μοντέλου M/M/1.

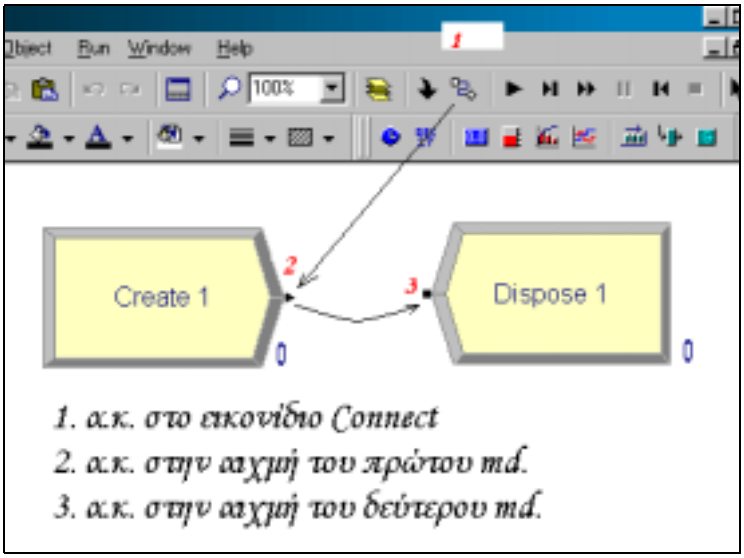
[3] Στην ακαδημαϊκή έκδοση του προγράμματος τα modules δεν επιδέχονται τροποποιήσεων από τον χρήστη. Στην επαγγελματική έκδοση ο αναλυτής έχει την δυνατότητα να κατασκευάζει τα δικά του modules.



Εικόνα 2.2.1 Το μοντέλο M/M/1

Σύνδεση των Modules

Αν εισάγουμε ένα module και αμέσως μετά ένα άλλο δεξιά του, η Arena τα συνδέει αυτόματα. Εναλλακτικά, χρησιμοποιούμε το εικονίδιο *Connect* [4]. Αν, για παράδειγμα, έχουμε εισάγει στην περιοχή σχεδίασης δύο modules και θέλουμε να τα συνδέσουμε ώστε οι οντότητες να μεταφέρονται από το πρώτο στο δεύτερο, χρησιμοποιούμε το εικονίδιο *Connect* και επιλέγουμε πρώτα το τέλος του πρώτου και έπειτα την αρχή του δεύτερου, όπως φαίνεται στην Εικόνα 2.2.2.

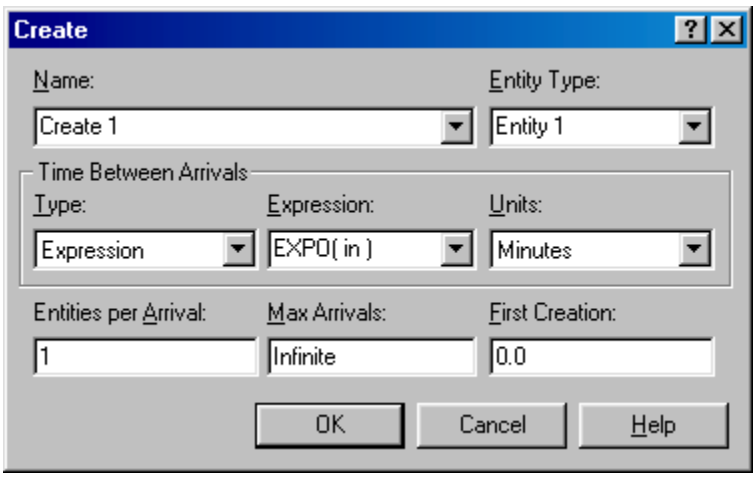


Εικόνα 2.2.2 Σύνδεση modules με το εικονίδιο Connect

Αφίξεις – Το *Create* module

Όπως προαναφέρθηκε, το *Create* module χρησιμοποιείται για τη μοντελοποίηση των αφίξεων. Στην Εικόνα 2.2.3 παρουσιάζεται το πλαίσιο διαμόρφωσης του.

[4] Υπάρχουν, παρ' όλα αυτά, και άλλοι τρόποι σύνδεσης που θα μπορούσαν να χαρακτηριστούν ως έμμεσοι. Για παράδειγμα, υπάρχουν blocks στα οποία ο προορισμός της οντότητας μπορεί να δηλωθεί εναλλακτικά μέσα στο ίδιο το block (στην περιοχή *Next Label*), π.χ. σε ένα *Queue* block (βλ. Παρ.3.6.2).



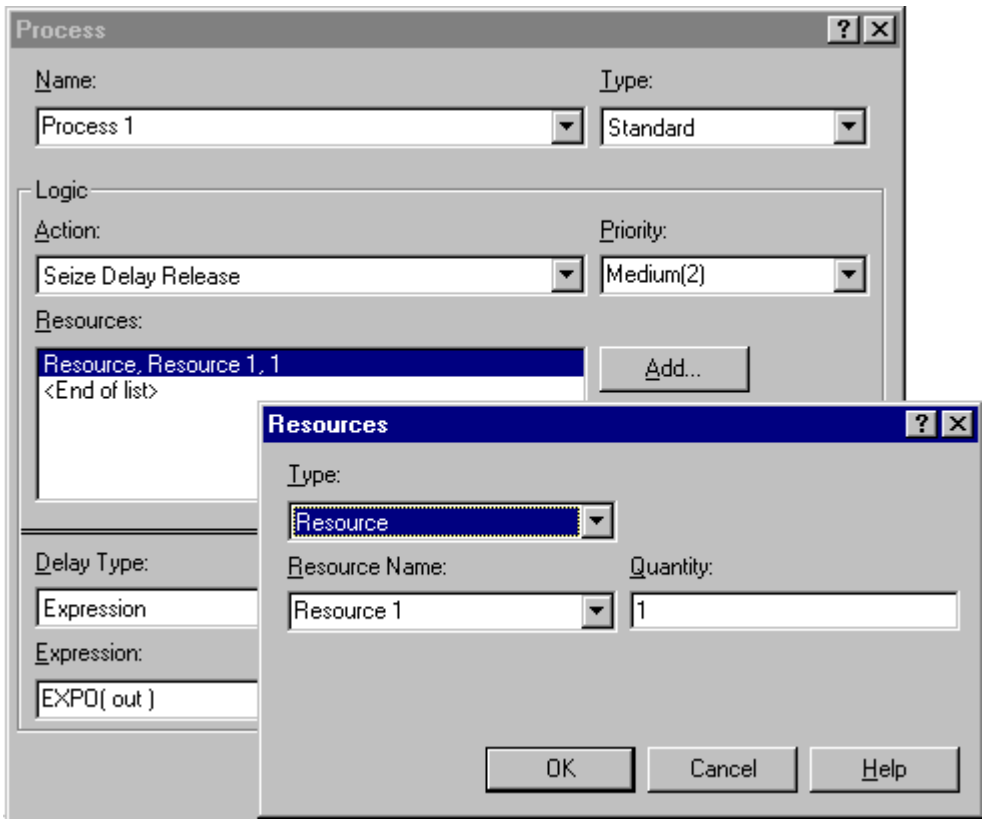
Εικόνα 2.2.3 Το πλαίσιο διαμόρφωσης του Create module

Name	Όνομα του module.
Entity Type	Όνομα (είδος) οντότητας που δημιουργείται.
Time Between Arrivals	Ενδοαφιξιακός χρόνος
Type	Expression
Expression	Expo(in). Εκθετική κατανομή με μέσο *in* (βλ. Παράρτ.1)
Units	Μονάδα χρόνου μεταξύ των αφίξεων (λεπτά).
Entities per Arrival	Αριθμός οντοτήτων που δημιουργούνται ταυτόχρονα.
Max Arrivals	Μέγιστος αριθμός οντοτήτων προς δημιουργία. Όταν δεν συμπληρωθεί προεπιλέγεται το άπειρο.
First Creation	Χρονική στιγμή που "γεννιέται" ο πρώτος πελάτης. Όταν δεν συμπληρωθεί προεπιλέγεται το 0.

Πίνακας 2.1 Συμπλήρωση του Create module

Εξυπηρέτηση Πελατών – Το *Process* module

Όταν ένας νέος πελάτης φθάσει στο σύστημα κατευθύνεται στο *Process* module. Εκεί είναι πιθανό να συμβούν δύο γεγονότα. Αν η ουρά αναμονής είναι άδεια, ο πελάτης πηγαίνει (ακαριαία) στην εξυπηρέτηση διαφορετικά, όταν ο εξυπηρετητής είναι απασχολημένος ο πελάτης περιμένει στην ουρά. Η πληροφορία που απαιτείται να δοθεί στο πλαίσιο διαμόρφωσης του *Process* module είναι αυτή του Πίνακα 2.2. Κάθε Process module περιλαμβάνει μία άπειρη ουρά. Έτσι, το module *Process 1* συνοδεύεται από την ουρά *Process 1.Queue*



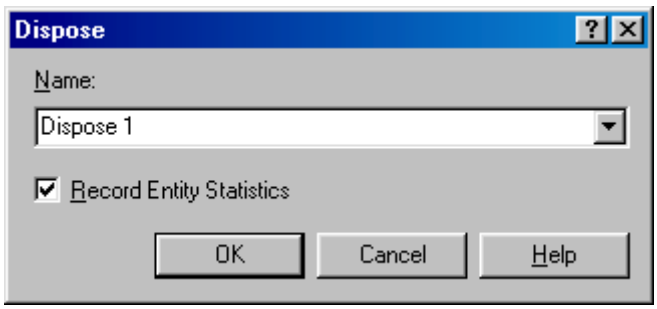
Εικόνα 2.2.4 Διαμόρφωση του Process module για την εξυπηρέτηση

`Name`	Όνομα του module (προεπιλογή)
`Type`	Standard (η γενική περίπτωση)
`Logic`	
`Action`	Seize Delay Release (δέσμευση πόρου, καθυστέρηση, αποδέσμευση πόρου)
`Resources`	Επίλεξε *Add* και όρισε τον πόρο εξυπηρέτησης
`Delay Type`	Expression
`Units`	Minutes. Μονάδα χρόνου είναι το λεπτό
`Allocation`	Value Added (η γενική περίπτωση)
`Expression`	Expo (out). Εκθετικά κατανεμημένος χρόνος εξυπηρέτησης με μέσο *out*. out: Μεταβλητή που ορίζεται στο Data module *Variable* της κατηγορίας BasicProcess

Πίνακας 2.2 Η πληροφορία που δίνεται στο Process module

Αναχώρηση πελατών

Κάθε πελάτης που εξυπηρετείται φεύγει. Αναχωρήσεις μοντελοποιούνται με ένα *Dispose* module.

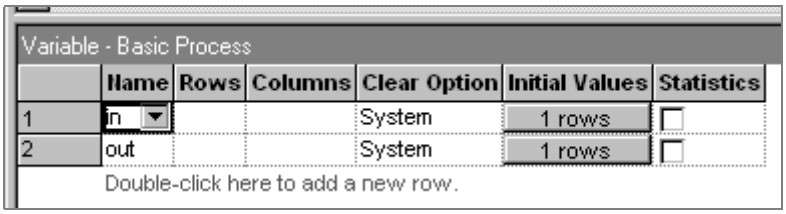


Εικόνα 2.2.5 Dispose. Η έξοδος του μοντέλου M/M/1

Μεταβλητές και Μέτρα Απόδοσης

Η Arena υπολογίζει αυτόματα μερικά μέτρα απόδοσης όπως τον μέσο χρόνο αναμονής στην ουρά (T_q) και το μέσο μέγεθος της (N_q). Είναι, παρ' όλα αυτά, πιθανό ο χρήστης να επιθυμεί να ορίσει και άλλα μέτρα απόδοσης. Για παράδειγμα, η παραγωγικότητα ενός συστήματος (TH) είναι ένα μέτρο απόδοσης που συχνά μας ενδιαφέρει, αλλά δεν υπολογίζεται αυτόματα (βλ. Παρ.3.1.2).

Στην συνέχεια ορίζουμε τις παραμέτρους του μοντέλου. Αυτές είναι ο μέσος της εκθετικής κατανομής του χρόνου μεταξύ δύο διαδοχικών αφίξεων (in) και ο μέσος της εκθετικής κατανομής της διάρκειας μιας εξυπηρέτησης (out). Ο φυσικός χώρος όπου ορίζονται οι μεταβλητές και οι παράμετροι είναι το *Variable* module της ομάδας *Basic Process*. Όπως φαίνεται και στην Εικόνα 2.2.6, εισάγουμε **αρχικές τιμές** στις μεταβλητές από το πλαίσιο Initial Values (in=1 και out=0.8), διαφορετικά προεπιλέγεται το 0.



Εικόνα 2.2.6 Ορισμός των ρυθμών αφίξεων και εξυπηρετήσεων

Προσοχή πρέπει να δίνεται στο πλαίσιο *Clear Option*. Υπάρχουν τρεις επιλογές:

1. *None*. Η τιμή της μεταβλητής δεν αρχικοποιείται παρά μόνο στην αρχή της εκτέλεσης του προγράμματος, δηλαδή κάθε φορά που πατάμε το εικονίδιο *Run*
2. *Statistics*. Αρχικοποίηση της μεταβλητής όταν αρχικοποιούνται οι στατιστικές
3. *System*. Αρχικοποίηση με το σύστημα

Εκτέλεση

Πριν την εκτέλεση κάθε προγράμματος κάνουμε τις τελικές ρυθμίσεις από ***Run/Setup/Replication Parameters***, εισάγοντας πληροφορίες όπως αυτές του Πίνακα 2.3.

`Number of Replications (NREP)`	Αριθμός προσομοιώσεων
`Replication Length`	Διάρκεια μίας προσομοίωσης
`Base Time Units`	Μονάδα μέτρησης του χρόνου για το ρολόι της προσομοίωσης
`Initialize Between Replications`	Αρχικοποίηση μεταξύ των προσομοιώσεων. Εφαρμόζεται μόνο όταν ο αριθμός των προσομοιώσεων είναι > 1.

Πίνακας 2.3 Ορισμένες βασικές παράμετροι ρυθμίζονται από Run/Setup

Στο πλαίσιο *Initialize Between Replications* υπάρχουν τέσσερις επιλογές, που είναι:

1η Επιλογή: Initialize System (yes), Initialize Statistics (yes)
Εκτελούνται στατιστικά ανεξάρτητες προσομοιώσεις. Κάθε προσομοίωση ξεκινάει με το σύστημα άδειο την χρονική στιγμή t=0 και διαρκεί τον ίδιο χρόνο[5].

2η Επιλογή: Initialize System (yes), Initialize Statistics (no)
Στατιστικά ανεξάρτητες προσομοιώσεις. Κάθε προσομοίωση ξεκινά με το σύστημα άδειο την χρονική στιγμή t=0 και τρέχει για τον ίδιο χρόνο. Οι αναφορές είναι προσθετικές. Έτσι, η αναφορά 2 θα περιλαμβάνει τις στατιστικές των δύο πρώτων προσομοιώσεων, η αναφορά 3 των τριών πρώτων κ.ο.κ.

3η Επιλογή: Initialize System (no), Initialize Statistics (yes)
Εδώ ο χρόνος είναι προσθετικός. Έτσι, αν η κάθε προσομοίωση διαρκεί 5000 λεπτά, η δεύτερη προσομοίωση θα τελειώσει την χρονική στιγμή $TFIN_2$=10000, η τρίτη την στιγμή $TFIN_3$=15000 κ.λπ. Κάθε λειτουργία που δεν ολοκληρώθηκε σε μία προσομοίωση θα συνεχιστεί στην επόμενη. Τα αποτελέσματα κάθε προσομοίωσης θα προέρχονται από στατιστικές της αυτής επανάληψης και μόνο.

4η Επιλογή: Initialize System (no), Initialize Statistics (no)
Ο χρόνος είναι προσθετικός. Αν η κάθε προσομοίωση διαρκεί 5000 λεπτά, η δεύτερη θα τελειώσει την χρονική στιγμή $TFIN_2$=10000, η τρίτη την στιγμή $TFIN_3$=15000 κ.λπ.. Κάθε λειτουργία που δεν ολοκληρώθηκε σε μία προσομοίωση θα συνεχιστεί στην επόμενη. Οι αναφορές θα είναι προσθετικές. Έτσι αν κάνουμε 10 προσομοιώσεις θα πάρουμε τα ίδια αποτελέσματα με αυτά που θα έδινε μία προσομοίωση που θα διαρκούσε όσο και οι δέκα μαζί.

Στην συνέχεια το μοντέλο ελέγχεται για σφάλματα με το εικονίδιο *Check* του πλαισίου εργαλείων *Run Interaction* και, αν δεν υπάρξει μήνυμα λάθους, προχωράμε στην εκτέλεση του προγράμματος (με *Run*).

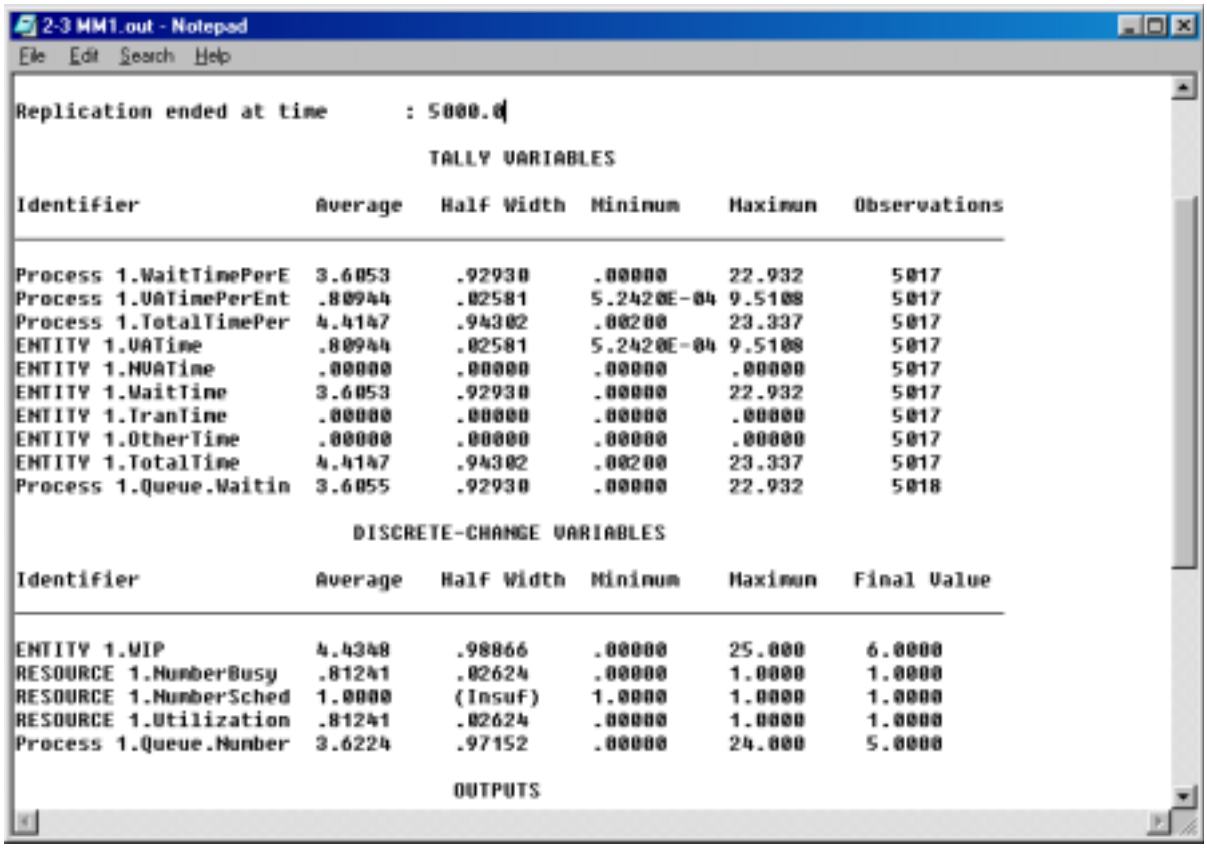
Ταχύτητα εκτέλεσης

Χρησιμοποιώντας τα πλήκτρα *Shift και* > *ή* < αυξάνουμε ή μειώνουμε αντίστοιχα την ταχύτητα εκτέλεσης του προγράμματος. Στην πραγματικότητα αυτό που αυξάνουμε είναι η απεικόνιση της κίνησης. Εξάλλου, με το κουμπί *Fast-Forward* εκτελούμε το πρόγραμμα με την μέγιστη ταχύτητα, αλλά χωρίς να φαίνεται η κίνηση των οντοτήτων (Animation Off).

[5] Η στατιστική ανεξαρτησία των πειραμάτων είναι αποτέλεσμα της χρήσης διαφορετικών τυχαίων αριθμών σε κάθε πείραμα. Οι τυχαίοι αριθμοί χρησιμοποιούνται για την περιγραφή των χρόνων αφίξεων και των διαρκειών εξυπηρετήσεων. Αυτό γίνεται αυτόματα από το πρόγραμμα χωρίς να απαιτείται ανάπτυξη κάποιου υποπρογράμματος από τον χρήστη.

2.2.2 Αποτελέσματα

Με το πέρας κάθε εκτέλεσης μπορούμε να δούμε τα αποτελέσματα της προσομοίωσης, τόσο αυτά που αυτομάτως υπολογίζονται, όσο και αυτά που έχει καθορίσει ο χρήστης (User Specified). Τα αποτελέσματα βρίσκονται κατά ομάδες στην περιοχή *Reports*, αλλά υπάρχει και η δυνατότητα προβολής τους μέσω του προγράμματος Notepad (βλ. Εικ.2.2.7). Για να επιλέξουμε τον δεύτερο τρόπο παρουσίασης των αποτελεσμάτων (Notepad) δίνουμε από ***Run/Setup/Reports*** στην περιοχή *Default Report* την επιλογή *SIMAN Summary Report(.out file).*



Εικόνα 2.2.7 M/M/1. Τα αποτελέσματα μιας εκτέλεσης από το πρόγραμμα Notepad

Στην αναφορά αποτελεσμάτων (Εικ.2.2.7) το πλήθος πελατών στην άπειρη ουρά *Process 1.Queue* αναφέρεται ως ***Process 1.Queue.NumberInQueue***. Αναμένεται στην ουρά να περιμένουν, κατά μέσο όρο, 3.62 πελάτες.

3. ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ARENA

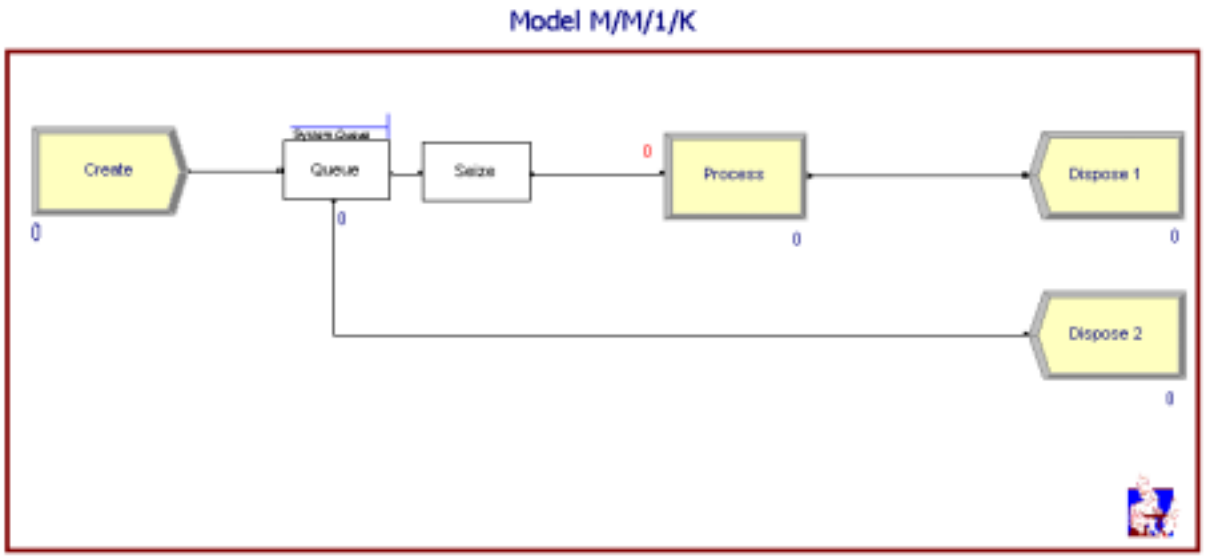
3.1 ΤΟ ΣΥΣΤΗΜΑ M/M/1/K

Σε αυτή την παράγραφο παρουσιάζεται ένας τρόπος μοντελοποίησης ενός συστήματος M/M/1/K, δηλαδή ενός συστήματος στο οποίο αφικνούνται πελάτες κατά Poisson, περιμένουν σε ουρά πεπερασμένης χωρητικότητας (σε αντιδιαστολή με το μοντέλο M/M/1 όπου η ουρά αναμονής ήταν άπειρη), εξυπηρετούνται κατά Poisson και αναχωρούν. Αν ένας πελάτης που φθάνει εύρει το σύστημα γεμάτο (δηλαδή K-1 πελάτες στην ουρά και έναν στην εξυπηρέτηση) τότε φεύγει.

3.1.1 Μοντελοποίηση

Για την ανάπτυξη ενός μοντέλου M/M/1/K χρησιμοποιούμε τα εξής modules:

- ένα Create για τις αφίξεις
- ένα Queue block για την ουρά πεπερασμένης χωρητικότητας
- ένα Seize block για την δέσμευση του εξυπηρετούντος
- ένα Process για την εξυπηρέτηση των πελατών και την αποδέσμευση του εξυπηρετούντος
- δύο Dispose modules. Ένα για την μοντελοποίηση της εξόδου των πελατών που εξυπηρετούνται και ένα για την έξοδο των πελατών που βρήκαν το σύστημα γεμάτο κατά την άφιξη τους και έφυγαν.



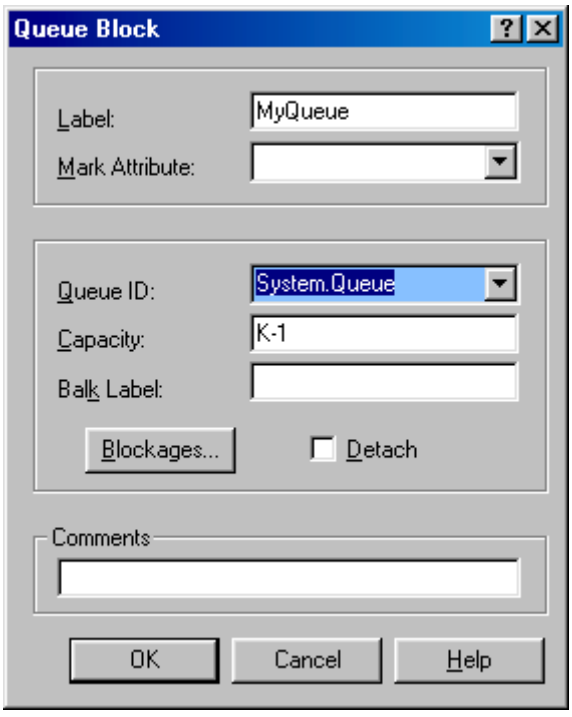
Εικόνα 3.1.1 Το μοντέλο M/M/1/K

Πελάτες φθάνουν στο σύστημα από το Create module και κατευθύνονται προς την πεπερασμένη ουρά. Αν η ουρά *System.Queue* (βλ. Εικ.3.1.2) είναι γεμάτη, οι πελάτες φεύγουν από το *Dispose 2*, διαφορετικά εισέρχονται σε αυτή περιμένοντας τη σειρά τους.

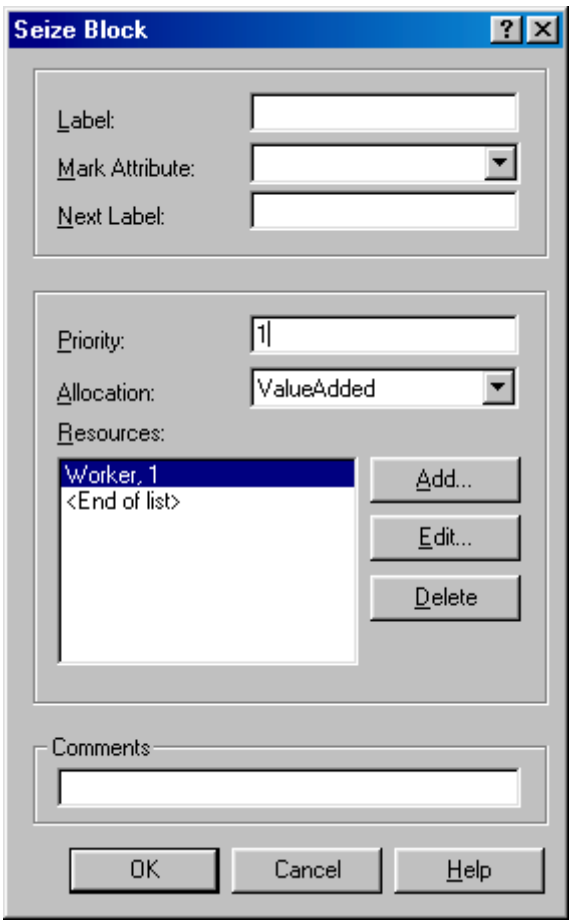
Όταν τελειώσει η εξυπηρέτηση ενός πελάτη δίνεται αυτόματα εντολή στο *Queue* block να επιτρέψει στον επόμενο πελάτη της ουράς να προχωρήσει. Ο τελευταίος περνά στο *Seize* block, όπου δεσμεύει τον εξυπηρετούντα. Κατόπιν, πάει στο σημείο εξυπηρέτησης *Process*, εξυπηρετείται, αποδεσμεύει τον εργαζόμενο και αναχωρεί από το *Dispose 1*.

Στην εφαρμογή που παρουσιάζεται

- οι αφίξεις είναι εκθετικά κατανεμημένες (με ρυθμό λ=1 πελάτη/λεπτό)
- η διάρκεια εξυπηρέτησης είναι εκθετικά κατανεμημένη (με ρυθμό μ=1.5 πελάτες/λεπτό)
- η ουρά αναμονής (System.Queue) έχει χωρητικότητα $K-1$ πελατών, ώστε το σύστημα να χωράει K πελάτες (εδώ K=10).
- οι μεταφορές στο σύστημα είναι κατά σύμβαση ακαριαίες, όπως και σε όλα τα μοντέλα της εργασίας
- η διάρκεια της προσομοίωσης (TSIM) είναι 5000 λεπτά, ενώ πραγματοποιείται μία προσομοίωση.



Εικόνα 3.1.2 Η πεπερασμένη ουρά στο μοντέλο M/M/1/K



Εικόνα 3.1.3 Δέσμευση πόρου εξυπηρέτησης στο μοντέλο M/M/1/K

3.1.2 Μέτρα απόδοσης

Με τα μέτρα απόδοσης έχουμε μία εικόνα για την καλή ή μη λειτουργία των συστημάτων. Ορισμένα βασικά μέτρα απόδοσης ενός αναμονητικού συστήματος είναι:

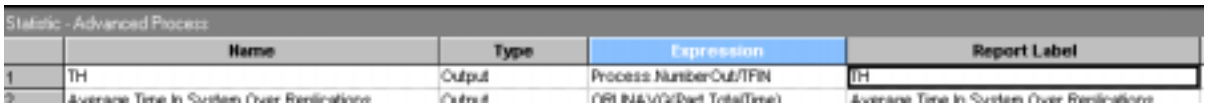
1. η παραγωγικότητα (TH, throughput). Δεν υπολογίζεται αυτόματα.
2. το μέσο πλήθος πελατών στην ουρά $(\overline{N_q})$. Υπολογίζεται αυτόματα
3. ο μέσος χρόνος αναμονής στην ουρά $(\overline{T_q})$. Υπολογίζεται αυτόματα.

Παραγωγικότητα

Για τον υπολογισμό της παραγωγικότητας δηλώνουμε στο Data module ***Statistic*** της ομάδας Advanced Process την εξίσωση:

$$TH = \frac{Process.NumberOut}{TFIN}$$, όπου

- *Process.NumberOut* είναι στην γλώσσα SIMAN ο αριθμός των πελατών που βγήκαν από το module *Process*, δηλαδή που εξυπηρετήθηκαν.
- TFIN είναι ο χρόνος περάτωσης της προσομοίωσης. Ο TFIN δείχνει και την διάρκεια μίας προσομοίωσης όταν αυτή ξεκινά την στιγμή t=0 (Initialize System Between Replications).



Εικόνα 3.1.4 Ορισμός της παραγωγικότητας (TH) ως μέτρου απόδοσης

Τα μέτρα απόδοσης 2 και 3 υπολογίζονται αυτόματα και βρίσκονται στην περιοχή *Reports*, στην κατηγορία *Queues*.

Αποτελέσματα - Σύγκριση με την θεωρία ουρών

Στον Πίνακα 3.1 παρουσιάζονται τα αποτελέσματα των μέτρων απόδοσης που προαναφέρθηκαν σε σχέση με τα αναμενόμενα από την θεωρία ουρών αναμονής (βλ. Παράρτ.3). Παρατηρούμε μία καλή προσέγγιση των αποτελεσμάτων της θεωρίας από αυτά της προσομοίωσης.

Μέτρο Απόδοσης	**Arena**	**Θεωρία**
$\overline{N_q}$	1,1245	1,2086
$\overline{T_q}$	1,1374	1,2157
TH= $\overline{N_q}/\overline{T_q}$	0,9860	0,9941

Πίνακας 3.1 Μέτρα απόδοσης υπολογισμένα από την Arena και σύγκριση τους με τα θεωρητικά αποτελέσματα

3.2 ΤΟ ΣΥΣΤΗΜΑ M/M/m/K

Στην ενότητα αυτή περιγράφεται ένας τρόπος μοντελοποίησης συστημάτων M/M/m/K, δηλαδή συστημάτων με m εξυπηρετούντες και χώρο αναμονής χωρητικότητας K-m πελατών.

Ήδη, στην παράγραφο 3.1, παρουσιάστηκε το μοντέλο M/M/1/K, το οποίο είναι σε αρκετά σημεία ίδιο με το μοντέλο που αναπτύσσουμε εδώ. Οι αφίξεις, η αναμονή και οι αναχωρήσεις πελατών μοντελοποιούνται με τον ίδιο τρόπο. Έτσι, ο αναγνώστης παραπέμπεται στην παράγραφο 3.1 για την μοντελοποίηση τους. Η κύρια διαφορά μεταξύ M/M/1/K και M/M/m/K (με m>1) είναι στη μοντελοποίηση του χώρου εξυπηρέτησης.

Στο μοντέλο M/M/1/K η ύπαρξη ενός εξυπηρετούντος ήταν εύκολο να αναπαρασταθεί, με χρήση ενός Seize block και ενός Process module. Αντίθετα, σε ένα σύστημα με περισσότερους πόρους εξυπηρέτησης[6], απαιτείται ένας συνδυασμός υπομονάδων σχεδίασης

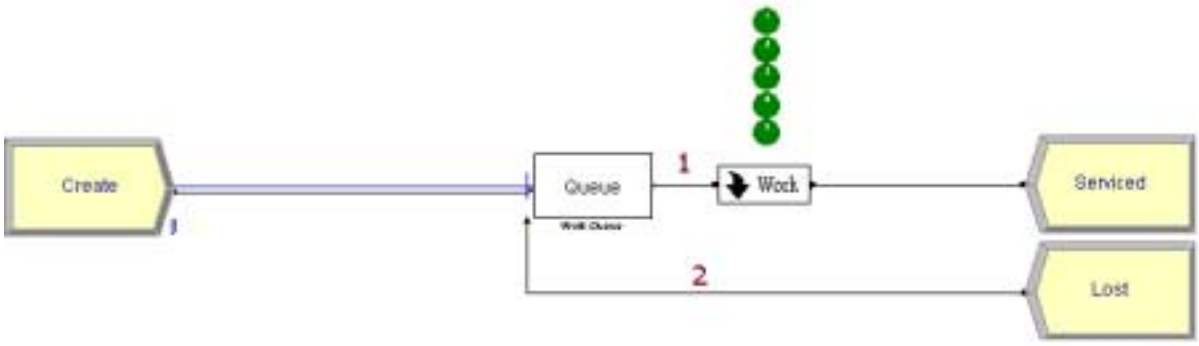
[6] Ο όρος *πόρος* χρησιμοποιείται για να δηλώσει έναν εξυπηρετούντα που ανήκει σε ένα σύνολο εξυπηρετούντων, οι οποίοι λειτουργούν εν παραλλήλω.

(modules) που να αναπαριστά τόσο το πλήθος των πόρων, όσο και τον τρόπο με τον οποίο επιλέγεται ποιος από αυτούς θα εξυπηρετήσει τον επόμενο πελάτη. Το μοντέλο που αναπτύχθηκε χρησιμοποιεί **πρωτόκολλο εξυπηρέτησης POR** (*Preferred Order Rule*), σύμφωνα με το οποίο επιλέγεται ο πρώτος διαθέσιμος πόρος.

3.2.1 Εφαρμογή για M/M/5/10

Αφίξεις

Για τις αφίξεις χρησιμοποιούμε ένα Create module, όπως στο μοντέλο M/M/1/K. Δηλώνουμε την κατανομή του χρόνου ανάμεσα στις αφίξεις, την μονάδα χρόνου (sec, min, hours) και, προαιρετικά, ονομάζουμε το module με όνομα της επιλογής μας.



Εικόνα 3.2.1 Το μοντέλο M/M/5/10

Ουρά

Για την ουρά πεπερασμένης χωρητικότητας χρησιμοποιούμε ένα Queue block. Η χωρητικότητα της ουράς αναμονής (K-5) δηλώνεται στην περιοχή *Capacity* του αντίστοιχου Queue block. Το K ορίζεται στη συνέχεια της μοντελοποίησης.

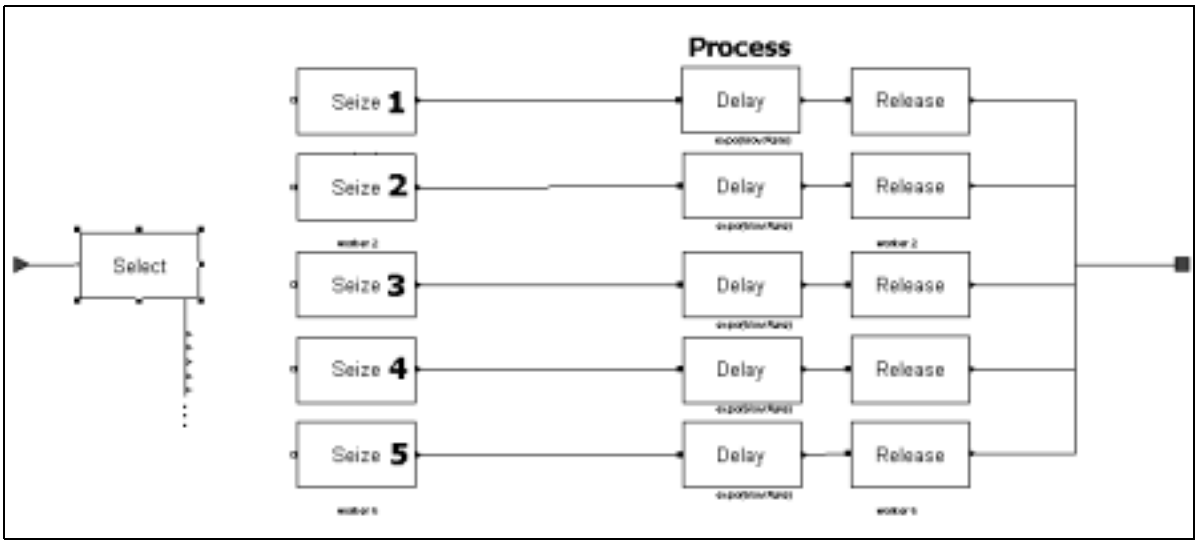
Έστω ότι φθάνει ένας πελάτης στο σύστημα. Εάν βρει την ουρά σε ενδιάμεση κατάσταση ακολουθεί το μονοπάτι 1 της εικόνας 3.2.1. Εάν, όμως, η ουρά είναι γεμάτη τότε θα απορριφθεί από το σύστημα ακολουθώντας το μονοπάτι 2.

Εξυπηρέτηση

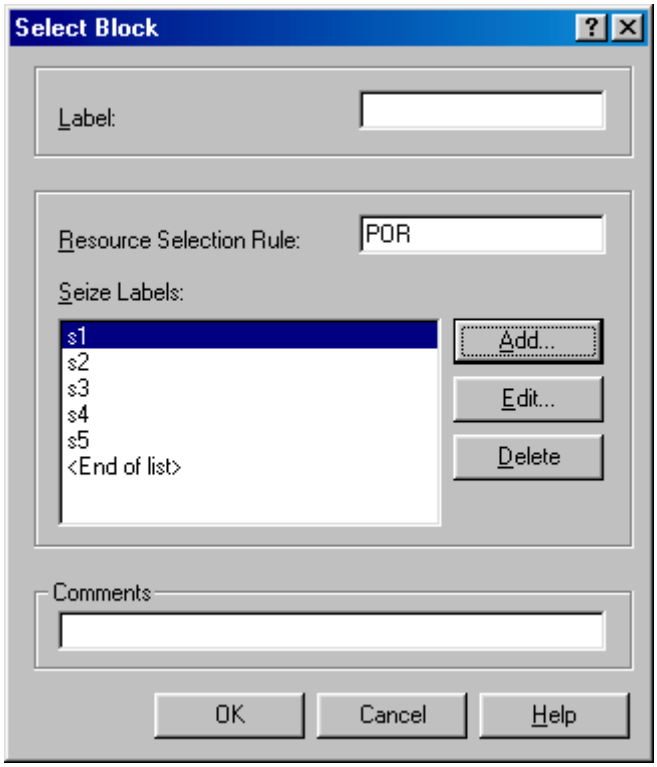
Η εξυπηρέτηση έπεται της αναμονής. Τοποθετούμε δεξιά του Queue block ένα **υπομοντέλο** (sub model) από *Object/Sub model/Add_Sub model*. Συνδέουμε το Queue block με το Sub model (*Work*) και το τελευταίο με ένα Dispose module (*Serviced).*

Ένα υπομοντέλο μας δίνει την δυνατότητα να ομαδοποιούμε ένα σύνολο από modules και να χωρίζουμε το σύστημα σε περισσότερα του ενός επίπεδα. Αυτό είναι χρήσιμο σε περιπτώσεις που ένα τμήμα του αναπτυσσόμενου μοντέλου απαιτεί μεγάλο αριθμό modules για να μοντελοποιηθεί. Για την διαδικασία της εξυπηρέτησης χρησιμοποιούμε τα δεκαέξι (16) modules της Εικόνας 3.2.2. Ανοίγουμε ένα sub model όπως και ένα module, δηλαδή με διπλό κλικ. Στο **Select** block επιλέγεται πού θα σταλεί κάθε οντότητα που περνά μέσα από αυτό, δηλαδή επιλέγεται ποιος πόρος θα χρησιμοποιηθεί στην επόμενη εξυπηρέτηση. Δηλώνουμε το πρωτόκολλο εξυπηρέτησης, που όπως προαναφέρθηκε είναι POR. Έπεται το τρίπτυχο *Seize-Delay-Release* πέντε (5) φορές, μία για κάθε εξυπηρετούντα. Σε κάθε ένα **Seize** block δηλώνεται ένας από τους πέντε εξυπηρετούντες. Έτσι, στο Seize *s1* δηλώνουμε

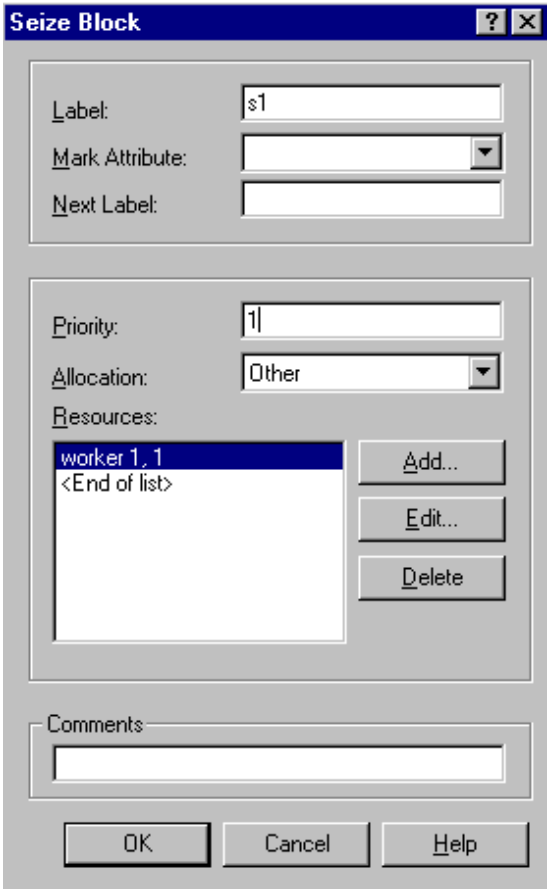
τον πρώτο εξυπηρετούντα (*worker1*), στο s2 δηλώνουμε τον δεύτερο (worker2) κ.ο.κ.. Κάθε **Delay** block μοντελοποιεί μια απλή καθυστέρηση. Χρειάζεται να οριστεί μόνο η διάρκεια της. Ανάλογα ενεργούμε και στα **Release** blocks, όπου δηλώνεται ποιος πόρος αποδεσμεύεται. Κλείνουμε το υπομοντέλο με δεξί κλικ και *Close Sub model*.



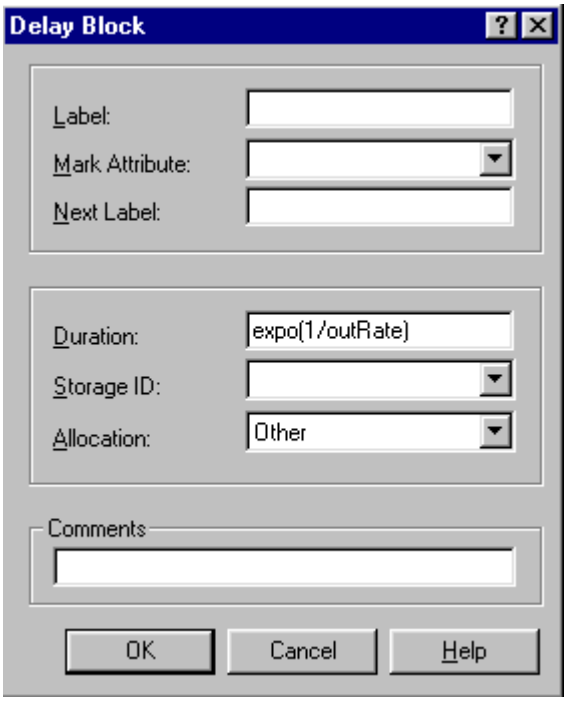
Εικόνα 3.2.2 Το εσωτερικό του sub model *Work*



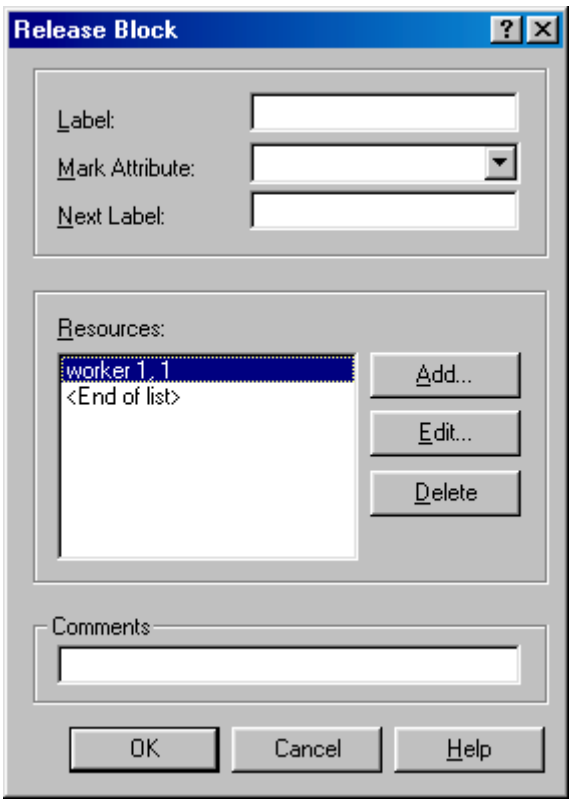
Εικόνα 3.2.3 Επιλογή του πόρου που θα δεσμευθεί για την επόμενη εξυπηρέτηση.
Επιλέγεται ο πρώτος διαθέσιμος πόρος (FOR)



Εικόνα 3.2.4 Το Seize block s1 για την δέσμευση του πρώτου εξυπηρετούντος (Worker 1)



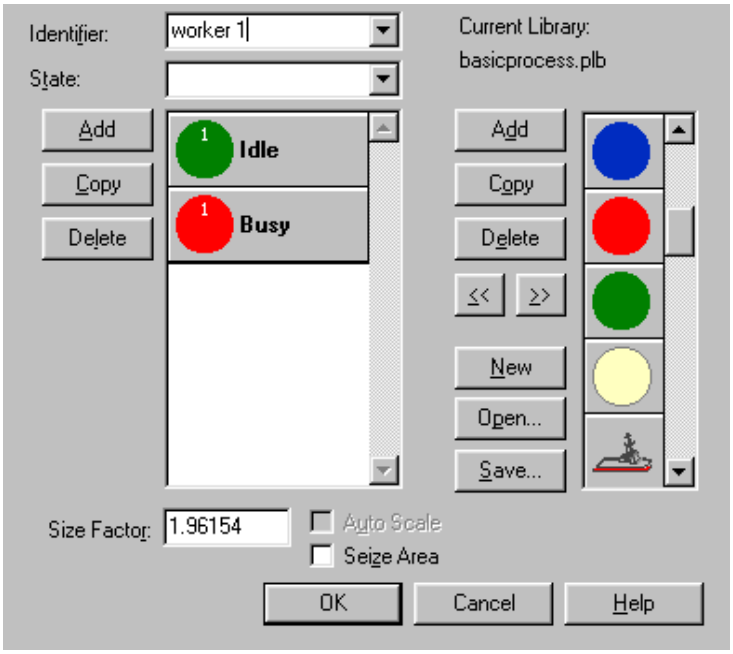
Εικόνα 3.2.5 Το Delay block για την μοντελοποίηση του χρόνου εξυπηρέτησης



Εικόνα 3.2.6 Το Release block για την αποδέσμευση των πόρων εξυπηρέτησης

Άλλες Ρυθμίσεις

Κατά την διάρκεια της εκτέλεσης μπορούμε να παρατηρούμε την εξέλιξη της κατάστασης κάθε εξυπηρετούντος, δηλαδή πότε είναι απασχολημένος και πότε όχι.



Εικόνα 3.2.7 Προσθήκη απεικόνισης του εξυπηρετούντος worker1

Υποθέτουμε ότι οι εξυπηρετούντες δεν σταματούν να είναι διαθέσιμοι για εργασία. Είτε θα εργάζονται, είτε θα περιμένουν, όντας σε ετοιμότητα. Συνεπώς, οι δύο πιθανές καταστάσεις τους είναι· **Busy** (απασχολημένος) και **Idle** (διαθέσιμος). Όπως φαίνεται και στην Εικόνα 3.2.1, πάνω από το υπομοντέλο υπάρχουν πέντε πράσινοι κύκλοι. Πρόκειται για απεικονίσεις των πέντε πόρων εξυπηρέτησης. Για να εισάγουμε κάθε μία από αυτές χρησιμοποιούμε το εικονίδιο Resource της γραμμής εργαλείων *Animate*.

Κάθε εικόνα (κύκλος) θα αλλάζει χρώμα ανάλογα με την κατάσταση του πόρου στον οποίο αντιστοιχεί. Στο δεξί μέρος της Εικόνας 3.2.7 βρίσκεται ένα σύνολο υποψήφιων απεικονίσεων από τις οποίες διαλέγουμε ποια θα εμφανίζεται στην κατάσταση Busy και ποια στην κατάσταση Idle (αριστερά της εικόνας είναι οι απεικονίσεις που επιλέξαμε). Με *OK* επιστρέφουμε στην περιοχή σχεδίασης και εισάγουμε την απεικόνιση.

`Identifier`	Όνομα πόρου που απεικονίζεται
`Seize`	Κατάσταση πόρου
`Size Factor`	Μέγεθος σχήματος (με δοκιμή και σφάλμα)

Πίνακας 3.2 Απεικόνιση των πόρων του συστήματος και της εξέλιξης της κατάστασης τους στο χρόνο

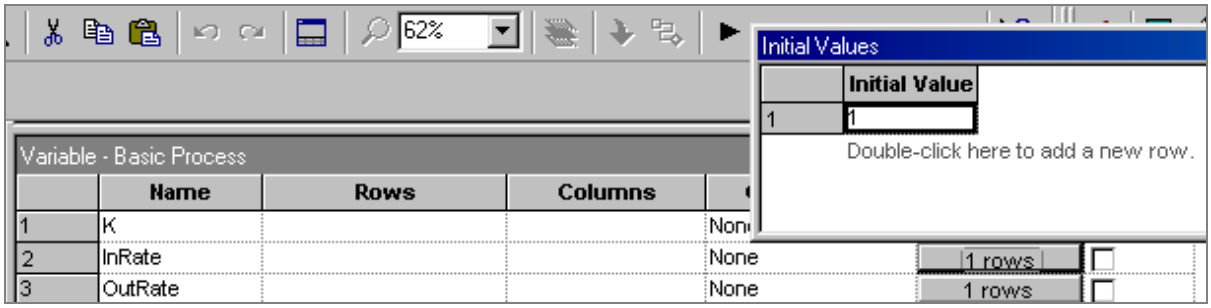
Τελικές Ρυθμίσεις

Ολοκληρώνουμε την μοντελοποίηση ορίζοντας τις παραμέτρους του συστήματος. Από *Basic Process/Variable* ορίζουμε τιμές για τα εξής:

- τον ρυθμό αφίξεων λ (εδώ inRate), τον οποίο χρησιμοποιήσαμε στο Create module για να ορίσουμε την στατιστική τους.
- τον ρυθμό εξυπηρέτησης μ (εδώ outRate) του ενός εξυπηρετούντος.
- την χωρητικότητα K του συστήματος (ουρά+εξυπηρέτηση=10)

Και οι τρεις αυτές παράμετροι μένουν σταθερές κατά την διάρκεια της εκτέλεσης, συνεπώς δεν χρειάζεται να αρχικοποιούνται (initialize) μεταξύ των προσομοιώσεων. Έτσι, και στις τρεις επιλέγουμε:

Clear Option: **None**



Εικόνα 3.2.8 Ορισμός των παραμέτρων και των τιμών τους

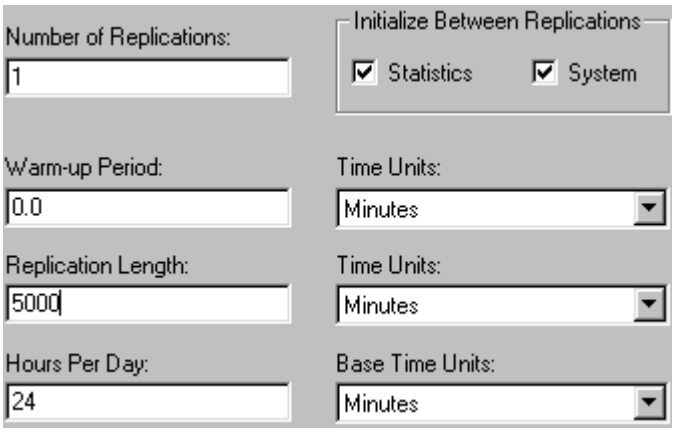
Ορίζουμε δύο επιπλέον στατιστικές (μέτρα απόδοσης)· την παραγωγικότητα (*TH*) και τον αριθμό των πελατών που βρήκαν το σύστημα γεμάτο και απορρίφθηκαν (*Lost.NumberOut*).

Statistic - Advanced Process

	Name	Type	Expression	Report Label	Output File
1	TH	Output	Serviced.NumberOut/TFIN	TH	
2	Lost Customers	Output	Lost.NumberOut	Lost Customers	

Εικόνα 3.2.9 Data module Statistic. Ορισμός επιπλέον στατιστικών

Τέλος, από *Advanced Process/Statistic* επιλέγουμε τα εξής:



Εικόνα 3.2.10 Τελικές ρυθμίσεις πριν την εκτέλεση του προγράμματος

<u>Εκτέλεση</u>
Εκτελέσαμε το πρόγραμμα για μία (1) προσομοίωση των 5000 λεπτών με:

- $\lambda = inRate = 1$
- $\mu = outRate = \frac{1,1}{5} = 0.22$ για κάθε εξυπηρετούντα

<u>Σύγκριση αποτελεσμάτων με την θεωρία ουρών</u>
Στον πίνακα που ακολουθεί παρουσιάζονται τα αποτελέσματα της προσομοίωσης για τρία βασικά μέτρα απόδοσης, σε αντιπαραβολή με τα αναμενόμενα αποτελέσματα από την θεωρία ουρών αναμονής (βλ. Παράρτ.3). Τα αποτελέσματα μπορούν να θεωρηθούν καλά, ειδικά αν ληφθεί υπόψη ότι προέκυψαν από μία προσομοίωση των 5000 λεπτών.

Μέτρο Απόδοσης	**Arena**	**Θεωρία**
$\overline{N_q}$	1,193	1,359
$\overline{T_q}$	1,308	1,475
TH= $\overline{N_q}/\overline{T_q}$	0,912	0,921

3.3 ΣΤΑΤΙΣΤΙΚΟΣ ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΤΟΥ ΠΛΗΘΟΥΣ ΠΡΟΣΟΜΟΙΩΣΕΩΝ

3.3.1 Εισαγωγή

Ένα σημαντικό θέμα στην Προσομοίωση Συστημάτων είναι το πλήθος προσομοιώσεων που απαιτούνται ώστε τα αποτελέσματα μιας εφαρμογής να μην έχουν μεγάλο σφάλμα. Το πρόβλημα αυτό μπορεί να διατυπωθεί ως εξής: *Να εκτιμηθεί το πλήθος προσομοιώσεων που απαιτούνται ώστε, με πιθανότητα 1-α %, το σχετικό σφάλμα της εκτίμησης ενός μέτρου απόδοσης ενός συστήματος να μην είναι μεγαλύτερο από ένα ποσοστό* ε_{max}. Για την επίλυση του προβλήματος αυτού χρησιμοποιείται ο αλγόριθμος της επόμενης παραγράφου [2], τον οποίο θα παρουσιάσουμε για την περίπτωση ενός συστήματος αναμονής M/M/m/K.

3.3.2 Επίλυση σε M/M/m/K για την μέση παραγωγικότητα

Χρησιμοποιούμε το μοντέλο M/M/5/10 που αναπτύχθηκε στην παράγραφο 3.2 και επιλύουμε το πρόβλημα του πλήθους προσομοιώσεων (n) για το μέτρο της παραγωγικότητας ή μέσου ρυθμού παραγωγής (TH). Έχοντας έτοιμο το μοντέλο M/M/5/10, στόχος είναι να κατασκευαστεί μία επιπλέον δομή που να ελέγχει την συνθήκη τερματισμού (Εξ.(1)).

$$\text{n=n}^{*}\text{, όταν} \qquad \sigma = t_{a/2,n-1}\frac{S_{TH}(n)}{\sqrt{n}} \le \frac{\varepsilon_{\max}\left|\overline{TH}(n)\right|}{1+\varepsilon_{\max}}\text{, όπου} \tag{1}$$

- $t_{\alpha/2,n-1}$ η τιμή της κατανομής t με n-1 βαθμούς ελευθερίας που έχει προς τα δεξιά της μάζα πιθανότητας α/2,
- $S_{TH}(n)$ η δειγματική τυπική απόκλιση που προκύπτει από n προσομοιώσεις.

Αλγόριθμος επίλυσης

Ο αλγόριθμος που επιλύει το πρόβλημα του πλήθους προσομοιώσεων ως προς το TH είναι:

A. Νέα προσομοίωση $(n = n + 1)$

B. Τέλος προσομοίωσης

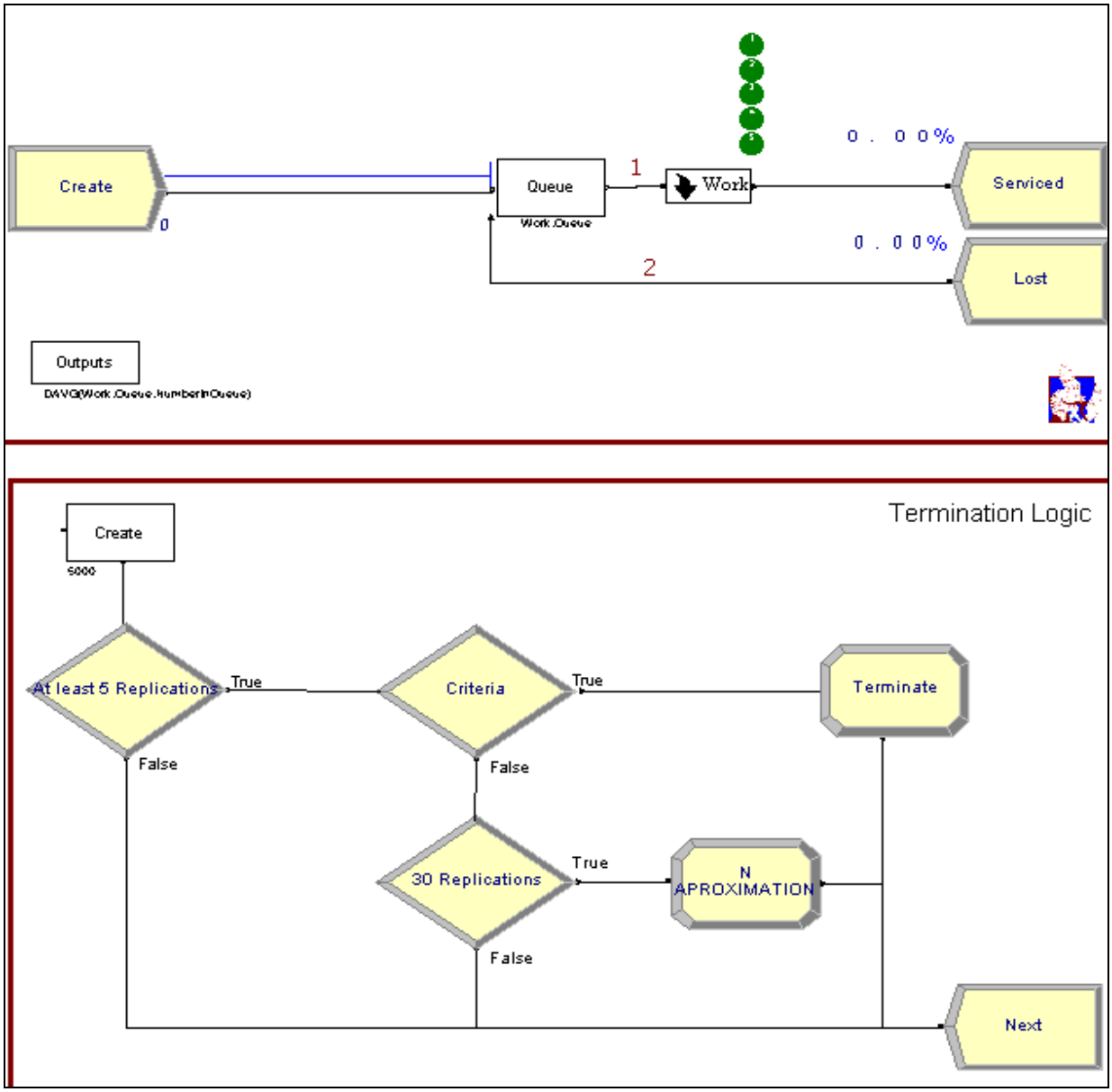
1. υπολόγισε την παραγωγικότητα του συστήματος από όλες τις έως τώρα προσομοιώσεις
2. εάν n>4 πήγαινε στο βήμα 3 διαφορετικά πήγαινε στο Α
3. εάν ισχύει η Εξ.(1) περάτωσε την εκτέλεση ($n=n^{*}$) διαφορετικά πήγαινε στο 4
4. εάν n=30 περάτωσε την εκτέλεση και υπολόγισε προσεγγιστικά τον απαιτούμενο αριθμό προσομοιώσεων (n^{*}) με την Εξ.(2) διαφορετικά πήγαινε στο Α.

$$n^{*} \cong n\frac{\varepsilon^{2}}{\left(\varepsilon_{\max}\cdot\overline{TH}\right)^{2}} \tag{2}$$

Από *Run/Setup/ Replication Parameters/ Number of Replications*, τίθεται ως (μέγιστος) αριθμός προσομοιώσεων (MREP) οι τριάντα (30). Μετά το πέρας κάθε μίας θα εκτελείται το μέρος Β του παραπάνω αλγορίθμου.

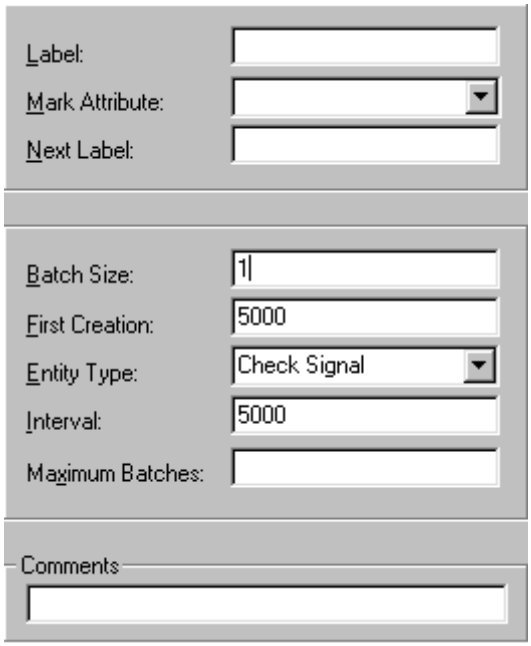
3.3.3 Μοντελοποίηση του αλγορίθμου

Ανοίγουμε το έτοιμο μοντέλο M/M/5/10 που παρουσιάστηκε στην Παράγραφο 3.2.1 και προσθέτουμε την δομή της Εικόνας 3.3.1, η οποία θα υλοποιήσει το μέρος Β του παραπάνω αλγορίθμου. Τόσο το κυρίως μοντέλο, όσο και η νέα δομή εκτελούνται ταυτόχρονα, παρ' όλο που στη πράξη η δεύτερη θα ενεργοποιείται μόνο κάθε 5000 λεπτά και θα εκτελείται ακαριαία.



Εικόνα 3.3.1 Το μοντέλο M/M/5/K με μηχανισμό υπολογισμού του απαιτούμενου πλήθους προσομοιώσεων

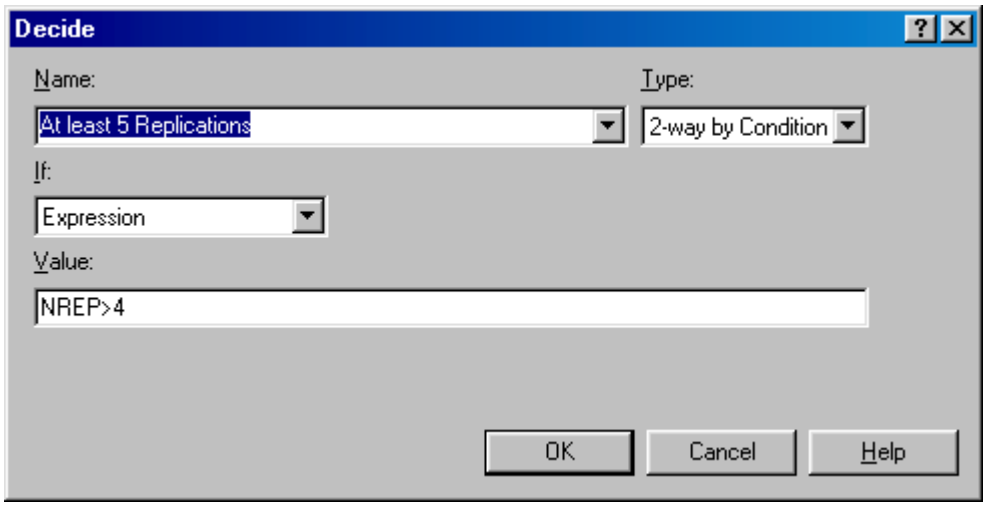
Αφετηρία του μηχανισμού ελέγχου είναι ένα **Create block**. Ζητάμε το λεπτό 5000 (διάρκεια μιας προσομοίωσης) να «γεννηθεί» η πρώτη οντότητα ελέγχου (Check Signal). Έως την στιγμή αυτή έχει λειτουργήσει μόνο το επάνω μέρος του μοντέλου. Επόμενες οντότητες ελέγχου δημιουργούνται μία κάθε 5000 λεπτά.



Εικόνα 3.3.2 Περιοδική δημιουργία οντοτήτων ελέγχου από το Create block

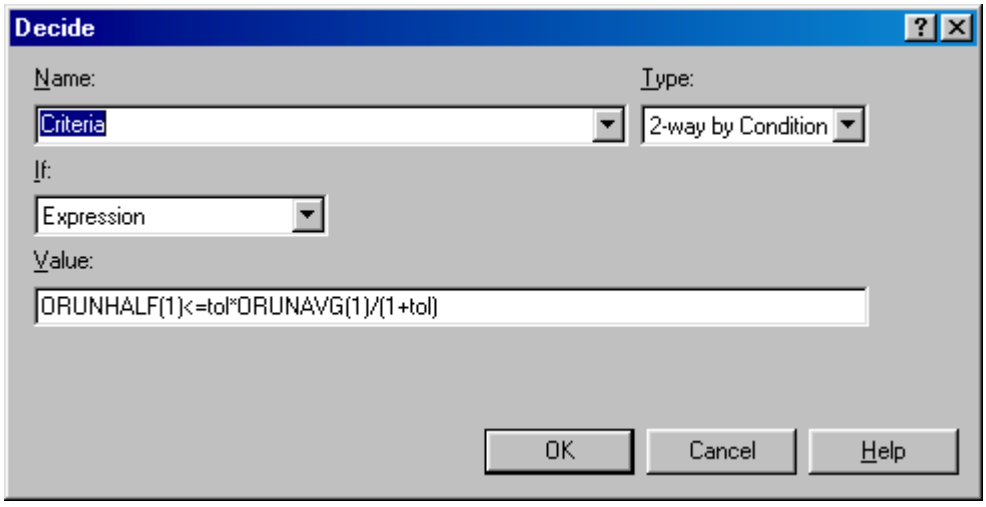
Τα βήματα του Β μέρους

1. Η παραγωγικότητα ορίζεται στην περιοχή *Advanced Process/Statistic*
2. Εξετάζεται εάν έως τώρα έχουν γίνει τουλάχιστον πέντε προσομοιώσεις (Εικ.3.3.3)



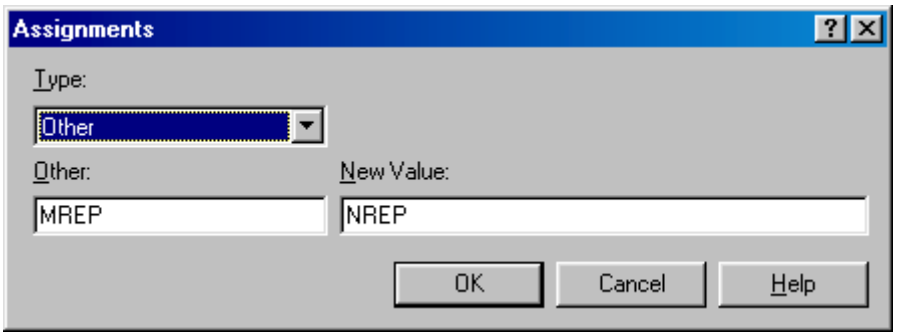
Εικόνα 3.3.3 Το βήμα 2, ώστε να εκτελεστούν τουλάχιστον πέντε (5) προσομοιώσεις

3. Το Decide module *Criteria* ελέγχει την ισχύ της Εξ.(1) (Εικ.3.3.4).



Εικόνα 3.3.4 Το βήμα 3 για τον έλεγχο της Εξ.(1)

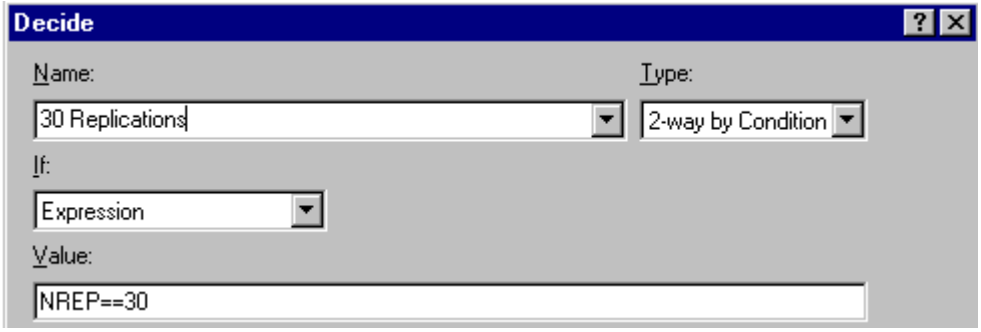
Ακολουθεί ένα **Assign module** (*Terminate*) που ολοκληρώνει το βήμα 3. Εκεί τίθεται ως τελευταία προσομοίωση (MREP) η τρέχουσα (NREP), οπότε τερματίζεται έμμεσα η εκτέλεση του προγράμματος (Εικ.3.3.5).



Εικόνα 3.3.5 Τερματισμός των προσομοιώσεων από ένα Assign module

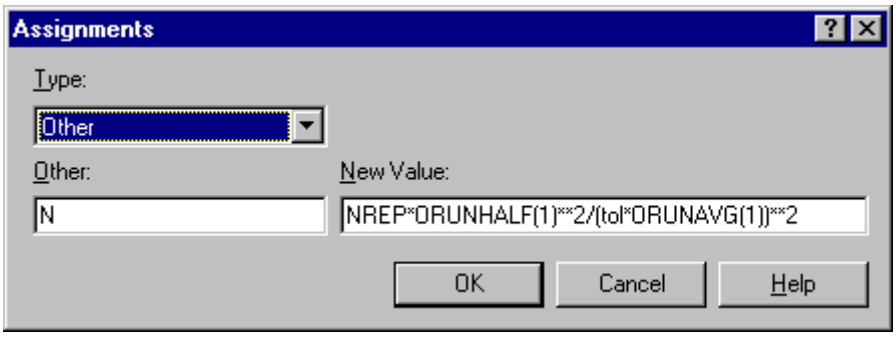
Τέλος, προστίθεται στη δομή το Dispose module *Next* που αποτελεί την έξοδο των οντοτήτων ελέγχου από το σύστημα.

4. Αν εκτελεσθούν και οι τριάντα προσομοιώσεις χωρίς να ικανοποιηθεί η Εξ.(1), η μεταβλητή N δίνει μια εκτίμηση της λύσης από την Εξ.(2).



Εικόνα 3.3.6 Ο έλεγχος του βήματος 4

Η ενημέρωση της μεταβλητής N γίνεται από το Assign module *N APROXIMATION* (βλ. Εικ. 3.3.7).



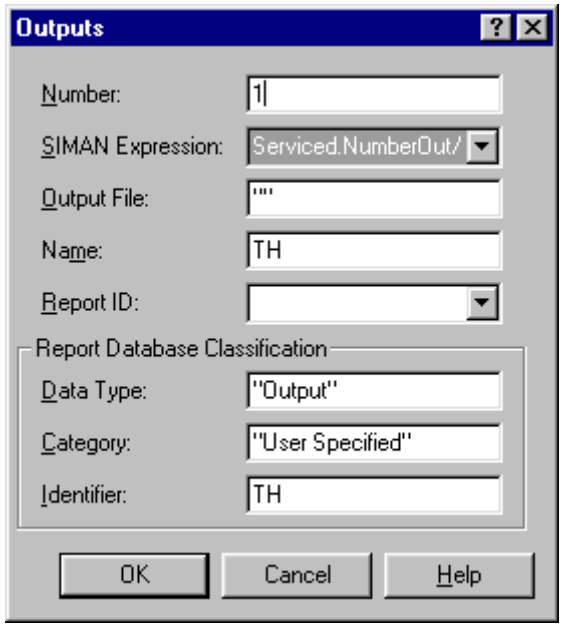
Εικόνα 3.3.7 Εφαρμογή της Εξ.(2)

3.3.4 Οι εντολές ORUNHALF και ORUNAVG

Η εντολή ORUNHALF(x) επιστρέφει το αναμενόμενο σφάλμα της εκτίμησης του μέτρου απόδοσης x, σαν αποτέλεσμα όλων των έως τώρα προσομοιώσεων με εγγύηση 95%. Ομοίως, η εντολή ORUNAVG(x) επιστρέφει τη μέση τιμή του μέτρου απόδοσης x μεταξύ όλων των έως τώρα προσομοιώσεων

ORUNAVG (TH)	Μέση τιμή του TH μετά από n προσομοιώσεις
ORUNHALF (TH)	Αναμενόμενο απόλυτο σφάλμα μετά από n προσομοιώσεις. Το πρώτο μέλος της Εξ.(1)

Στην Εικόνα 3.3.7 ο αριθμός που εμφανίζεται στις παρενθέσεις των εντολών ORUNHALF και ORUNAVG παραπέμπει την Arena σε εκείνο το μέτρο απόδοσης που φέρει τον αριθμό 1 στο Outputs element (Εικ.3.3.8).



Εικόνα 3.3.8 Ορισμός του TH στο Outputs element

Τελικές Ρυθμίσεις

Το μοντέλο ολοκληρώνεται ορίζοντας από *Project Bar/Basic Process/ Variable* τις μεταβλητές K, InRate(=λ), OutRate(=μ), tol, N(=n^*).

Variable - Basic Process

	Name	Rows	Columns	Clear Option	Initial Values	Statistics
1	K			None	1 rows	☐
2	InRate			None	1 rows	☐
3	OutRate			None	1 rows	☐
4	Tol			None	1 rows	☐
5	N			System	0 rows	☑

Εικόνα 3.3.9 Ορισμός του TH στο Outputs element

3.3.5 Εκτέλεση και Αποτελέσματα

Η Arena έχει αυτόματα δημιουργήσει κώδικα σε γλώσσα SIMAN, τον οποίο μπορούμε να δούμε από *Run/SIMAN/View*. Αν όλα είναι σωστά, πατώντας το κουμπί **Run** ξεκινά η εκτέλεση.

Εφαρμογή

Στο σενάριο που εξετάστηκε επιλέχθηκαν τα εξής:

- tol=1% μέγιστο επιτρεπτό σχετικό σφάλμα εκτίμησης
- Replication Length=5000 min. η διάρκεια της μίας προσομοίωσης

Η συνθήκη τερματισμού δεν ίσχυσε κατά την διάρκεια των τριάντα προσομοιώσεων, οπότε εφαρμόστηκε το πέμπτο βήμα του αλγορίθμου. Ακολουθεί μέρος των αποτελεσμάτων. Κατά προσέγγιση απαιτούνται περίπου 233 προσομοιώσεις για να ικανοποιηθεί η Εξ.(1).

Replication 30 Start Time: 0,00 Stop Time: 5.000,00 Time Units: Minutes

Time Persistent

Variable	Average	Half Width	Minimum	Maximum
N	0	(Insufficient)	0	232.75

Other

None	Value
Queue Length	1.4178

Output	Value
Lost Customers	466.00
TH	0.9328

Εικόνα 3.3.10 Αποτελέσματα για τις μεταβλητές N, TH

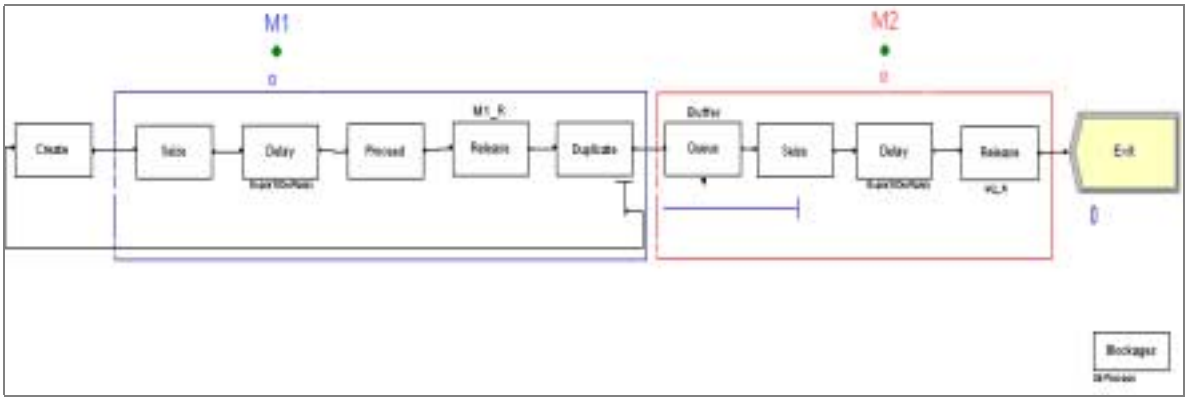
3.4 ΓΡΑΜΜΕΣ ΠΑΡΑΓΩΓΗΣ

Έστω μία γραμμή παραγωγής Ν φάσεων, στην οποία εισέρχονται ακατέργαστα κομμάτια προς παραγωγή τελικού προϊόντος. Πριν από κάθε μηχανή υπάρχει μία ενδιάμεση αποθήκη πεπερασμένης χωρητικότητας. Η πρώτη μηχανή θεωρείται ως διαδικασία αφίξεων των ακατέργαστων κομματιών από μία άπειρη αποθήκη. Η διάρκεια της επεξεργασίας σε κάθε φάση ακολουθεί συγκεκριμένη στατιστική (εδώ την εκθετική κατανομή).

Αρχικά, μελετάται η μεταβολή της παραγωγικότητας του συστήματος συναρτήσει του μεγέθους της γραμμής, ενώ στη συνέχεια εξετάζεται η μεταβολή της παραγωγικότητας μιας γραμμής δύο φάσεων (δύο μηχανών) σε σχέση με τη χωρητικότητα των ενδιάμεσων αποθηκών.

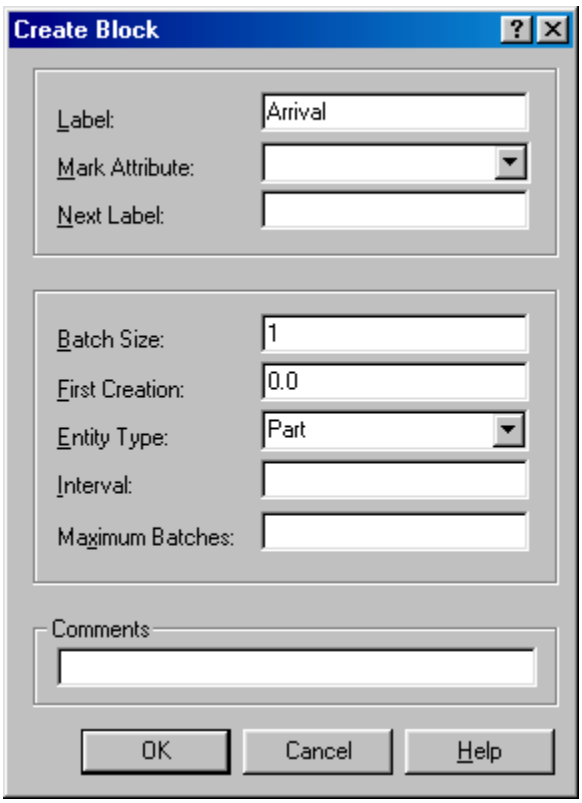
3.4.1 Μοντελοποίηση

Στην Εικόνα 3.4.1 παρουσιάζεται το μοντέλο μιας γραμμής παραγωγής δύο φάσεων.

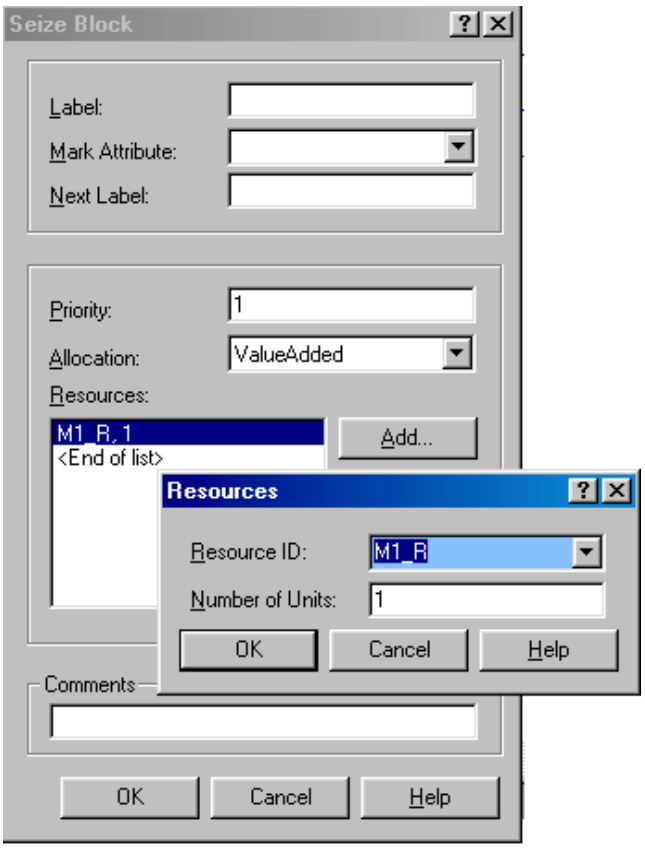


Εικόνα 3.4.1 Γραμμή παραγωγής δύο φάσεων (μηχανών)

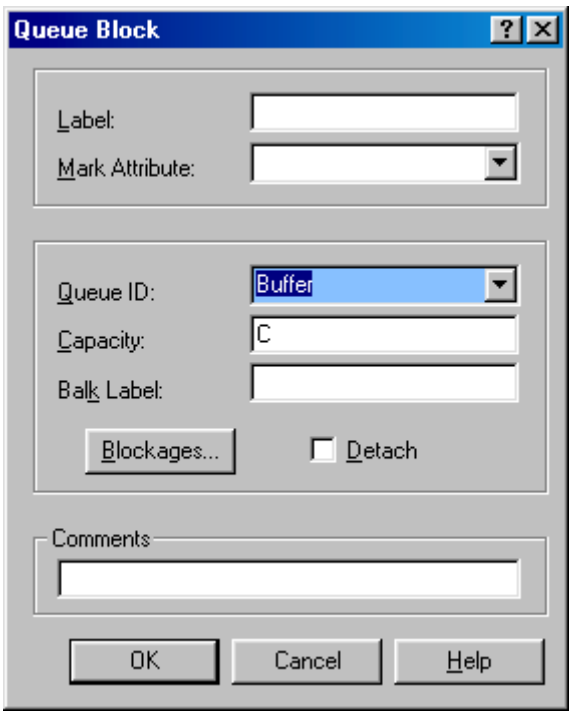
Κάθε κομμάτι αφικνείται στο σύστημα από ένα **Create block**. Ακολουθεί το **Seize block**, όπου δεσμεύεται η πρώτη μηχανή (M1). Ακολουθεί η επεξεργασία από την M1, η οποία επεξεργασία μοντελοποιείται από ένα ***Delay* block**. Κατόπιν, το **Proceed block** επιτρέπει στο κομμάτι να φύγει από την μηχανή M1 μόνο και τότε μόνο, όταν η αποθήκη (**Buffer.Queue**) μπορεί να το δεχτεί, δηλαδή όταν δεν είναι γεμάτη. Σε μια τέτοια περίπτωση, το κομμάτι προχωρά στο **Duplicate block** που «παράγει» μία δεύτερη οντότητα, η οποία θα αποτελέσει την επόμενη άφιξη, αφού αποδεσμεύσει την M1 περνώντας από το **Release block**.



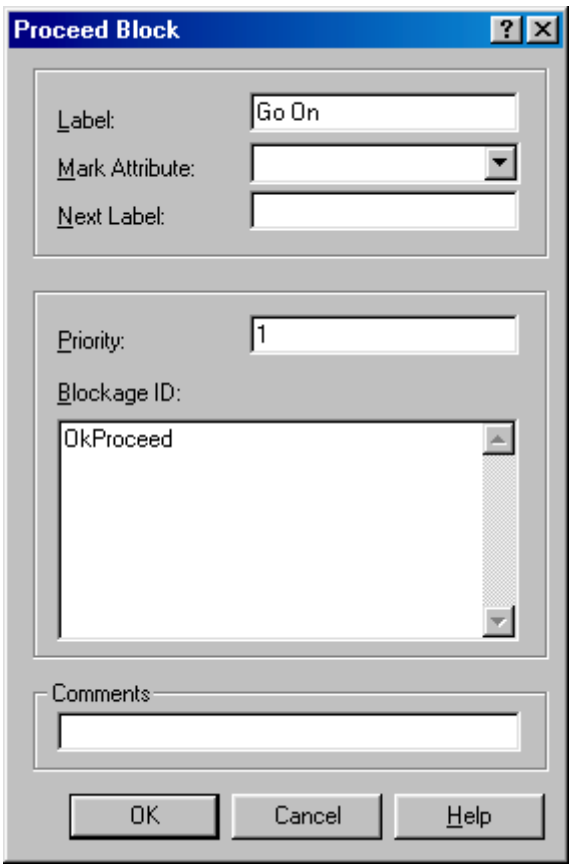
Εικόνα 3.4.2 Το Create block για τις αφίξεις



Εικόνα 3.4.3 Η δέσμευση του πόρου της M1 (M1_ R) από το Seize block



Εικόνα 3.4.4 Η ενδιάμεση αποθήκη



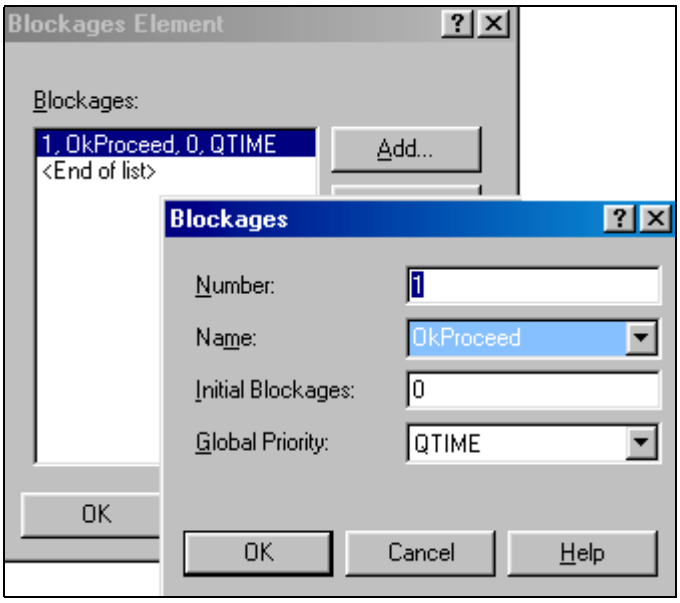
Εικόνα 3.4.5 Σημείο διακοπής της κίνησης των κομματιών σε περίπτωση που η ενδιάμεση αποθήκη γεμίσει

Στο δεύτερο τμήμα της γραμμής (δεύτερο πλαίσιο) συναντάται ένα Queue block που αναπαριστά την αποθήκη πεπερασμένης χωρητικότητας. Ακολουθεί το τρίπτυχο Seize-Delay-Release για την μοντελοποίηση ενός σημείου επεξεργασίας (ή εξυπηρέτησης). Στην προκειμένη περίπτωση μοντελοποιείται η δεύτερη μηχανή της γραμμής (M2). Το μοντέλο ολοκληρώνεται με την αποδέσμευση της μηχανής M2 (Release block) και ένα Dispose module που αναπαριστά την έξοδο του συστήματος.

Προσθέσαμε στο μοντέλο μία απεικόνιση της αποθήκης () με το κουμπί Queue  της γραμμής εργαλείων *Animate*. Έτσι, κατά τη διάρκεια της εκτέλεσης, μπορούμε να έχουμε εικόνα της εξέλιξης της στάθμης της αποθήκης στον χρόνο.

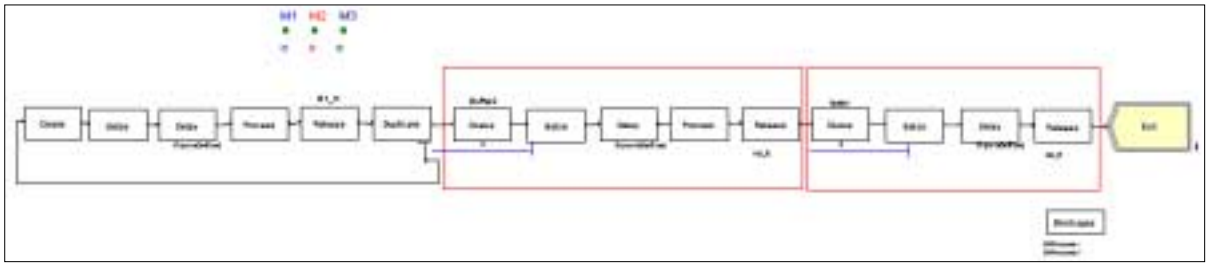
Συνολικά, για την ανάπτυξη ενός μοντέλου μιας γραμμής παραγωγής N μηχανών χρησιμοποιούμε τα εξής modules:

- *1* Create block, για τις αφίξεις κομματιών προς επεξεργασία, σαν να έλκονται από άπειρη αποθήκη
- *N-1* Queue blocks, ένα για κάθε ενδιάμεση αποθήκη
- *N* Delay blocks, ένα για κάθε μηχανή
- *N-1* Proceed blocks, για την αποστέρηση των κατάντη μηχανών όταν οι αποθήκες που προηγούνται γεμίζουν
- *N* Release blocks, για την αποδέσμευση των μηχανών (πόρων)
- *1* Dispose module, για την αναπαράσταση της εξόδου των έτοιμων προϊόντων από το σύστημα
- *1* Blockages element, όπου δηλώνονται τα σημεία μπλοκαρίσματος της κίνησης των κομματιών όταν γεμίζει μια αποθήκη.
- *1* Duplicate block, που διπλασιάζει τις οντότητες που περνούν μέσα από αυτό, ώστε οι νέες να αποτελέσουν τα νεοεισερχόμενα κομμάτια στη γραμμή (έλξη κομματιών από μία άπειρη αποθήκη).



Εικόνα 3.4.6 Το Blockages element.

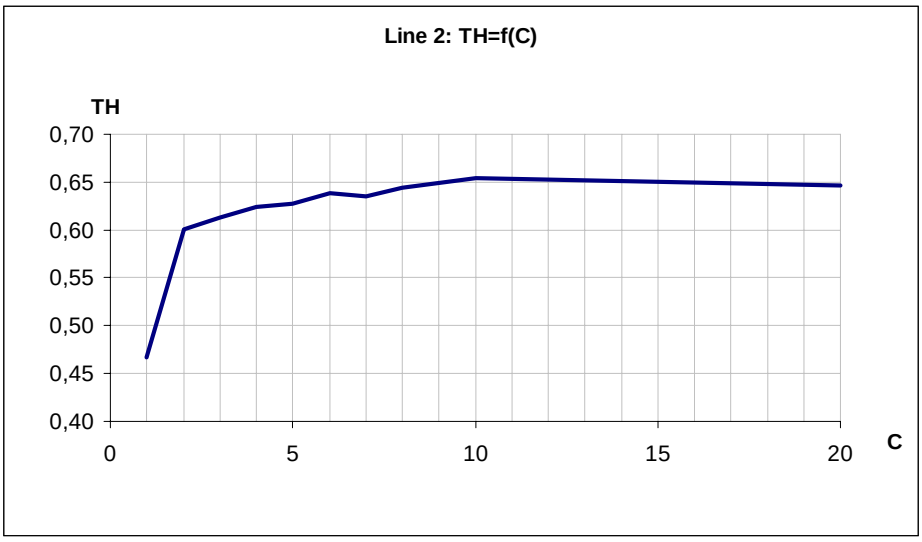
Βάσει των παραπάνω, για την μοντελοποίηση μιας γραμμής τριών φάσεων (N=3) θα επαναλαμβάναμε το δεύτερο τμήμα της γραμμής, προσαρμόζοντας την πληροφορία μέσα σε αυτά (βλ. Εικ.3.4.7).



Εικόνα 3.4.7 Το μοντέλο μιας γραμμής παραγωγής με τρεις μηχανές

TH=f(C) για την γραμμή δύο μηχανών

Τρέξαμε το μοντέλο των δύο μηχανών με κοινό ρυθμό επεξεργασίας (0.8 κομμάτια/λεπτό), για χρόνο 5000 λεπτά και για διάφορες τιμές της χωρητικότητας (C) της ενδιάμεσης αποθήκης. Στο παρακάτω διάγραμμα παρουσιάζονται τα αποτελέσματα των προσομοιώσεων ως προς τον μέσο ρυθμό παραγωγής.

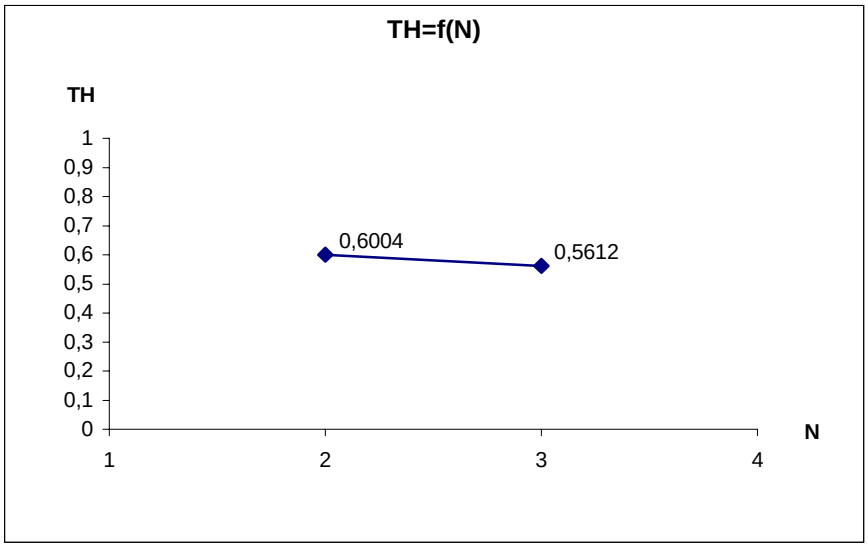


Διάγραμμα 3.1 TH=f(C) για το μοντέλο γραμμής παραγωγής δύο μηχανών

Παρατηρούμε ότι στην αρχή η παραγωγικότητα (TH) αυξάνεται με εκθετική μορφή σε σχέση με την χωρητικότητα της αποθήκης, αλλά φθάνει σε ένα επίπεδο όπου σταθεροποιείται.

TH=f(N), όπου N ο αριθμός των μηχανών

Μελετήθηκε και η μεταβολή της παραγωγικότητας, για χωρητικότητα αποθήκης C=2, σε σχέση με το μέγεθος της γραμμής, δηλαδή το πλήθος των μηχανών. Στην συνέχεια παρουσιάζονται τα αποτελέσματα για γραμμή δύο και τριών μηχανών.



Διάγραμμα 3.2 Σχέση παραγωγικότητας και μεγέθους μιας γραμμής παραγωγής

Παρατηρούμε ότι η παραγωγικότητα φθίνει με το πλήθος των μηχανών της γραμμής, γεγονός αναμενόμενο.

3.5 ΣΧΕΔΙΑΣΗ ΕΝΟΣ ΣΥΣΤΗΜΑΤΟΣ M/M/1/K - ΜΕΘΟΔΟΣ ΑΝΑΛΥΣΗΣ ΔΙΑΤΑΡΑΧΩΝ

3.5.1 Εισαγωγή

Έστω ένα υπό σχεδίαση σύστημα M/M/1/K. Αναζητείται η βέλτιστη επιλογή των ρυθμών αφίξεων (λ) και εξυπηρέτησης (μ), ώστε να μεγιστοποιείται ο μέσος ρυθμός παραγωγής του συστήματος δοθέντος ότι υπάρχει περιορισμένο κεφάλαιο C που διατίθεται για τους κόμβους εξυπηρέτησης. Για την αύξηση του ρυθμού λ κατά ένα (1) πελάτη/μονάδα χρόνου η επιχείρηση επιβαρύνεται με α χρηματικές μονάδες, ενώ για αντίστοιχη αύξηση του ρυθμού μ επιβαρύνεται με β χρηματικές μονάδες. Μαθηματικά, το πρόβλημα διατυπώνεται ως εξής:.

$$\max TH(\lambda,\mu),$$
$$\text{όταν } \alpha\lambda + \beta\mu = C$$

Οι αναγκαίες συνθήκες βέλτιστου είναι:

$$S(\lambda^*)-\lambda\alpha=0$$
$$S(\mu^*)-\mu\beta=0,$$

$$\text{όπου} \quad S(\lambda^*)=\frac{\partial TH(\lambda,\mu)}{\partial \lambda^*} \text{ και } S(\mu^*)=\frac{\partial TH(\lambda,\mu)}{\partial \mu^*}$$

3.5.2 Ο αλγόριθμος βελτιστοποίησης

Ο αλγόριθμος βελτιστοποίησης είναι ένας απλός αλγόριθμος μέγιστης ανόδου που βασίζεται στη θεωρία Ανάλυσης Διαταραχών [4], για τον υπολογισμό των μερικών παραγώγων του TH ως προς λ και μ.

Ορίζουμε τις μεταβλητές S_{11}, S_{12}, S_{21}, S_{22}, όπου S_{im} συμβολίζει την παράγωγο του χρόνου αναχώρησης του τρέχοντος κομματιού από την μηχανή m ως προς τον μέσο ρυθμό παραγωγής στη μηχανή i.

Μηχανές: 1= αφίξεις, 2= εξυπηρέτηση

Βήματα:

Α. Νέα προσομοίωση.

1. Όταν συμβεί άφιξη, $S_{11} = S_{11} - \dfrac{ενδο - αφιξιακόχρόνο}{\lambda}$
2. Όταν τελειώσει μία εξυπηρέτηση, $S_{22} = S_{22} - \dfrac{διάρκεια - εξυπηρέτησης}{\mu}$
3. Όταν γεμίσει η ουρά, $\begin{matrix} S_{11} = S_{12} \\ S_{21} = S_{22} \end{matrix}$
4. Όταν αδειάσει η ουρά και ο εξυπηρετών είναι αδρανής (Idle), $\begin{matrix} S_{12} = S_{11} \\ S_{22} = S_{21} \end{matrix}$

Β. Στο τέλος κάθε προσομοίωσης

5. Υπολόγισε

$$TH = \frac{Σύνολο\_πελατών\_που\_εξυπηρετήθηκαν}{TNOW}$$

$$S(\lambda) = S_1 = -\frac{Σύνολο\_πελατών\_που\_εξυπηρετήθηκαν}{TNOW^2} \cdot S_{12}$$

$$S(\mu) = S_2 = -\frac{Σύνολο\_πελατών\_που\_εξυπηρετήθηκαν}{TNOW^2} \cdot S_{22}$$

6. Εάν $|TH - TH_{παλιό}| < \varepsilon$ STOP (6a).

   Διαφορετικά συνέχισε εκτελώντας μία νέα προσομοίωση με νέες παραμέτρους λ και μ που υπολογίζονται από τις σχέσεις:

   (6b) $S = \dfrac{\alpha \cdot S_1 + \beta \cdot S_2}{a^2 + \beta^2}$

   $\lambda = \lambda + J \cdot [S_1 - \alpha \cdot S]$, $\mu = \mu + J \cdot [S_2 - \beta \cdot S]$,

   $TH_{παλίο}$=TH, $\kappa = \kappa + 1$, $J = \dfrac{J}{\kappa}$. Επέστρεψε στο βήμα Α.

3.5.3 Μοντελοποίηση

Στον Πίνακα 3.1 αντιστοιχίζονται οι ονομασίες των μεταβλητών και παραμέτρων του αλγορίθμου με αυτές που χρησιμοποιούνται στο μοντέλο που αναπτύσσεται στην συνέχεια.

Αλγόριθμος	Arena
S11	S11
S12	S12
S21	S21
S22	S22
λ	l
μ	SRate
TH	TH
παλιό TH	LTH
α	a_number
β	b_number
ε	limit
S	S_number
J	J_number
K (χωρητικότητα ουράς)	Q.Cap
S1	S1
S2	S2
Ενδοαφιξιακός χρόνος	TInterarrival
Διάρκεια εξυπηρέτησης	Tservice
Χρονική στιγμή τελευταίας άφιξης	LasrtArrivalT
Χρονική στιγμή έναρξης της εξυπηρέτησης	Tbefore

Πίνακας 3.3 Οι μεταβλητές και οι παραμέτρων της εφαρμογής

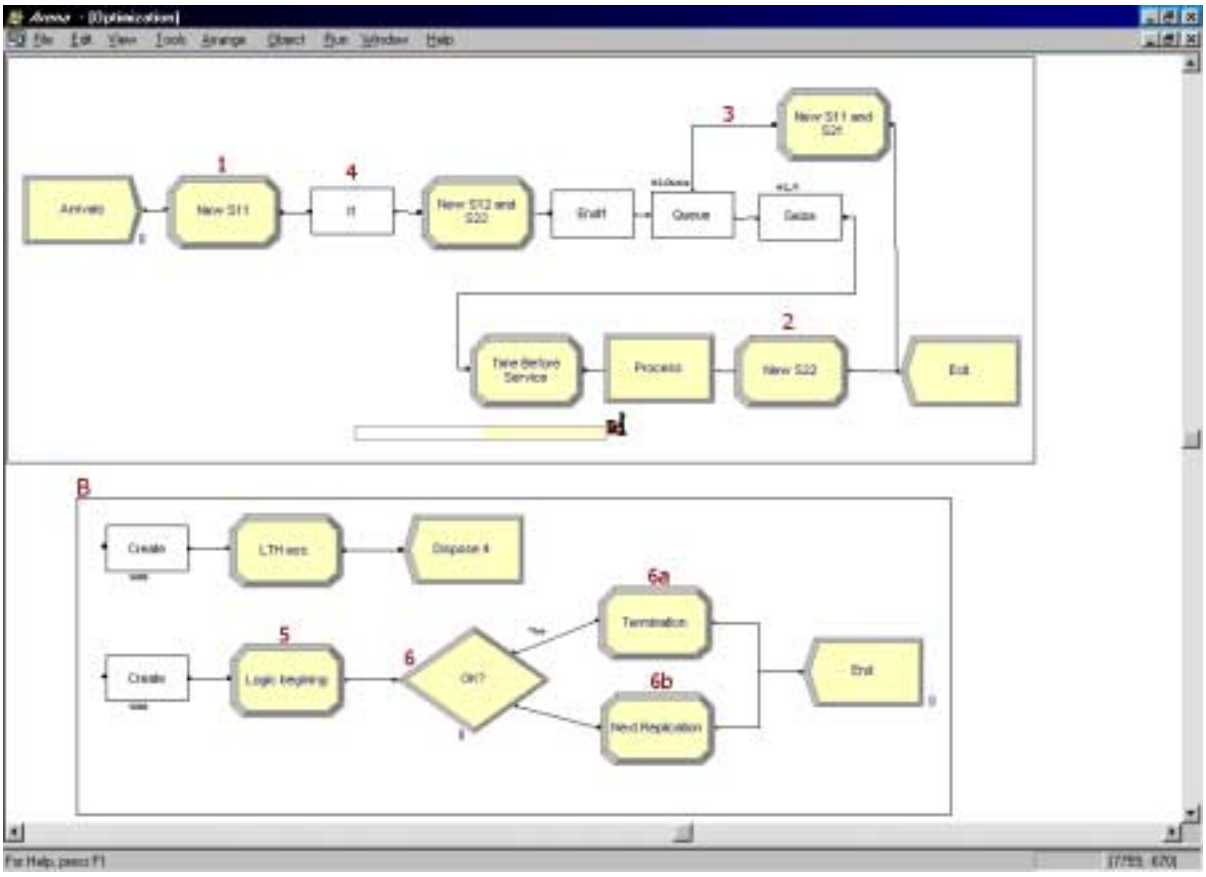
Στην Εικόνα 3.5.1 παρουσιάζεται το μοντέλο *Optimization*. Παρατηρήστε τους αριθμούς δίπλα στα modules που δηλώνουν το βήμα του αλγορίθμου που μοντελοποιείται σε αυτά. Πρόκειται για ένα M/M/1/K που, όπως είδαμε στην παράγραφο 3.1, αποτελείται από τα εξής μέρη:

- ένα Create για τις αφίξεις (Arrivals)
- ένα Queue block για την ουρά πεπερασμένης χωρητικότητας
- ένα Seize block για την δέσμευση του εξυπηρετούντος
- ένα Process για την εξυπηρέτηση των πελατών και την αποδέσμευση του εξυπηρετούντος
- δύο Dispose modules. Εδώ χρησιμοποιούμε μόνο ένα για λόγους οικονομίας, τόσο για την μοντελοποίηση της εξόδου των πελατών που εξυπηρετούνται, όσο και για την έξοδο των πελατών που βρήκαν το σύστημα γεμάτο κατά την άφιξη τους και απορρίφθηκαν (Exit). Στο πάνω τμήμα μοντελοποιείται το σύστημα M/M/1/K και το μέρος Α του αλγορίθμου βελτιστοποίησης, ενώ το κάτω πλαίσιο χρησιμοποιείται για την εφαρμογή του Β μέρους.

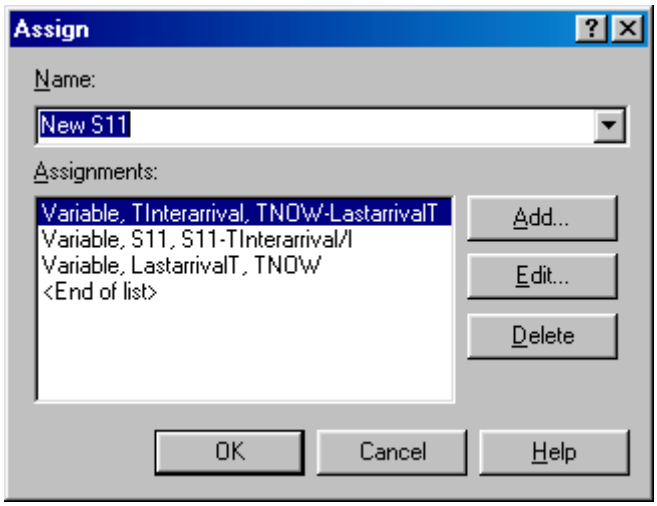
Μέρος Α (Βήματα 1-4)

1. Δεξιά του Create module *Arrivals* τοποθετούμε ένα Assign module για την ενημέρωση της S_{11}. Ο ενδοαφιξιακός χρόνος ισούται με την διαφορά του χρόνου της τελευταίας άφιξης (LastArrivalT) από τον τωρινό (TNOW).

$$TInterarrival = TNOW - LastArrivalT$$



Εικόνα 3.5.1 Το μοντέλο M/M/1/K με εφαρμογή της Μεθόδου Ανάλυσης Διαταραχών

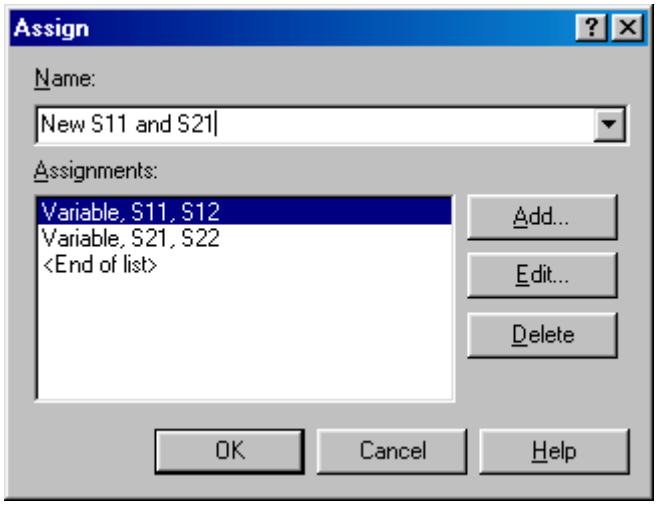


Εικόνα 3.5.2 Το πρώτο βήμα του αλγορίθμου βελτιστοποίησης

2. Δεξιά του module της εξυπηρέτησης *Process* τοποθετούμε ένα Assign module για την ενημέρωση της S_{22} (βλ. Εικ.3.5.4). Η διάρκεια της εξυπηρέτησης θα ισούται με την διαφορά του χρόνου που αυτή ξεκίνησε μείον τον τωρινό χρόνο.

Εικόνα 3.5.3 Το βήμα 2 του αλγορίθμου βελτιστοποίησης

3. Το βήμα 3 ενεργοποιείται όταν γεμίσει η ουρά. Για τον λόγο αυτό τοποθετείται ένα Assign module στην διαδρομή των απορριφθέντων πελατών προς την έξοδο του συστήματος (εικ. 3.3.5).



Εικόνα 3.5.4 Το βήμα 3

4. Εδώ χρησιμοποιούμε τη δομή If-Endif. Πριν την είσοδο των πελατών στην ουρά, τοποθετούμε ένα **If block** στο οποίο κάνουμε τον έλεγχο αυτού του βήματος (βλ. Εικ.3.3.5), που σε γλώσσα SIMAN γράφεται:

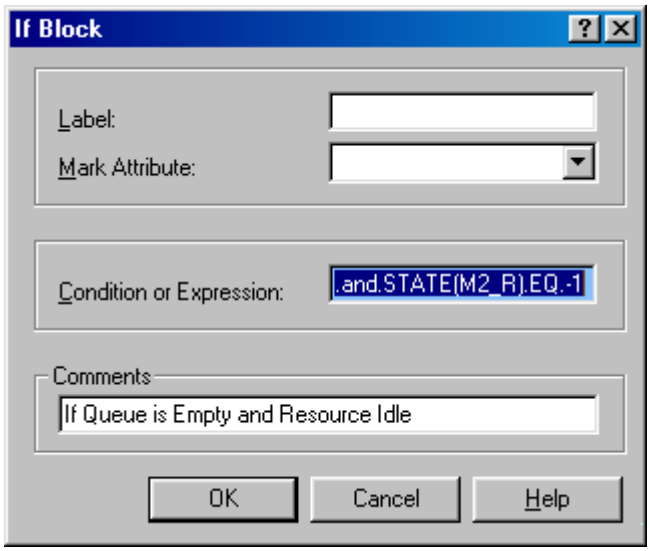
NQ(M2.Queue).EQ.0.and.STATE(M2_R).EQ.-1, όπου

NQ(x)=Number in Queue=στάθμη της ουράς x

EQ=equal=ισούται

STATE(x).EQ.-1= η κατάσταση του πόρου x είναι Idle

Σύνολο Καταστάσεων: {-1=Idle, -2=Busy, -3=Inactive, -4=Failed}.



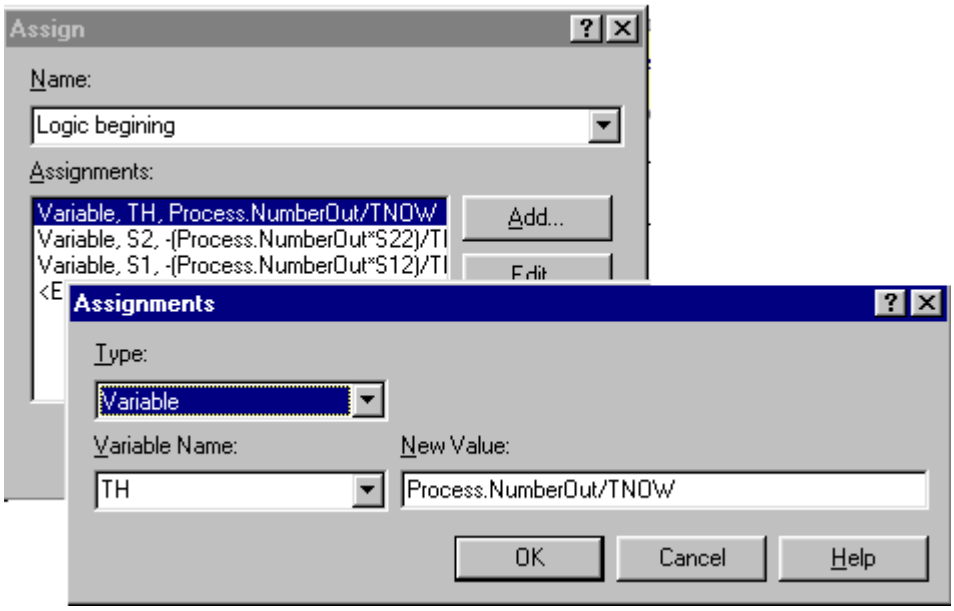
Εικόνα 3.5.5 Ο έλεγχος του βήματος 4

Εάν ισχύσει ο έλεγχος, τότε έπεται το module *New S12 and S22* διαφορετικά ακολουθεί το *Endif* χωρίς να αλλάξει τίποτα. Τα παραπάνω τέσσερα βήματα επανελέγχονται καθ' όλη τη διάρκεια μιας προσομοίωσης σύμφωνα με την κίνηση των πελατών (Customers) μέσα στο σύστημα. Ακολουθεί το μέρος Β του αλγορίθμου.

Μέρος Β (Βήματα 5-6)

Δημιουργούμε μία δομή με την οποία θα υλοποιήσουμε το δεύτερο μέρος (Β).

5. Τοποθετούμε ένα Create block, επιλέγοντας η πρώτη άφιξη να γίνει το λεπτό 5000, δηλαδή στο τέλος της πρώτης προσομοίωσης, ενώ οι επόμενες οντότητες να αφικνούνται με ενδοαφιξιακό χρόνο 5000 λεπτά. Επιλέγουμε ένα όνομα για τις οντότητες που θα δημιουργούνται (π.χ. *Check ent*), διαφορετικό από αυτό των πελατών (*Customer*), για να μην υπάρξει σύγχυση των στατιστικών των δύο αφίξεων. Στην συνέχεια τοποθετούμε ένα Assign module, όπου ενημερώνονται οι μεταβλητές του βήματος 5.



Εικόνα 3.5.6 Το βήμα 5

6. Ακολουθεί το Decide module *OK* στο οποίο εκτελείται ο έλεγχος βέλτιστου. Η μεταβλητή LTH έχει πάρει την τιμή της TH της προηγούμενης προσομοίωσης, όπως εξηγείται στην συνέχεια.

Εικόνα 3.5.7 Το Βήμα 6. Έλεγχος βέλτιστου

6a. Μέσω ενός Assign module (Termination), η τρέχουσα προσομοίωση (NREP) τίθεται ως η τελευταία (MREP), οπότε έμμεσα τερματίζεται η εκτέλεση.

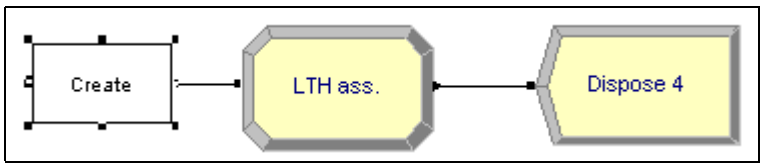
$$MREP = NREP$$

6b. Εάν δεν ισχύσει η συνθήκη βέλτιστου, τότε υπολογίζονται οι μεταβλητές TH, S, λ, μ και J από άλλο ένα Assign module και το μοντέλο ξεκινά την επόμενη προσομοίωση.

Μεταβλητή	*Νέα Τιμή*	*Αντιστοιχία με αλγόριθμο*
Temp	TH	
S_number	(a_number*S1+b_number*S2)/(a_number**2+b_number**2)	$S = \frac{\alpha \cdot S_1 + \beta \cdot S_2}{a^2 + \beta^2}$
l	l+J_number*(S1-a_number*S_number)	$\lambda = \lambda + J \cdot [S_1 - \alpha \cdot S]$
SRate	SRate+J_number*(S2-b_number*S_number)	$\mu = \mu + J \cdot [S_2 - \beta \cdot S]$
J_number	J_number/(NREP+1)	$J = J/K$

Πίνακας 3.4 Ενημέρωση μεταβλητών πριν την εκκίνηση της επόμενης προσομοίωσης

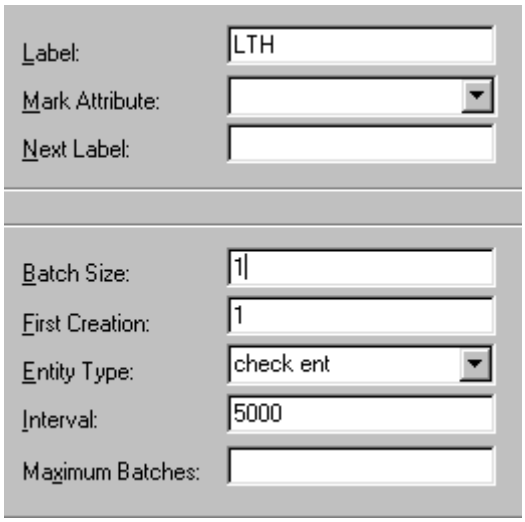
Η Μέθοδος Ανάλυσης Διαταραχών χρησιμοποιεί δύο τιμές του μέσου ρυθμού παραγωγής (TH)· την τιμή που αντιστοιχεί στην τρέχουσα προσομοίωση και αυτή της προηγούμενης. Έτσι, στο τέλος κάθε προσομοίωσης (λεπτό 5000) αποθηκεύουμε την τρέχουσα τιμή του TH στη μεταβλητή Temp για μελλοντική χρήση. Στην αρχή της επόμενης προσομοίωσης (το $1^{ο}$ λεπτό) ενημερώνουμε τη μεταβλητή LTH με την τιμή Temp. Έτσι, γνωρίζουμε τις δύο τιμές του TH που χρειάζονται για το βήμα 6.



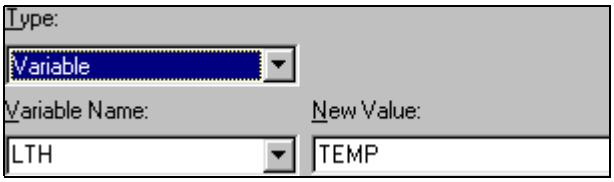
Εικόνα 3.5.8 Τα απαιτούμενα modules για την ενημέρωση της μεταβλητής LTH στην αρχή κάθε προσομοίωσης.

Σχηματικά έχουμε:

- $TH_n \rightarrow Temp$
- $Temp \rightarrow LTH_{n+1}$



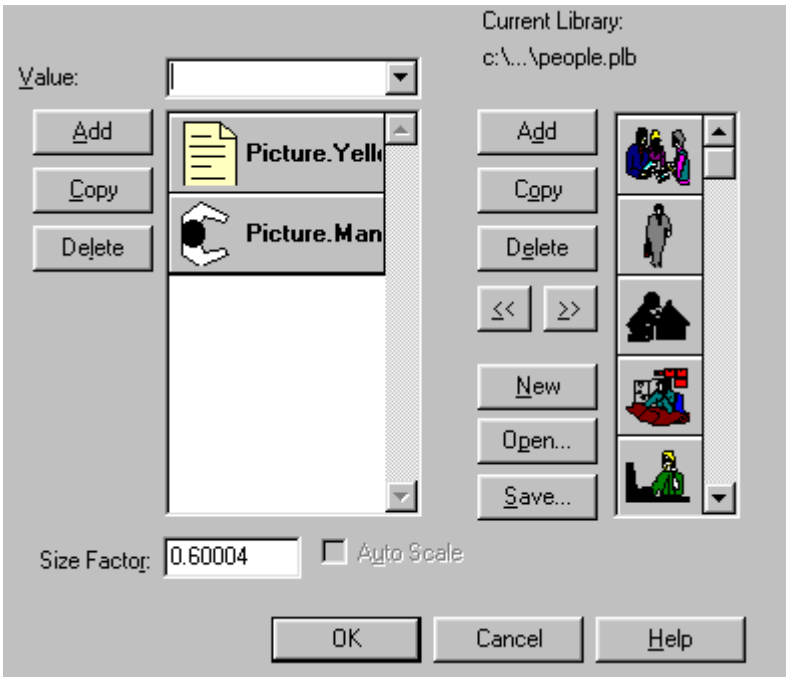
Εικόνα 3.5.9 Δημιουργία μιας οντότητας με την οποία θα αποθηκευτεί η τιμή του TH της προηγούμενης προσομοίωσης (LTH)



Εικόνα 3.5.10 Διαμόρφωση του Assign module *LTH. ass*

3.5.4 Κίνηση οντοτήτων στο μοντέλο

Μπορούμε να καθορίσουμε την όψη των οντοτήτων του μοντέλου. Από Edit/Entity Pictures επιλέγουμε τις εικόνες που επιθυμούμε. Κατόπιν από Basic Process/ Entity ορίζουμε τις εικόνες που επιλέξαμε σαν εικόνες των οντοτήτων.



Εικόνα 3.5.11 Ορισμός των επιθυμητών απεικονίσεων των οντοτήτων του μοντέλου

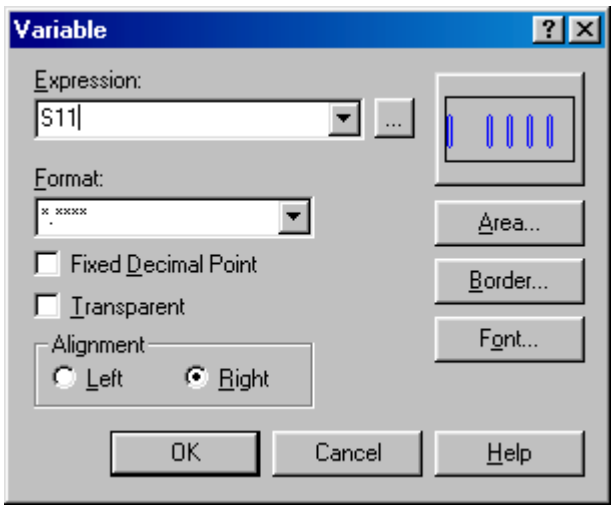
Entity - Basic Process

	Entity Type	Initial Picture
1	check ent	Picture.Yellow Page
2	Customer	Picture.Man

Εικόνα 3.5.12 Επιλογή μιας εικόνας για κάθε οντότητα

3.5.5 Παρατήρηση της εξέλιξης των μεταβλητών κατά την διάρκεια της προσομοίωσης

Όπως εισάγαμε απεικονίσεις για τις οντότητες, μπορούμε να κάνουμε το ίδιο και για τις μεταβλητές. Με τη γραμμή εργαλείων *Animate* και το πλήκτρο Variable επιλέγουμε ποιες μεταβλητές ή παράμετροι θα εισάγουμε.



Εικόνα 3.5.13 Εισαγωγή απεικόνισης της μεταβλητής S11

3.5.6 Εφαρμογή-Αποτελέσματα

Έως αυτό το σημείο έχει ολοκληρωθεί το κύριο κομμάτι της μοντελοποίησης. Μένει να ορισθούν οι επιπλέον στατιστικές που ο χρήστης επιθυμεί, να δοθούν (αρχικές) τιμές σε όλες τις παραμέτρους του μοντέλου και, τέλος, να εκτελεστεί το πρόγραμμα.

Εφαρμόσαμε το παραπάνω πρόβλημα με τις εξής τιμές των παραμέτρων:

l (λ)	5
Srate (μ)	71.2
a_number (α)	2.247
b_number (β)	1.247
limit (ε)	0,01
J_number (J)	100
Q.Cap (K)	5

Πίνακας 3.5 Αρχικές τιμές των παραμέτρων του μοντέλου στην εφαρμογή

Κάθε προσομοίωση διαρκεί 5000 λεπτά, ενώ δίνεται ένας πρακτικά άπειρος αριθμός, ως μέγιστος αριθμός προσομοιώσεων:

Replication Length	5000
Number of Replications	999999

Πίνακας 3.6 Διάρκεια και μέγιστος αριθμός προσομοιώσεων

Στην εικόνα που ακολουθεί παρουσιάζονται τα αποτελέσματα της εκτέλεσης.

User Specified 9:52:44ìì
Αýãïõóôïõ 26, 2001

Unnamed Project Replications: 3

Replication 1 Start Time: 0,00 Stop Time: 5.000,00 Time Units: Minutes

Time Persistent

Variable	Average	Half Width	Minimum	Maximum
I	5.0000	(Insufficient)	5.0000	28.1938
LTH	0	(Insufficient)	0	0
SRate	71.2000	(Insufficient)	29.4066	71.2000
TH	0.00100000	(Insufficient)	0.00100000	4.9252
TService	0.01313619	(Correlated)	0	0.1670

Replication 2 Start Time: 0,00 Stop Time: 5.000,00 Time Units: Minutes

Time Persistent

Variable	Average	Half Width	Minimum	Maximum
I	28.1938	(Insufficient)	28.1938	28.3763
LTH	4.9242	(Insufficient)	0	4.9252
SRate	29.4066	(Insufficient)	29.0777	29.4066
TH	4.9252	(Insufficient)	4.9252	24.6312
TService	0.03146027	0,00031047 3	0	0.3966

Replication 3 Start Time: 0,00 Stop Time: 5.000,00 Time Units: Minutes

Time Persistent

Variable	Average	Half Width	Minimum	Maximum
I	28.3763	(Insufficient)	28.3763	28.3763
LTH	24.6273	(Insufficient)	4.9252	24.6312
SRate	29.0777	(Insufficient)	29.0777	29.0777
TH	24.6312	(Insufficient)	24.5778	24.6312
TService	0.03194459	(Correlated)	0	0.4708

Εικόνα 3.5.14 Τα αποτελέσματα

Η συνθήκη βέλτιστου ίσχυσε μετά από τρεις (3) προσομοιώσεις. Οι τιμές των λ και μ είναι περίπου ίσες. Αυτό ήταν το αναμενόμενο αποτέλεσμα, αφού η θεωρία της μεθόδου θέλει τους ρυθμούς παραγωγής/εξυπηρέτησης ενός σειριακού συστήματος ίσους κατά βέλτιστο.

3.6 ΕΝΑ ΣΥΣΤΗΜΑ ΣΥΝΑΡΜΟΛΟΓΗΣΗΣ

3.6.1 Περιγραφή

Έστω δύο γραμμές παραγωγής. Στη γραμμή 1 που αποτελείται από την μηχανή M1 εισέρχεται πρώτη ύλη και παράγεται προϊόν τύπου a, ενώ στη γραμμή 2 (M2) παράγεται προϊόν τύπου b. Τα προϊόντα a,b ενώνονται στο σημείο συναρμολόγησης δίνοντας το τελικό προϊόν (c). Πριν το σημείο συναρμολόγησης υπάρχουν δύο αποθήκες. Στη μία (Buffer 1) παραμένουν τα κομμάτια a μέχρι να χρησιμοποιηθούν στην συναρμολόγηση. Αντίστοιχα, τα κομμάτια b παραμένουν στην αποθήκη 2 (Buffer 2).

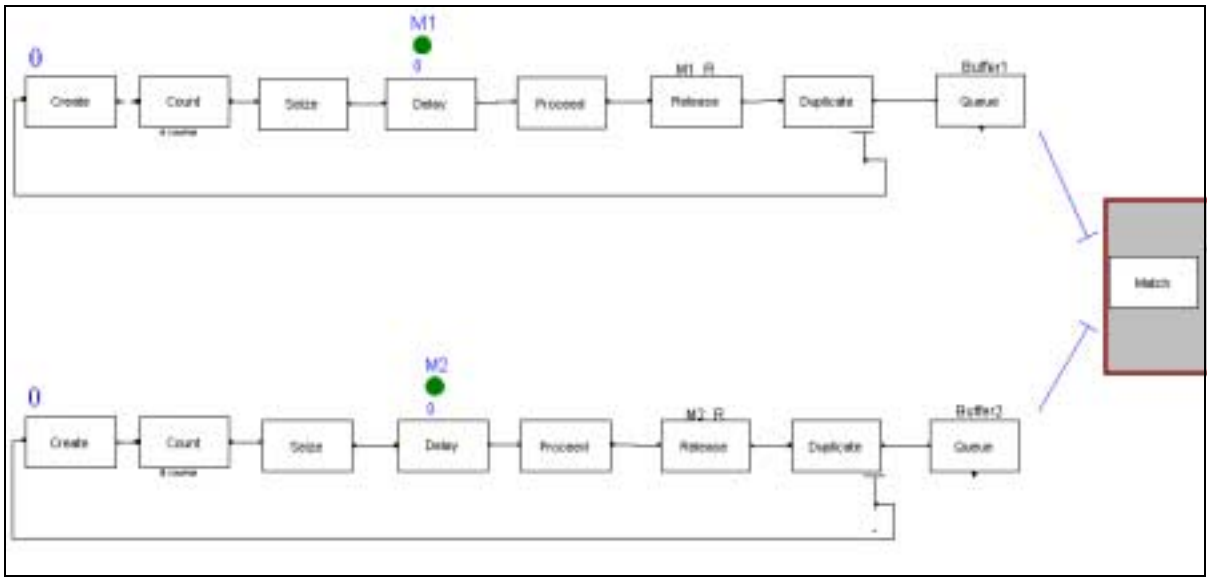
Όταν γεμίσει η αποθήκη 1, σταματούν να παράγονται κομμάτια a. Το ίδιο ισχύει και για τα κομμάτια b, όταν γεμίζει η αποθήκη 2. Υποθέτουμε ότι η συναρμολόγηση διαρκεί πολύ λιγότερο από την μέση διάρκεια παραγωγής του προϊόντος a είτε του b.

3.6.2 Μοντελοποίηση

Η μοντελοποίηση του παραπάνω συστήματος μπορεί να χωριστεί σε τρία μέρη:

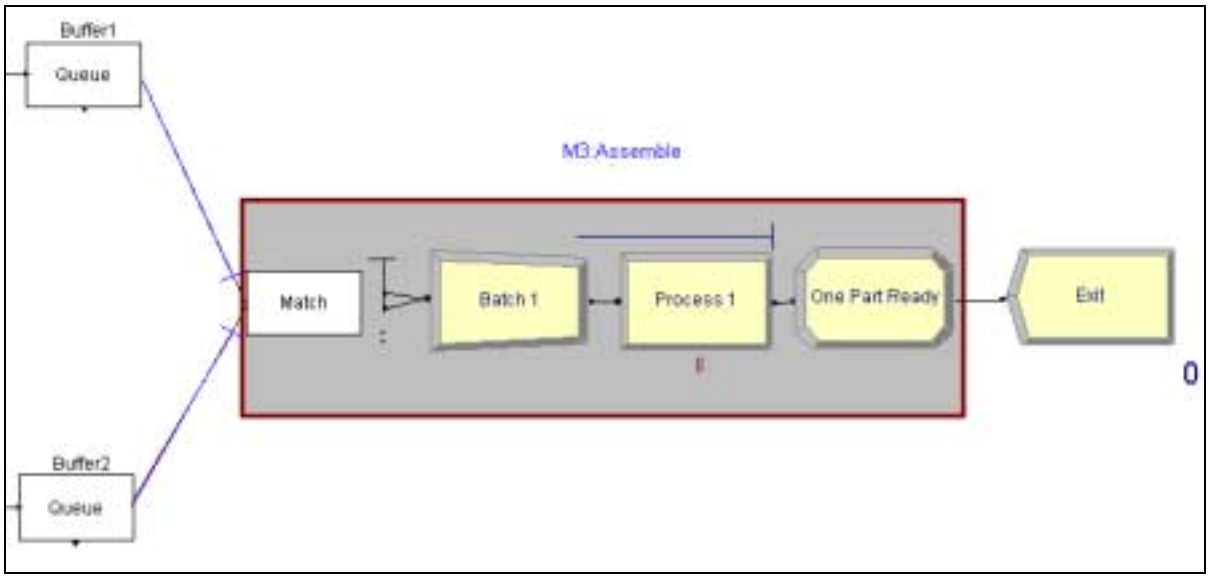
- στην αναπαράσταση της γραμμής 1 με την αποθήκη των κομματιών a
- την αναπαράσταση της γραμμής 2 με την αποθήκη των κομματιών b και
- την αναπαράσταση της συναρμολόγησης (παραγωγή προϊόντος c)

Για τη μοντελοποίηση των δύο πρώτων μερών ακολουθούμε τη μέθοδο της παραγράφου 3.4. Έτσι, ο αναγνώστης παραπέμπεται στην παράγραφο αυτή. Ωστόσο, προστέθηκε ένα **Count block** μετά τις αφίξεις κάθε γραμμής (counter1, counter2), επειδή με τον τρόπο αυτό μπορούμε να γνωρίζουμε τον αριθμό τους. Τοποθετούμε και το αντίστοιχο **Counters element**, όπου ορίζουμε τους δύο μετρητές που εισάγαμε. Αν δεν χρησιμοποιήσουμε το στοιχείο αυτό θα λάβουμε μήνυμα σφάλματος, το οποίο θα αναφέρει την έλλειψη του.



Εικόνα 3.6.1 Οι γραμμές 1 και 2 συναντώνται στον σταθμό συναρμολόγησης

Για την συναρμολόγηση χρησιμοποιήθηκε ένα **Match block**, το οποίο στέλνει ένα ζεύγος προϊόντων a,b στο επόμενο module που είναι τύπου **Batch**. Εκεί, τα δύο κομμάτια μετατρέπονται σε ένα[7] που κατά μία έννοια είναι η λειτουργία της συναρμολόγησης.



Εικόνα 3.6.2 Συναρμολόγηση

Όμως ένα Batch module λειτουργεί χωρίς χρήση κάποιου πόρου. Επειδή στόχος είναι η μοντελοποίηση μιας διαδικασίας συναρμολόγησης την οποία να εκτελεί κάποιος πόρος (M3) προσθέτουμε ένα Process module. Στο Process module μοντελοποιείται η διάρκεια της συναρμολόγησης καθώς και η χρήση της μηχανής συναρμολόγησης (M3_R). Το μοντέλο ολοκληρώνεται με την αναπαράσταση της εξόδου τους συστήματος με χρήση ενός Dispose module (Exit).

Σημειώνεται η χρήση της έμμεσης σύνδεσης μεταξύ δύο modules. Στα Queue blocks των δύο αποθηκών επιλέγουμε **Detach** και δηλώνουμε το προορισμό των κομματιών. Το Match block δεν συνδέεται παρά μόνο έμμεσα με άλλα modules (προσπαθώντας να συνδέσουμε ένα Queue block με το Match διαπιστώνουμε ότι αυτό δεν είναι δυνατόν).

3.6.3 Παράμετροι μοντελοποίησης

Όπως στα περισσότερα μοντέλα της εργασίας, έτσι και εδώ ορίζουμε κάποια επιπλέον μέτρα απόδοσης. Στον πίνακα που ακολουθεί παρουσιάζονται οι παράμετροι του συστήματος, με τις αντίστοιχες ονομασίες τους στο μοντέλο.

Αποθήκη 1	Buffer1	Για τα κομμάτια a
Αποθήκη 2	Buffer2	Για τα κομμάτια b
Ουρά Συναρμολόγησης	Process1.Queue	Ζεύγη a,b σε αναμονή για συναρμολόγηση
Μηχανή 1	M1_R	Παραγωγή κομματιών a

[7] Η αντίστροφη ενέργεια, δηλαδή μία οντότητα να μετατραπεί σε πολλές, γίνεται από το Separate module της ίδιας κατηγορίας.

Μηχανή 2	M2_R	Παραγωγή κομματιών b
Μηχανή 3	M3_R	Παραγωγή κομματιών c
	1/outRate1	Μέση διάρκεια παραγωγής ενός κομματιού a
	1/outRate2	Μέση διάρκεια παραγωγής ενός κομματιού b
	1/outRate3	Μέση διάρκεια παραγωγής ενός κομματιού c
	c1	Χωρητικότητα $1^{ης}$ αποθήκης
	c2	Χωρητικότητα $2^{ης}$ αποθήκης

Πίνακας 3.7 Οι παράμετροι του συστήματος συναρμολόγησης

Statistic - Advanced Process

	Name	Type	Expression	Report Label	Output File
1	TH	Output	Exit.NumberOut/TFIN	TH	
2	Total out	Output	Exit.NumberOut	C parts Assembled	
3	Buffer 1 last value	Output	NQ(Buffer1)	Level of Buffer 1	
4	Buffer 2 last value	Output	NQ(Buffer2)	Level of Buffer 2	
5	Line 1 arrived	Output	NC(1)	Line 1 arrived	
6	Line 2 arrived	Output	NC(2)	Line 2 arrived	

Double-click here to add a new row.

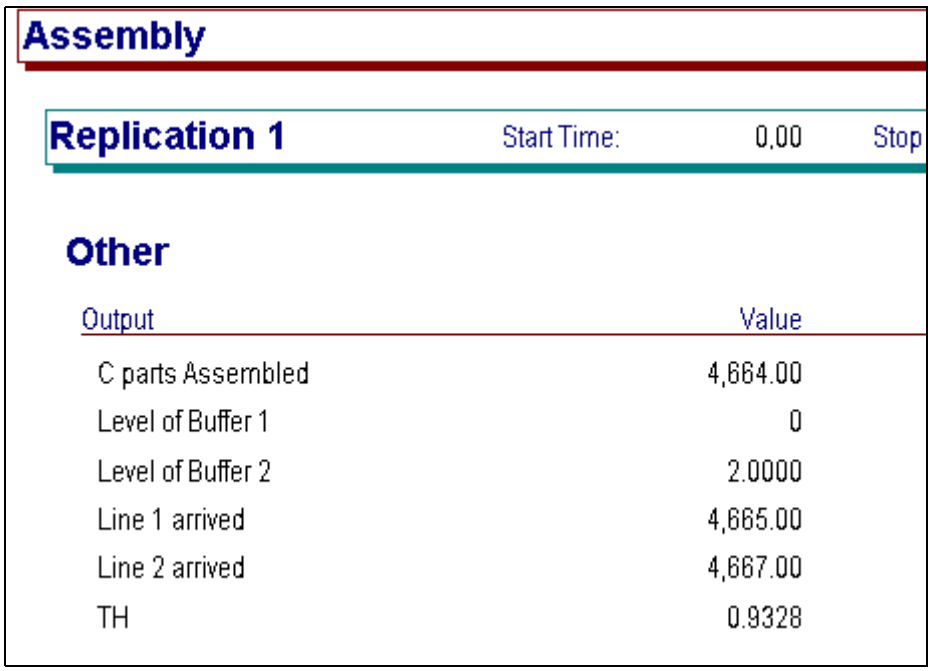
Εικόνα 3.6.3 Ορισμός επιπλέον στατιστικών για το μοντέλο

Εφαρμογή

Το πρόγραμμα εκτελέστηκε με τα εξής αριθμητικά δεδομένα:

- OutRate1=1
- OutRate2=1
- OutRate3=50
- C1=5
- C2=5
- Αριθμός προσομοιώσεων: 1
- Διάρκεια προσομοίωσης (TSIM): 5000 λεπτά.

Στην εικόνα που ακολουθεί παρουσιάζονται τα ορισμένα από τον χρήστη αποτελέσματα της εκτέλεσης (User Specified).



Assembly

Replication 1 Start Time: 0,00 Stop

Other

Output	Value
C parts Assembled	4,664.00
Level of Buffer 1	0
Level of Buffer 2	2.0000
Line 1 arrived	4,665.00
Line 2 arrived	4,667.00
TH	0.9328

Εικόνα 3.6.4 Μερικά αποτελέσματα της προσομοίωσης

4. ΣΥΝΟΨΗ

Στην εργασία αυτή παρουσιάστηκαν οι δυνατότητες του προγράμματος Arena στην προσομοίωση ουρών αναμονής και ανεπτύχθησαν εφαρμογές σε προβλήματα ανάλυσης απλών συστημάτων παραγωγής, γραμμών παραγωγής και συστημάτων συναρμολόγησης.

Με την ενσωμάτωση ενός κώδικα βελτιστοποίησης και ενός κώδικα υπολογισμού των παραγώγων του ρυθμού παραγωγής ενός συστήματος ως προς τις παραμέτρους του, αναπτύχθηκε επίσης μία ακόμη εφαρμογή που λύνει το πρόβλημα της επιλογής μηχανών σε μια γραμμή παραγωγής με δύο μηχανές και ενδιάμεση αποθήκη.

Αρχικός σκοπός της εργασίας ήταν να αποτελέσει εγχειρίδιο για τις ασκήσεις εργαστηρίου του μαθήματος *Προσομοίωση* που διδάσκεται στο 9^{ο} εξάμηνο του προγράμματος σπουδών του τμήματος Μ.Π.Δ. του Πολυτεχνείου Κρήτης.

Ωστόσο η ευκολία στη χρήση του προγράμματος και τα εργαλεία που διαθέτει, καθιστούν το πρόγραμμα Arena ένα αποτελεσματικό εργαλείο τόσο για ερευνητικές όσο και για πρακτικές εφαρμογές. Οι επεκτάσεις που μπορούν να γίνουν στα μοντέλα που παρουσιάστηκαν στην εργασία είναι πρακτικά απεριόριστες. Μερικές έχουν ήδη αναπτυχθεί αλλά θεωρήθηκε σκόπιμο να μην παρουσιαστούν στην εργασία προκειμένου να μην κουρασθεί ο αναγνώστης, αφού ο αρχικός στόχος της εργασίας έχει εκπληρωθεί.

ΠΑΡΑΡΤΗΜΑ

1. ΚΑΤΑΝΟΜΕΣ

Η κατανομή Poisson
Μία διακριτή τυχαία μεταβλητή (τ.μ.) x ακολουθεί την κατανομή Poisson με μέση τιμή λ όταν

$$P(X = x) = \frac{e^{-\lambda} \cdot \lambda^{x}}{x!}$$

Η εκθετική κατανομή (expo)
Μία συνεχής τ.μ. ακολουθεί την εκθετική κατανομή με μέση τιμή 1/μ όταν η *συνάρτηση πυκνότητας πιθανότητας* της (σ.π.π.) είναι:

$$f(x) = \lambda \cdot e^{-\lambda x}, \quad x > 0$$

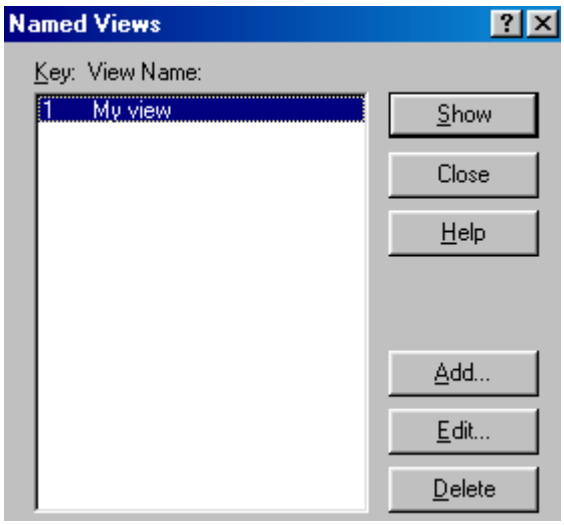
Σημαντική ιδιότητα
Για ένα φαινόμενο που εμφανίζεται κατά Poisson με μέσο ρυθμό λ εμφανίσεις/μονάδα χρόνου οι χρόνοι μεταξύ διαδοχικών εμφανίσεων ακολουθούν εκθετική κατανομή με μέση τιμή 1/λ.

2. ΟΙ ΟΨΕΙΣ ΕΝΟΣ ΜΟΝΤΕΛΟΥ

Το πλαίσιο *Navigate* της περιοχής *Project* Bar περιέχει όλες τις όψεις ενός μοντέλου που ορίζει ο χρήστης. Οι διάφορες όψεις βοηθούν ώστε να είναι ευκολότερη η παρατήρηση του μοντέλου, ιδιαίτερα κατά την εκτέλεση του (Run Mode).

Παράδειγμα εισαγωγής μιας νέας όψης
Για να ορίσουμε μία νέα όψη που να λέγεται *My view* και να εμφανίζεται με το πλήκτρο "1" ακολουθούμε τα εξής βήματα:

- εισάγουμε τα εικονίδια "View"  από View/Toolbars/View
- επιλέγουμε την επιθυμητή περιοχή με χρήση του "View Region"  καθώς και των παραπάνω εικονιδίων
- με το εικονίδιο "Manage Named Views"  εισάγουμε την πληροφορία της εικόνας που ακολουθεί.
- κλείνουμε με OK.



3. ΤΥΠΟΙ ΓΙΑ ΤΑ ΣΥΣΤΗΜΑΤΑ M/M/1/K ΚΑΙ M/M/m/K

Το σύστημα M/M/1/K [3]

$$p(n) = \frac{(1-\rho)\rho^n}{1-\rho^{k+1}} \quad \text{if } \lambda \neq \mu$$

$$p(n) = \frac{1}{K+1} \quad \text{if } \lambda = \mu$$

$$E(N) = \rho\left[\frac{1-(K+1)\rho^K + K\rho^{K+1}}{(1-\rho)(1-\rho^{K+1})}\right] \quad \text{if } \lambda \neq \mu$$

$$E(N) = \frac{K}{2} \quad \text{if } \lambda = \mu$$

$$E(L) = E(N) - (1 - p(0))$$

$$E(W) = E(L)/\lambda(1 - p(K))$$

όπου E(L)=μέσο πλήθος πελατών στην ουρά,
E(W)=μέσος χρόνος αναμονής στην ουρά,
E(N)=μέσο πλήθος πελατών στο σύστημα,
E(T)=μέσος χρόνος παραμονής στο σύστημα και ρ=λ/μ

Το σύστημα M/M/C/K [3]

$$p(n) = p(0)\frac{\rho^n}{n!} \qquad \text{for } n \leqslant C$$

$$p(n) = p(0)\frac{\rho^C}{C!}\left(\frac{\rho}{C}\right)^{n-C} \qquad \text{for } C \leqslant n \leqslant K$$

$$p(0) = \left[\sum_{n=0}^{C}\frac{\rho^n}{n!} + \frac{\rho^C}{C!}\sum_{n=1}^{K-C}\left(\frac{\rho}{C}\right)^n\right]^{-1}$$

$$E(N) = E(L) + \sum_{n=0}^{C-1} np(n) + C\left(1 - \sum_{n=0}^{C-1} p(n)\right)$$

$$E(L) = p(0)\frac{\rho^{C+1}/C}{C!(1-\rho/C)^2}\left[1 - \left(\frac{\rho}{C}\right)^{k-C+1} - K - C + 1\left(\frac{\rho}{C}\right)^{K-C}\left(1 - \frac{\rho}{C}\right)\right]$$

$$E(W) = E(L)/\lambda(1 - p(K))$$

$$E(T) = E(N)/\lambda(1 - p(K))$$

όπου E(L)=μέσο πλήθος πελατών στην ουρά,
E(W)=μέσος χρόνος αναμονής στην ουρά,
E(N)=μέσο πλήθος πελατών στο σύστημα,
E(T)=μέσος χρόνος παραμονής στο σύστημα και ρ=λ/μ

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] W. D. Kelton, R. P. Sadowski and D. A. Sadowski, *Simulation with Arena*, McGraw-Hill, Boston, Massachusetts, 1998.

[2] Β. Σ. Κουϊκόγλου, *Προσομοίωση. Σημειώσεις μαθήματος*, Πολυτεχνείο Κρήτης, Χανιά, 2001.

[3] E. Gelende and G. Pujolle, *Introduction to Queuing Networks*, 2nd ed., John Wiley & Sons, Chicester,1998.

[4] Y. C. Ho and X. R. Cao, "Perturbation analysis and optimization of queuing networks," *Journal of Optimization Theory and Applications*, vol. 40, issue 4, pp. 559-582, 1983.