
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ


**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# "ΠΡΟΣΟΜΟΙΩΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΙΑΣ ΞΕΝΟΔΟΧΕΙΑΚΗΣ ΜΟΝΑΔΑΣ ΜΕ ΤΗ ΧΡΗΣΗ ΤΟΥ ΕΡΓΑΛΕΙΟΥ JAAMSIM"


ΒΑΣΙΛΕΙΟΣ ΜΠΟΚΑΣ – ΜΑΡΚΟΣ ΛΙΟΣΗΣ



ΠΑΤΡΑ 2021

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα xx/xx/2021

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.

2.

3.

Υπεύθυνη Δήλωση Φοιτητή

*Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.*

*Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.*

# Περίληψη

Η προσομοίωση συστημάτων διακριτών συμβάντων αναπαριστάνει τη μοντελοποίηση, την προσομοίωση και την ανάλυση συστημάτων που χρησιμοποιούν τις υπολογιστικές και μαθηματικές τεχνικές, ενώ δημιουργεί ένα μοντέλο από ένα εννοιολογικό πλαίσιο που περιγράφει ένα σύστημα. Ο χρήστης του συστήματος αναλύει την έξοδο της προσομοίωσης ή των προσομοιώσεων για να εξαγάγει συμπεράσματα που θα τον βοηθήσουν στη λήψη αποφάσεων. Οι τεχνολογίες προσομοίωσης συστημάτων διακριτών συμβάντων έχουν χρησιμοποιηθεί εκτενώς από τη βιομηχανία και τον ακαδημαϊκό χώρο για την αντιμετώπιση διαφόρων βιομηχανικών προβλημάτων.

Η εννοιολογική μοντελοποίηση είναι θεμελιώδες βήμα σε ένα έργο προσομοίωσης. Ωστόσο, μόνο πρόσφατα οι δομημένες προσεγγίσεις, για τον ορισμό και τη διαμόρφωση εννοιολογικών μοντέλων, απέκτησαν σημασία στην κοινότητα προσομοίωσης συστημάτων διακριτών συμβάντων. Όμως, τόσο η οργάνωση των συνιστωσών του μοντέλου όσο και ο προσδιορισμός της συμπεριφοράς και του ελέγχου του συστήματος από τυπικές προσεγγίσεις της εννοιολογικής μοντελοποίησης έχουν ελλείψεις που περιορίζουν την εφαρμοσιμότητά της στην προσομοίωση συστημάτων διακριτών συμβάντων.

Το JaamSim είναι ένα δωρεάν πακέτο προσομοίωσης ανοιχτού κώδικα γραμμένο στη γλώσσα προγραμματισμού Java. Παρέχεται μια σύγχρονη γραφική διεπαφή χρήστη που είναι συγκρίσιμη με αυτή του εμπορικού λογισμικού, συμπεριλαμβανομένης της δημιουργίας μοντέλων με μεταφορά και απόθεση, ενός επεξεργαστή εισόδου, ενός προβολέα εξόδου και τρισδιάστατων γραφικών.

Βασισμένοι στις παραπάνω έννοιες και στο εργαλείο JaamSim περιγράφουμε την μοντελοποίηση και την προσομοίωση ενός ξενοδοχείου και μελετάμε τα αποτελέσματα.

# Ευχαριστίες

Η παρούσα πτυχιακή εργασία με θέμα "ΠΡΟΣΟΜΟΙΩΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΙΑΣ ΞΕΝΟΔΟΧΕΙΑΚΗΣ ΜΟΝΑΔΑΣ ΜΕ ΤΗ ΧΡΗΣΗ ΤΟΥ ΕΡΓΑΛΕΙΟΥ JAAMSIM" πραγματοποιήθηκε στο πλαίσιο της πτυχιακής εργασίας του ΤΜΗΜΑΤΟΣ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΥ ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ το έτος 2021.

Η παρούσα διπλωματική εργασία είναι το αποτέλεσμα μιας σειράς αλληλεπιδράσεων με διάφορα άτομα, καθένα από τα οποία έπαιξε ένα σημαντικό ρόλο στην εξέλιξή της. Αξίζει λοιπόν, να αφιερωθεί η παρούσα σελίδα για να ευχαριστήσουμε ειλικρινά τα άτομα αυτά για τη βοήθεια που μας προσέφεραν.

Στο σημείο αυτό αισθανόμαστε την ανάγκη να εκφράσουμε τις ειλικρινείς και θερμές ευχαριστίες μας σε όσους συνέβαλλαν στην ολοκλήρωση αυτής της προσπάθειας:

Και πρώτα απ’ όλα, στον επιβλέποντα καθηγητή της πτυχιακής μας εργασίας, κύριο Σωτήρη Χριστοδούλου για τη συνεχή καθοδήγηση, την αμέριστη υποστήριξη, τις ουσιώδεις συμβουλές, καθώς επίσης και την αδιάκοπη συμπαράσταση και ενθάρρυνση που μου παρείχε σε όλο αυτό το χρονικό διάστημα.

Επιπλέον θα θέλαμε να να ευχαριστήσουμε όλους εκείνους τους συμφοιτητές μας οι οποίοι μας ενθάρρυναν και μας προέτρεψαν στην ολοκλήρωση της ακαδημαϊκής μας πορείας.

Τέλος, το μεγαλύτερο «ευχαριστώ» το οφείλουμε στα αγαπημένα μας πρόσωπα, στους γονείς μας τα αδέρφια μας και τους οικείους μας, που αποδέχθηκαν όλες τις επιλογές μας και μας παρείχαν στήριξη όλο αυτό το διάστημα, χωρίς την οποία τίποτα από όσα έχουμε καταφέρει μέχρι σήμερα δε θα γίνονταν πραγματικότητα.

# Πίνακας Περιεχομένων

# 1. 1. Προσομοίωση διακριτών συμβάντων

## 1.1 Συστήματα και μοντέλα

Είναι λογικό να ξεκινήσουμε με το τη διευκρίνιση της έννοιας "σύστημα" και να παρουσιάσουμε τις θεμελιώδεις έννοιες που σχετίζονται με τη θεωρία του συστήματος, όπως αυτή εξελίχθηκε με τα χρόνια.

Όμως, στην καθημερινή ζωή μας σε έναν κόσμο που όλο περισσότερο εξαρτάται από τον υπολογιστή παρατηρούμε δύο πράγματα. Πρώτον, πολλά από τα μεγέθη που συναντάμε είναι "διακριτές" ποσότητες, που συνήθως περιλαμβάνουν την καταμέτρηση ακέραιων αριθμών (πόσα μέρη βρίσκονται σε ένα απόθεμα, πόσα αεροπλάνα βρίσκονται στον αεροδιάδρομο, πόσες τηλεφωνικές κλήσεις είναι ενεργές, κ.α.). Δεύτερον, τα συστήματα και τα μοντέλα οδηγούν πολλές από τις διεργασίες που χρησιμοποιούμε και εξαρτώνται από στιγμιαία "συμβάντα", όπως το πάτημα ενός κουμπιού, το πάτημα ενός πλήκτρου του πληκτρολογίου ή ένα φανάρι που γίνεται πράσινο. Στην πραγματικότητα, μεγάλο μέρος της τεχνολογίας, που έχουμε εφεύρει και στην οποία βασιζόμαστε, καθοδηγείται από συμβάντα: τα δίκτυα επικοινωνίας, οι εγκαταστάσεις παραγωγής ή η εκτέλεση ενός προγράμματος υπολογιστή είναι τυπικά παραδείγματα.

### 1.1.1 Η έννοια του συστήματος

Η έννοια **σύστημα** μπορεί να οριστεί ως εξής:

*Σύστημα θεωρείται ένα σύνολο συναφών μερών.*

Συνεπώς κάτι δεν είναι σύστημα, αλλά μπορεί να θεωρηθεί ως ένα σύστημα. Ένας άλλος ορισμός είναι του IEEE Standard Dictionary of Electrical and Electronic Terms:

> *Σύστημα είναι ένας συνδυασμός συνιστωσών που δρούνε μαζί για να εκτελέσουν μια λειτουργία που δεν είναι εφικτή με κανένα από τα μεμονωμένα μέρη.*

Υπάρχουν δύο εξέχοντα χαρακτηριστικά στο δεύτερο ορισμό. Πρώτον, ένα σύστημα αποτελείται από αλληλεπιδρώσες "συνιστώσες", και δεύτερον ένα σύστημα σχετίζεται με μια "λειτουργία" που προτίθεται να εκτελέσει. Αξίζει επίσης να επισημανθεί ότι ένα σύστημα δεν πρέπει πάντα να σχετίζεται με φυσικά αντικείμενα και φυσικούς νόμους. Για παράδειγμα, η θεωρία συστημάτων έχει παράσχει πολύ βολικά πλαίσια για την περιγραφή οικονομικών μηχανισμών ή μοντελοποίησης της ανθρώπινης συμπεριφοράς.

### 1.1.2 Η διαδικασία μοντελοποίησης εισόδου-εξόδου

Οι επιστήμονες και οι μηχανικοί ασχολούνται κυρίως με την ποσοτική ανάλυση των συστημάτων και την ανάπτυξη τεχνικών σχεδιασμού, ελέγχου και σαφούς μέτρησης των επιδόσεων ενός συστήματος βάσει σαφώς καθορισμένων κριτηρίων. Ως εκ τούτου, οι καθαρά ποιοτικοί ορισμοί που αναφέρονται παραπάνω είναι ανεπαρκείς. Αντ' αυτού, αναζητούμε ένα μοντέλο ενός πραγματικού συστήματος. Μπορούμε να θεωρήσουμε ένα **μοντέλο** ως ένα μηχανισμό που απλά αναπαράγει τη συμπεριφορά του ίδιου του συστήματος. Για να είμαστε πιο ακριβείς, πρέπει να αναπτύξουμε κάποια μαθηματικά μέσα για την περιγραφή αυτής της συμπεριφοράς.

Για να πραγματοποιήσουμε τη διαδικασία μοντελοποίησης, ξεκινάμε με τον ορισμό ενός συνόλου **μετρήσιμων μεταβλητών** που σχετίζονται με ένα δεδομένο σύστημα. Για παράδειγμα, θέσεις και ταχύτητες σωματιδίων ή τάσεις και ρεύματα σε ένα ηλεκτρικό κύκλωμα, που είναι όλα πραγματικοί αριθμοί. Με τη μέτρηση αυτών των μεταβλητών σε μια χρονική περίοδο $[t_0, t_f]$ μπορούμε στη συνέχεια να συλλέξουμε δεδομένα. Μετά επιλέγουμε ένα υποσύνολο αυτών των μεταβλητών και υποθέτουμε ότι έχουμε τη δυνατότητα να τις μεταβάλλουμε με την πάροδο του χρόνου. Αυτό ορίζει ένα σύνολο συναρτήσεων χρόνου που θα τις καλέσουμε **μεταβλητές εισόδου**

$$\{ u_1(t), \ldots, u_p(t) \} \qquad t_0 \le t \le t_f$$

Στη συνέχεια, επιλέγουμε ένα άλλο σύνολο μεταβλητών που υποθέτουμε ότι μπορούμε να μετρήσουμε άμεσα ενώ μεταβάλλονται οι $u_1(t), \ldots, u_p(t)$, και έτσι ορίζουμε ένα σύνολο **μεταβλητών εξόδου**

$$\{ y_1(t), \ldots, y_m(t) \} \qquad t_0 \le t \le t_f$$

Για να απλοποιήσουμε το συμβολισμό, αναπαριστάνουμε τις μεταβλητές εισόδου μέσω ενός διανύσματος στήλης $u(t)$ – βλέπε υποσημείωση 1 - και τις μεταβλητές εξόδου μέσω ενός άλλου διανύσματος στήλης $y(t)$. Για συντομία, αναφερόμαστε σ' αυτά ως η **είσοδος** και η **έξοδος** αντίστοιχα. Έτσι, θα γράψουμε

$$\mathbf{u}(t) = [u_1(t), \ldots, u_p(t)]^T$$

όπου $[\bullet]^T$ δηλώνει τη μεταφορά[1] ενός διανύσματος, και ομοίως

---

1 Έσω ένα διάνυσμα $V = (v_1, \ldots, v_m) \in R^{1 \times m}$. Η μεταφορά του διανύσματος V είναι ένας $V^T \in R^{m \times 1}$ πίνακας με μία μόνο στήλη, γνωστός ως **διάνυσμα στήλης**.

$$\mathbf{y}(t) = [y_1(t), \ldots, y_m(t)]^T$$

Για να ολοκληρώσουμε ένα μοντέλο, είναι λογικό να υποθέσουμε ότι υπάρχει κάποια μαθηματική σχέση μεταξύ εισόδου και εξόδου. Επομένως, υποθέτουμε ότι μπορούμε να ορίσουμε συναρτήσεις

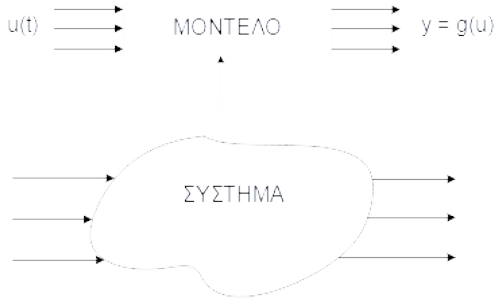
$$y_1(t) = g_1(u_1(t), \ldots, u_p(t)), \ldots, y_m(t) = g_m(u_1(t), \ldots, u_p(t))$$

και αποκτούμε το μοντέλο συστήματος στη μαθηματική μορφή

$$\mathbf{y} = \mathbf{g}(\mathbf{u}) = [g_1(u_1(t), \ldots, u_p(t)), \ldots, g_m(u_1(t), \ldots, u_p(t))]^T$$

όπου $\mathbf{g}(.)$ δηλώνει το διάνυσμα στήλης του οποίου οι συνιστώσες είναι οι συναρτήσεις $g_1(.), \ldots, g_m(.)$.

Αυτή είναι η απλούστερη δυνατή διαδικασία μοντελοποίησης και απεικονίζεται στο παρακάτω σχήμα.




**Σχήμα 1.1** Απλή διαδικασία μοντελοποίησης

Για να κυριολεκτήσουμε, ένα σύστημα είναι "κάτι πραγματικό" (π.χ. ένας ενισχυτής, ένα αυτοκίνητο, ένα εργοστάσιο, ένα ανθρώπινο σώμα), ενώ ένα μοντέλο είναι μια "αφαίρεση" του συστήματος (ένα σύνολο μαθηματικών εξισώσεων). Συχνά, το μοντέλο μόνο προσεγγίζει την πραγματική συμπεριφορά του συστήματος. Ωστόσο, όταν είμαστε πεπεισμένοι ότι έχουμε αποκτήσει ένα "καλό" μοντέλο, αυτή η διάκριση συνήθως απορρίπτεται και οι όροι σύστημα και μοντέλο χρησιμοποιούνται εναλλακτικά.

### 1.1.3 Η έννοια της κατάστασης

Σε γενικές γραμμές, **η κατάσταση** (state) ενός συστήματος θα πρέπει, σε κάποια χρονική στιγμή, να περιγράψει τη συμπεριφορά του με κάποιο μετρήσιμο τρόπο. Στη θεωρία

συστημάτων, ο όρος κατάσταση έχει πολύ πιο ακριβή σημασία και αποτελεί τον ακρογωνιαίο λίθο της διαδικασίας μοντελοποίησης και πολλών αναλυτικών τεχνικών.

Επιστρέφοντας στη διαδικασία μοντελοποίησης εισόδου-εξόδου που περιγράφεται στην προηγούμενη ενότητα, υποθέτουμε ότι το διάνυσμα εισόδου u(t) έχει καθοριστεί πλήρως για όλα τα $t \geq t0$. Επιπλέον, υποθέτουμε ότι η έξοδος y(t) παρατηρήθηκε σε κάποια χρονική στιγμή $t = t1 \geq t0$. Το ερώτημα που τίθεται είναι αν αυτές οι πληροφορίες είναι επαρκείς για να προβλέψουν μοναδικά όλη τη μελλοντική έξοδο y(t), $t > t1$. Σίγουρα, αν το μοντέλο μας πρόκειται να έχει κάποια χρήση, η ικανότητα πρόβλεψης του τρόπου συμπεριφοράς του συστήματος είναι κρίσιμη. Ακολουθεί ο ορισμός της κατάστασης:

> *Η κατάσταση ενός συστήματος κατά τη χρονική στιγμή t0 είναι οι πληροφορίες που απαιτούνται στην t0, έτσι ώστε η έξοδος y(t), για όλες τις τιμές $t \geq t0$, να καθορίζεται μοναδικά από αυτές τις πληροφορίες και από το u(t), $t \geq t0$.*

Όπως η είσοδος u(t) και η έξοδος y(t), η κατάσταση είναι επίσης γενικά ένα διάνυσμα, το οποίο δηλώνουμε με x(t). Οι συνιστώσες αυτού του διανύσματος, $x_1(t)$, ..., $x_n(t)$, ονομάζονται **μεταβλητές κατάστασης**.

### 1.1.4 Χώροι καταστάσεων

Μέχρι στιγμής, οι τιμές των μεταβλητών κατάστασης που έχουμε εξετάσει είναι πραγματικοί αριθμοί. Οι πραγματικές μεταβλητές είναι φυσικά πολύ βολικές όταν πρόκειται για την παραγωγή μοντέλων που βασίζονται σε διαφορικές εξισώσεις. Ωστόσο, δεν υπάρχει τίποτα ιερό σχετικά με τις μεταβλητές κατάστασης που λαμβάνουν πάντα πραγματικές αριθμητικές τιμές, σε αντίθεση με τις ακέραιες τιμές ή απλά τιμές από ένα δεδομένο διακριτό σύνολο, όπως το {ON, OFF}, {HIGH, MEDIUM, LOW}, ή {GREEN, RED, BLUE}. Στην πραγματικότητα, θα πρέπει πάντα να έχουμε κατά νου ότι η διαδικασία μοντελοποίησης επιτρέπει σημαντική ευελιξία στον καθορισμό της κατάστασης, της εισόδου και της εξόδου ενός συστήματος ανάλογα με την εφαρμογή ή το πρόβλημα που ενδιαφέρει.

Στα μοντέλα διακριτών καταστάσεων, ο χώρος καταστάσεων είναι ένα διακριτό σύνολο. Σε αυτήν την περίπτωση, ένα τυπικό δείγμα διαδρομής είναι μια κομματιαστή σταθερή συνάρτηση, καθώς οι μεταβλητές κατάστασης επιτρέπεται να μεταπηδούν, μόνο σε διακριτά χρονικά σημεία, από τη μία διακριτή τιμή κατάστασης στην άλλη. Φυσικά, υπάρχουν πολλές περιπτώσεις στις οποίες ένα υβριδικό μοντέλο μπορεί να είναι κατάλληλο, δηλαδή ορισμένες μεταβλητές κατάστασης είναι διακριτές και μερικές είναι συνεχείς.

## 1.2 Συστήματα διακριτών συμβάντων

Όταν ο χώρος καταστάσεων ενός συστήματος περιγράφεται από ένα διακριτό σύνολο όπως {0, 1, 2, . .} και οι μεταβάσεις κατάστασης παρατηρούνται μόνο σε διακριτά χρονικά σημεία, συνδέουμε αυτές τις μεταβάσεις κατάστασης με "συμβάντα" και μιλάμε για ένα "σύστημα διακριτών συμβάντων". Ξεκινάμε τη μελέτη αυτών των συστημάτων εντοπίζοντας πρώτα τα θεμελιώδη χαρακτηριστικά τους και εξετάζοντας μερικά γνωστά παραδείγματα συστημάτων διακριτών συμβάντων.

### 1.2.1 Η έννοια του συμβάντος

Όπως και με τον όρο "σύστημα", δεν θα προσπαθήσουμε να ορίσουμε αυστηρά τι είναι ένα "συμβάν". Θέλουμε μόνο να τονίσουμε ότι ένα **συμβάν** πρέπει να θεωρείται ότι λαμβάνει χώρα στιγμιαία και προκαλεί μεταβάσεις από τη μία κατάσταση σε άλλη. Παραδείγματα συμβάντων είναι: περίπατοι, χοροί, γάμοι, χαμόγελα, εκρήξεις, λόξυγγες, αφίξεις και αναχωρήσεις, γεννήσεις και θάνατοι, βροντές, αστραπές, κτλ.

Ένα συμβάν μπορεί να αναγνωριστεί από μια συγκεκριμένη πραγματοποιηθείσα ενέργεια (π.χ. κάποιος πατά ένα κουμπί). Μπορεί να θεωρηθεί ως ένα αυθόρμητο περιστατικό που υπαγορεύεται από τη φύση (π.χ., ένας υπολογιστής "πέφτει" για λόγο πολύ περίπλοκο για να κατανοηθεί). Μπορεί, όμως, να είναι το αποτέλεσμα πολλών συνθηκών που πληρούνται ξαφνικά (π.χ., η στάθμη υγρού σε δεξαμενή ξεπερνάει μια δεδομένη τιμή).

Όταν εξετάζουμε ένα σύστημα που επηρεάζεται από διαφορετικούς τύπους συμβάντων, θα υποθέτουμε ότι μπορούμε να ορίσουμε ένα σύνολο συμβάντων E του οποίου τα στοιχεία είναι όλα αυτά τα συμβάντα. Είναι σαφές ότι το E είναι ένα διακριτικό σύνολο.

### 1.2.2 Συστήματα καθοδηγούμενα από συμβάντα

Σε διάφορες χρονικές στιγμές (όχι απαραίτητα γνωστές εκ των προτέρων και όχι απαραίτητα συμπίπτουσες με τους χτύπους του ρολογιού), κάποιο συμβάν e "ανακοινώνει" ότι λαμβάνει χώρα. Κάθε συμβάν $e \in E$ ορίζει μια ξεχωριστή διαδικασία δια μέσω της οποίας καθορίζονται οι χρονικές στιγμές, όταν συμβαίνει το συμβάν e. Οι μεταβάσεις κατάστασης είναι το αποτέλεσμα του συνδυασμού αυτών των ασύγχρονων και ταυτόχρονων διαδικασιών συμβάντων. Επιπλέον, αυτές οι διαδικασίες δεν χρειάζεται να είναι ανεξάρτητες μεταξύ τους.

### 1.2.3 Συστήματα διακριτών συμβάντων

Τα συστήματα διακριτών συμβάντων ικανοποιούν τις ακόλουθες δύο ιδιότητες:

1. Ο χώρος καταστάσεων είναι ένα διακριτό σύνολο.

2. Ο μηχανισμός μεταβάσεων κατάστασης καθοδηγείται από συμβάντα.

Παρακάτω παρουσιάζουμε έναν άτυπο πρώτο ορισμό των διακριτών συστημάτων συμβάντων με βάση αυτές τις ιδιότητες.

> *Το **Διακριτό Σύστημα Συμβάντων** (DES - Discrete Event System) είναι ένα σύστημα διακριτών καταστάσεων, καθοδηγούμενο από συμβάντα, δηλαδή, η εξέλιξη της κατάστασής του εξαρτάται εξ ολοκλήρου από την εμφάνιση ασύγχρονων διακριτών συμβάντων με την πάροδο του χρόνου.*

Πολλά συστήματα, ιδιαίτερα τεχνολογικά, είναι πραγματικά συστήματα διακριτών καταστάσεων. Ακόμα κι αν αυτό δε συμβαίνει, για πολλές εφαρμογές που ενδιαφέρουν, μπορεί να είναι απαραίτητη μια άποψη (view) διακριτών καταστάσεων ενός σύνθετου συστήματος. Ακολουθούν μερικά απλά παραδείγματα συστημάτων διακριτών καταστάσεων:

- Η κατάσταση ενός υπολογιστή μπορεί να επιλεγεί από ένα σύνολο όπως {ON, OFF} ή {BUSY, IDLE, DOWN}.
- Ένας υπολογιστής που εκτελεί ένα πρόγραμμα μπορεί να θεωρηθεί ότι βρίσκεται σε μία από τις τρεις καταστάσεις: {WAITING FOR INPUT, RUNNING, DOWN}. Επιπλέον, η κατάσταση RUNNING μπορεί να αναλυθεί σε διάφορες μεμονωμένες καταστάσεις ανάλογα, για παράδειγμα, με τη γραμμή κώδικα που εκτελεί ανά πάσα στιγμή.
- Οποιοσδήποτε τύπος αποθέματος που αποτελείται από διακριτές οντότητες (π.χ. προϊόντα, νομισματικές μονάδες, άτομα) έχει φυσικό χώρο καταστάσεων στους μη αρνητικούς ακέραιους αριθμούς {0, 1, 2, . . .}.
- Τα περισσότερα παιχνίδια μπορούν να μοντελοποιηθούν ως έχοντα ένα διακριτό χώρο καταστάσεων. Στο σκάκι, για παράδειγμα, κάθε πιθανή διάταξη επί της σκακιέρας καθορίζει μια κατάσταση. Ο προκύπτων χώρος καταστάσεων είναι τεράστιος, αλλά είναι διακριτός.

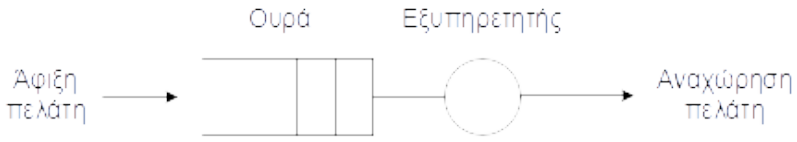
### 1.2.4 Παραδείγματα συστημάτων διακριτών συμβάντων

Οι επόμενες υποενότητες περιέχουν παραδείγματα DES που αντλούνται από τον πραγματικό κόσμο και την κοινή εμπειρία της τεχνολογίας. Προκειμένου να παρουσιάσουμε αυτά τα συστήματα αποτελεσματικά, ξεκινάμε με ένα απλό "δομικό στοιχείο" το οποίο θα χρησιμεύσει για την αναπαράσταση πολλών DES που ενδιαφέρουν.

**Συστήματα ουράς**

Συγκεκριμένοι πόροι μπορεί να μην είναι άμεσα διαθέσιμοι, οπότε οι ενδιαφερόμενοι πρέπει να περιμένουν. Για παράδειγμα, για να χρησιμοποιήσουν τον πόρο "ταμίας σε τράπεζα", οι άνθρωποι σχηματίζουν μια γραμμή και περιμένουν τη σειρά τους. Μερικές φορές, η αναμονή δεν γίνεται από ανθρώπους, αλλά από διακριτά αντικείμενα ή πιο αφηρημένες "οντότητες". Για παράδειγμα, για να χρησιμοποιηθεί ο πόρος "φορτηγό", προϊόντα περιμένουν σε μια αποθήκη. Ομοίως, για να χρησιμοποιηθεί ο πόρος "CPU", διάφορες "εργασίες" περιμένουν κάπου εντός ενός υπολογιστή έως ότου τους δοθεί πρόσβαση στην CPU μέσω δυνητικά περίπλοκων μηχανισμών.

Θα χρησιμοποιήσουμε ένα σύστημα ουράς ως βασική δομική ενότητα για πολλά από τα πιο περίπλοκα DES που θα εξετάσουμε. Στο σχήμα που ακολουθεί απεικονίζουμε γραφικά ένα απλό σύστημα ουράς:




**Σχήμα 1.2** Ένα απλό σύστημα ουράς

Ο κύκλος αναπαριστάνει έναν εξυπηρετητή και το ανοιχτό πλαίσιο αναπαριστάνει μια ουρά που προηγείται του εξυπηρετητή. Τα πλαίσια εντός της ουράς δείχνουν τις θέσεις των πελατών που περιμένουν. Οι πελάτες φτάνουν στην ουρά και αναχωρούν από τον εξυπηρετητή. Εννοείται ότι η διαδικασία εξυπηρέτησης πελατών απαιτεί συνήθως αυστηρά θετικό χρόνο (διαφορετικά δεν θα υπήρχε αναμονή). Οπότε ένας εξυπηρετητής μπορεί να θεωρηθεί ως ένα "μπλοκ καθυστέρησης" που κρατά έναν πελάτη για κάποιο χρονικό διάστημα εξυπηρέτησης.

Προκειμένου να προσδιοριστούν πλήρως τα χαρακτηριστικά ενός συστήματος ουράς, όπως αυτό του παραπάνω σχήματος, πρέπει επίσης να ορίσουμε (α) την χωρητικότητα της ουράς και (β) την πειθαρχία ουράς. Η **χωρητικότητα** μιας ουράς είναι ο μέγιστος αριθμός πελατών που μπορούν να φιλοξενηθούν στον πραγματικό χώρο αναμονής. Σε πολλά μοντέλα, θεωρείται ότι η χωρητικότητα είναι "άπειρη", με την έννοια ότι ο χώρος αναμονής είναι τόσο μεγάλος που η πιθανότητα πλήρωσης είναι αμελητέα. Η **πειθαρχία ουράς** αναφέρεται στον κανόνα σύμφωνα με τον οποίο επιλέγεται ο επόμενος πελάτης, που θα εξυπηρετηθεί, από την

ουρά. Τέτοιος απλός κανόνας είναι ο First-In-First-Out (FIFO) βάσει του οποίου οι πελάτες εξυπηρετούνται αυστηρά με τη σειρά με την οποία φτάνουν.

**Συστήματα παραγωγής**

Οι πελάτες μιας διαδικασίας παραγωγής είναι **εξαρτήματα** ή **κομμάτια της παραγωγής**. Τα εξαρτήματα ανταγωνίζονται για την πρόσβαση σε διάφορους εξυπηρετητές, οι οποίοι σε ένα τυπικό εργοστάσιο είναι **μηχανήματα**, που εκτελούν συγκεκριμένες λειτουργίες, και **συσκευές χειρισμού υλικών**, όπως ρομπότ και ιμάντες μεταφοράς. Όταν δε γίνεται επεξεργασία των εξαρτημάτων, αυτά αποθηκεύονται σε πρόσκαιρη μνήμη έως ότου ο εξυπηρετητής, που απαιτείται για την επόμενη λειτουργία, γίνει διαθέσιμος. Λόγω των πραγματικών φυσικών περιορισμών, οι πρόσκαιρες μνήμες σε ένα σύστημα παραγωγής έχουν συνήθως πεπερασμένη χωρητικότητα.

Για άλλη μια φορά, τα μοντέλα ουράς παρέχουν ένα βολικό πλαίσιο για την περιγραφή συστημάτων παραγωγής. Ένα απλό παράδειγμα παρουσιάζεται στο ακόλουθο σχήμα:


**Σχήμα 1.3** Μοντέλο ουράς ενός συστήματος παραγωγής

Τα εξαρτήματα περνούν από δύο μηχανήματα, όπου η χωρητικότητα της προσωρινής μνήμης πριν από το δεύτερο μηχάνημα περιορίζεται στις δύο θέσεις. Ως αποτέλεσμα, είναι πιθανό η εργασία ολοκλήρωσης εξαρτημάτων στο μηχάνημα 1 να διαπιστώσει ότι το μηχάνημα 2 είναι απασχολημένο και ότι είναι κατειλημμένες και οι δύο θέσεις αποθήκευσης της ουράς. Σε αυτήν την περίπτωση, το εξάρτημα πρέπει να παραμείνει στο μηχάνημα 1, παρόλο που δε χρειάζεται πλέον σέρβις εκεί. Επιπλέον, άλλα εξαρτήματα που περιμένουν να έχουν πρόσβαση στο μηχάνημα 1 αναγκάζονται να παραμείνουν στην ουρά. Αυτή η κατάσταση αναφέρεται ως **αποκλεισμός**.

Το σύνολο συμβάντων που έχει οριστεί για αυτό το παράδειγμα είναι

$$E = \{a, c_1, d_2\}$$

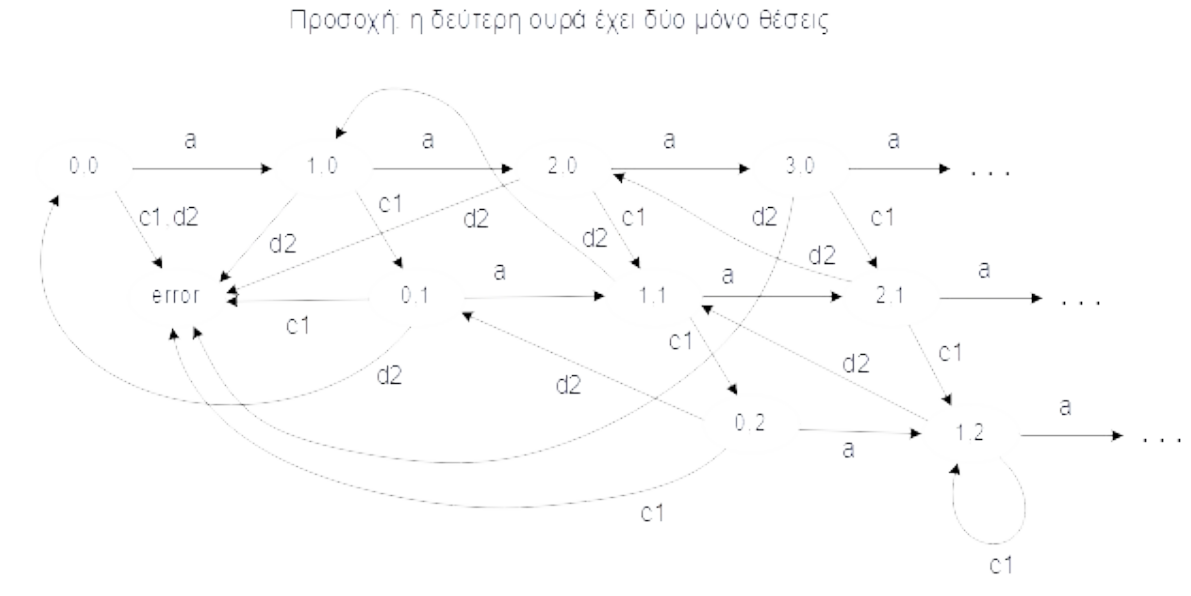
όπου

$a$ είναι μια άφιξη από τον εξωτερικό κόσμο στο πρώτο μηχάνημα

$c_1$ είναι η ολοκλήρωση της εξυπηρέτησης στο πρώτο μηχάνημα

$d_2$ είναι μια αναχώρηση από το δεύτερο μηχάνημα

Το αυτόματο που ακολουθεί περιγράφει τη λειτουργία του συστήματος παραγωγής του παραπάνω σχήματος. Κάθε κατάσταση είναι ένα διατεταγμένο ζεύγος (n,m), όπου n είναι ο αριθμός των εξαρτημάτων που περιμένουν στην πρώτη ουρά και m είναι ο αριθμός των εξαρτημάτων που περιμένουν στη δεύτερη ουρά και ισχύει $m < 3$.




**Σχήμα 1.4 Α**υτόματο λειτουργίας του συστήματος παραγωγής

Σημειώνουμε ότι ένα συμβάν $c_1$ δε συνεπάγεται μετακίνηση ενός εξαρτήματος από το μηχάνημα 1 στην επόμενη προσωρινή μνήμη, καθώς ο αποκλεισμός είναι πιθανός. Όταν $m = 2$, το μηχάνημα 1 μπορεί είτε να μην εξυπηρετεί εξάρτημα είτε να εξυπηρετεί έναν αριθμό εξαρτημάτων.

$$X = \{(x1, x2) : x1 \geq 0, x2 \in \{0, 1, 2\}\}$$

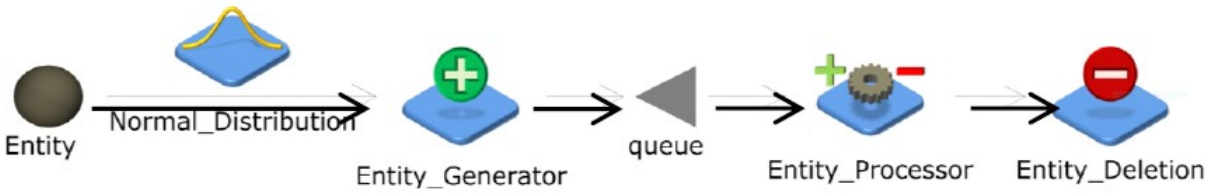
Βέβαια όσο περισσότερα είναι τα συμβάντα, τόσο πιο περίπλοκο γίνεται το αυτόματο.

## 1.3 Προσομοίωση συστημάτων διακριτών συμβάντων

Ο όρος **προσομοίωση συστήματος διακριτών συμβάντων** (DESS - Discrete Event System Simulation) έχει καθιερωθεί ως ένας όρος που περιλαμβάνει διάφορες προσεγγίσεις προσομοίωσης με υπολογιστή, όλες βασισμένες στη γενική ιδέα της μοντελοποίησης οντοτήτων/αντικειμένων και γεγονότων. Στη βιβλιογραφία για την DESS, αναφέρεται συχνά ότι η DESS βασίζεται στην έννοια των "οντοτήτων που ρέουν μέσω του συστήματος" [πιο συγκεκριμένα, μέσω ενός **δικτύου αναμονής** (queueing network)]. Αυτό είναι το υπόδειγμα μιας ολόκληρης κατηγορίας λογισμικού προσομοίωσης στην παράδοση του GPSS (Gordon,

1961) και SIMAN/Arena (Pegden & Davis, 1992). Ωστόσο αυτό το υπόδειγμα χαρακτηρίζει μόνο μια ειδική (αλλά σημαντική) κατηγορία DESS, δεν ισχύει για όλα τα διακριτά δυναμικά συστήματα.

Η DESS είναι μια παραδοσιακή μέθοδος προσομοίωσης βάσει συμβάντων, που μπορεί να περιγράψει τη συμπεριφορά διαδικασιών και υπηρεσιών (Kreutzer, 1979). Έχει τη δυνατότητα να συμπεριλάβει τη δυναμική των διακριτών συστημάτων χαμηλού έως μεσαίου επιπέδου αφαίρεσης. Το μοντέλο αποτελείται βασικά από οντότητες, χαρακτηριστικά, ουρές και πόρους που υποστηρίζουν διαφορετικές δραστηριότητες. Αυτές οι δραστηριότητες μπορεί να είναι, π.χ. δημιουργία, διαγραφή, καθυστέρηση και ένωση οντοτήτων, ή μαθηματικοί υπολογισμοί, που συμβαίνουν σε διακριτά χρονικά σημεία. Επιπλέον, τέτοιες προσομοιώσεις επιτρέπουν την εφαρμογή κατανομών πιθανότητας για τον έλεγχο της δειγματοληψίας ή άλλων παραμέτρων, Το παρακάτω σχήμα απεικονίζει ένα απλοϊκό μοντέλο προσομοίωσης διακριτών συμβάντων, που σχεδιάστηκε με το λογισμικό ανοιχτού κώδικα JaamSim (Harrison et al., 2016). Αυτό το μοντέλο δημιουργεί οντότητες με κανονικά κατανεμημένο διάστημα, εισάγει σε ουρά και καταλαμβάνει έναν πόρο για την επεξεργασία της οντότητας για ένα χρονικό διάστημα, απελευθερώνει τον πόρο και, τέλος, διαγράφει την οντότητα.




**Σχήμα 1.5** Βασικό μοντέλο προσομοίωσης διακριτών συμβάντων στο JaamSim

Παρά το αυτονόητο, αυτές οι προσομοιώσεις επιτρέπουν μια λεπτομερή ανάλυση του κόστους, της δυναμικότητας, της ποιότητας και της διαχείρισης του χρόνου. Επομένως τα μοντέλα διακριτών συμβάντων μπορούν να κατασκευαστούν ώστε να καλύπτουν ένα μεγάλο εύρος εφαρμογών, όπως συστήματα ουρών, υπολογιστών, επικοινωνιών, παραγωγής και κυκλοφορίας. Ένα πρόσφατο παράδειγμα που αποδεικνύει αυτήν την ευελιξία είναι η εφαρμογή διακριτών συμβάντων προσομοίωσης και βελτιστοποίησης μιας νοσοκομειακής υπηρεσίας (Dosi et al., 2019). Επιπλέον τα τελευταία χρόνια παρατηρείται αυξανόμενο ενδιαφέρον για την προσομοίωση **συστημάτων προϊόντος-υπηρεσίας** (Product-Service Systems - PSS) (Weidemann et al., 2015). Όπως επισημάνθηκε στο άρθρο, η χρήση διακριτών συμβάντων για PSS έχει υψηλό δυναμικό εφαρμογής, καθιστώντας δυνατούς τρεις τύπους ανάλυσης:

- ***Προσανατολισμένος στη διεργασία*** - χρησιμοποιείται για τη μοντελοποίηση ολόκληρης διαδικασίας, συμπεριλαμβανομένης της πλευράς του πελάτη και του παρόχου.
- ***Προσανατολισμένος στη χρήση*** - χρησιμοποιείται για την προσομοίωση της χρήσης του προϊόντος από τον πελάτη, προκειμένου να βρεθεί ο σωστός αριθμός πόρων για την παροχή ποιοτικής υπηρεσίας, δεδομένης συγκεκριμένης αβεβαιότητας.
- ***Προσανατολισμένος στο προϊόν*** - χρησιμοποιείται για την πρόβλεψη της συμπεριφοράς του προϊόντος κατά τη χρήση και, επομένως, προσαρμόζει τις προδιαγραφές του.

Εκτός από τις δυνατότητες που σχολιάστηκαν παραπάνω, η προσομοίωση μπορεί επίσης να περιέχει αλγόριθμους λήψης αποφάσεων, οι οποίοι είναι ένα κρίσιμο μέρος της ολιστικής μοντελοποίησης των PSS. Αυτό μπορεί να πραγματοποιηθεί μέσω της σύνδεσης δεδομένων που παρέχονται από τη χρήση του προϊόντος με τη διαχείριση της υπηρεσίας που παρέχουν αυτά τα προϊόντα.

Ως μειονέκτημα, τα μοντέλα διακριτών συμβάντων εξαρτώνται από την ανθρώπινη συμπεριφορά, η οποία πρέπει να προσεγγιστεί μέσω μοντέλων κατανομής και μπορεί να προκαλέσει ένα βαθμό αβεβαιότητας. Η μοντελοποίηση τέτοιων παραμέτρων απαιτεί μια λεπτομερή έρευνα πεδίου που σε πολλές περιπτώσεις μπορεί να μην είναι πρακτική, αλλά μπορεί να προσφέρει καλές προσεγγίσεις εάν υπάρχει κυκλική ή εποχική συμπεριφορά στα δεδομένα.

Η ανάπτυξη του λογισμικού προσομοίωσης συστημάτων διακριτών συμβάντων εξελίχθηκε σταδιακά από τη δεκαετία του 1960 και πολλά συστήματα έχουν αναπτυχθεί από τη βιομηχανία και τον ακαδημαϊκό χώρο για την αντιμετώπιση διαφόρων βιομηχανικών προβλημάτων. Εν συντομία, έχουν εξελιχθεί τέσσερες γενιές προϊόντων λογισμικού προσομοίωσης, όπως:

**1η γενιά (τέλη της δεκαετίας του 1960)** – Ο προγραμματισμός γίνεται σε γλώσσες υψηλού επιπέδου όπως η FORTRAN. Ο μοντελοποιητής ήταν υποχρεωμένος να προγραμματίσει τόσο τη λογική του μοντέλου όσο και τον κώδικα για τον έλεγχο των γεγονότων και των δραστηριοτήτων στο μοντέλο.

**2η γενιά (τέλη της δεκαετίας του 1970)** – Χρησιμοποιούνται γλώσσες προσομοίωσης που έχουν εντολές όπως "μηχανή" ελέγχου συμβάντων, παραγωγή στατιστικής κατανομής, αναφορά, κ.λπ. Ένα μοντέλο στη γλώσσα προσομοίωσης μεταγλωττίζονταν και στη συνέχεια

συνδέονταν με τις παρεχόμενες υπορουτίνες για να παραγάγει ένα εκτελέσιμο μοντέλο. Παραδείγματα είναι τα GPSS (IBM), See Why (AT&T), AutoMod (ASI).

**3η γενιά (αρχές της δεκαετίας του 1980)** – Χρησιμοποιούνται γεννήτριες γλωσσών προσομοίωσης που είναι πακέτα εμπρόσθιου-άκρου τα οποία δημιουργούν κώδικα σε γλώσσα προσομοίωσης. Ο παραγόμενος κώδικας μεταγλωττίζεται και στη συνέχεια συνδέεται με βιβλιοθήκες για την παραγωγή ενός εκτελέσιμου μοντέλου. Ο χρόνος ανάπτυξης του μοντέλου μειώθηκε, αλλά πάλι απαιτούνταν, από τον μοντελοποιητή, ο έλεγχος όλων των πτυχών του μηχανισμού προσομοίωσης. Παραδείγματα είναι το SIMAN (από την εταιρεία Systems Modeling), EXPRESS (από την AT&T).

**4η γενιά (τέλη της δεκαετίας του 1980)** - Διαδραστικά πακέτα προσομοίωσης καθιστούν δυνατό το "what you see is what you get", επιτρέπουν την τροποποίηση των μοντέλων ανά πάσα στιγμή και επιταχύνουν την "what-if" ανάλυση. Τα μοντέλα προσομοίωσης μπορούν να δημιουργηθούν πολύ γρήγορα από διευθυντές βιομηχανιών και μηχανικούς, ενθαρρύνοντας έτσι τα άτομα με γνώση και εμπειρία του προβλήματος, από πρώτο χέρι, να φτιάξουν τα ίδια το μοντέλο. Παραδείγματα είναι το WITNESS (AT&T) και το ARENA (Systems Modeling).

Στα μέσα της δεκαετίας του 1990, η τεχνολογία εικονικής πραγματικότητας δημιούργησε νέο ενθουσιασμό στην κοινότητα προσομοίωσης. Καταβλήθηκε σημαντική προσπάθεια για την ανάπτυξη ενός ολοκληρωμένου περιβάλλοντος προσομοίωσης, μέσω του οποίου οι μηχανικοί να μπορούν να προσομοιώνουν το σχεδιασμό και την παραγωγή προϊόντων χωρίς να ψάχνουν τα διαφορετικά πακέτα προσομοίωσης. Οι δύο κορυφαίοι προμηθευτές λογισμικού προσομοίωσης εκείνη την εποχή, η Lanner Group και η Deneb Inc., ανακοίνωσαν ένα σχέδιο για την από κοινού ανάπτυξη μιας νέας γενιάς λογισμικού προσομοίωσης για την υποστήριξη τόσο της διεργασίας όσο και της λεπτομερούς προσομοίωσης με περισσότερες δυνατότητες μοντελοποίησης και γραφικών. Ωστόσο ο ενθουσιασμός επισκιάστηκε σύντομα από πρωτοφανείς αλλαγές στις βιομηχανίες παραγωγής που ήταν αποτέλεσμα της παγκοσμιοποίησης. Οι προμηθευτές λογισμικού προσομοίωσης πέρασαν και αυτοί από τη βιομηχανική ενοποίηση. Οι AutoSimulation, System Modeling, Simple++, Deneb αποτελούν πλέον τμήματα μεγάλων εταιρειών. Υπάρχουν νέες γενιές προμηθευτών με διαφορετικά επιχειρηματικά μοντέλα που χρησιμοποιούν το διαδίκτυο για διαδικτυακές πωλήσεις και υποστήριξη προϊόντων, όπως η Simul8 Inc.

Γενικά, δεν υπάρχει σημαντική ανάπτυξη στις τεχνολογίες και το λογισμικό προσομοίωσης συστημάτων διακριτών συμβάντων, από την περίοδο της 4ης γενιάς. Από την άλλη πλευρά, οι

τεράστιες αλλαγές στο επιχειρηματικό περιβάλλον παρουσίασαν νέες προκλήσεις και ευκαιρίες στην προσομοίωση συστημάτων διακριτών συμβάντων.

## 1.4 Εφαρμογές DESS

### 1.4.1 DESS στον τομέα παραγωγής

Η DESS χρησιμοποιείται παραδοσιακά για βιομηχανικές εφαρμογές. Στις δεκαετίες του 1980 και του 1990, υπήρξε μια ραγδαία ανάπτυξη της τεχνολογίας παραγωγής στις βιομηχανικές χώρες: CAD (σχεδίαση με υποβοήθηση υπολογιστή), CAM (κατασκευή με υποβοήθηση υπολογιστή), AGV (αυτόματα οδηγούμενο όχημα), Ρομποτική, FMS (ευέλικτο σύστημα παραγωγής) και CIM (ολοκληρωμένη παραγωγή με υπολογιστή). Το ίδιο μπορεί να ειπωθεί για τις τεχνολογίες DESS. Πολλές εταιρείες είχαν επενδύσει σε μεγάλο βαθμό στις νέες τεχνολογίες προκειμένου να καταστήσουν ευέλικτες τις εργασίες παραγωγής. Το λογισμικό DESS ήταν το εργαλείο που βοήθησε τους μάνατζερ στη λήψη σωστών αποφάσεων. Κάθε μάνατζερ παραγωγής ήθελε να βελτιώσει την παραγωγικότητα, αυξάνοντας την απόδοση, συντομεύοντας το χρόνο παράδοσης και αυξάνοντας τη χρήση πόρων. Μέσω της προσομοίωσης, μπορούσαν να αξιολογήσουν τη συμπεριφορά μιας διαδικασίας παραγωγής υπό διαφορετικές συνθήκες, να πραγματοποιήσουν ανάλυση σεναρίου "what-if" προκειμένου να εντοπίσουν καλύτερη φυσική διαμόρφωση και καλύτερες επιχειρησιακές πολιτικές. Συνολικά το λογισμικό DESS χρησιμοποιήθηκε στους ακόλουθους τομείς:

1) Στο σχεδιασμό και αξιολόγηση νέων διαδικασιών παραγωγής.
2) Στη βελτίωση της επίδοσης των υφιστάμενων διαδικασιών παραγωγής, για παράδειγμα μελέτη σκοπιμότητας ενός αυτοματοποιημένου συστήματος διαχείρισης υλικών.
3) Στη θέσπιση βέλτιστης επιχειρησιακής πολιτικής.
4) Σε αλγόριθμο που υποστηρίζει το σχεδιασμό παραγωγής και το χρονοπρογραμματισμό.

Μια έρευνα, που χρηματοδοτήθηκε από το Υπουργείο Εμπορίου και Βιομηχανίας του Ηνωμένου Βασιλείου, έδειξε ότι το μοντέλο προσομοίωσης χρησιμοποιείται σε όλα τα επίπεδα διαχείρισης των 500 μεγαλύτερων εταιρειών των Ηνωμένων Πολιτειών. Διαπιστώθηκε επίσης ότι όπου χρησιμοποιήθηκε προσομοίωση, εξοικονομήθηκε κόστος κεφαλαίου μεταξύ του 5% και 10%. Ο παραγωγικός τομέας ήταν η κύρια αγορά για λογισμικό DESS.

Στα τέλη της δεκαετίας του 1990, ο τομέας παραγωγής άρχισε να αλλάζει ραγδαία, με την Κίνα να εμφανίζεται ως "παγκόσμια βάση παραγωγής". Πολλές εταιρείες είτε ανέθεσαν την παραγωγή τους σε τρίτους είτε μετεγκατέστησαν τις γραμμές παραγωγής τους σε χώρες με χαμηλά μεροκάματα. Όταν μια εταιρεία περνάει από μετασχηματισμό, οι εφαρμογές DESS βρίσκονται πάντα σε αδιέξοδο. Η μεγάλη κλίμακα του "βιομηχανικού μετασχηματισμού" δημιούργησε νέα προβλήματα στους μάνατζερ και νέες προκλήσεις στις τεχνολογίες DESS.

Υπάρχουν πολλές μελέτες σχετικά με την μοντελοποίηση της ανθρώπινης απόδοσης. Ωστόσο, το λογισμικό DESS δεν έλαβε υπόψη τα ευρήματα και παραμένει προβληματικό στη μοντελοποίηση της ανθρώπινης απόδοσης. Προς το παρόν, το εμπορικό λογισμικό DESS δεν είναι σε θέση να χειριστεί αυτά τα ζητήματα τόσο αποτελεσματικά όσο και αποδοτικά. Πρέπει να γίνει πολύ περισσότερη δουλειά για να καταστεί το λογισμικό DESS ικανό να μοντελοποιήσει όλες τις δραστηριότητες παραγωγής.

### 1.4.2 DESS στον τομέα των υπηρεσιών

Ενώ ο τομέας παραγωγής βρίσκεται σε παρακμή, ο τομέας των υπηρεσιών στις βιομηχανικές χώρες επεκτείνεται γρήγορα. Το μέγεθος λειτουργίας έχει αυξηθεί σημαντικά και η φύση της έχει γίνει πολύ περίπλοκη. Οι μάνατζερ προσπάθησαν να εξισορροπήσουν την άριστη εξυπηρέτηση πελατών με τη λειτουργική αποδοτικότητα (που σημαίνει μικρότερος χρόνος επεξεργασίας, λιγότερος χρόνος αναμονής για τους πελάτες και μεγαλύτερη χρήση πόρων). Πολλοί από αυτούς βρήκαν ότι η DESS μπορεί να τους βοηθήσει να λάβουν τη σωστή απόφαση. Με τον ίδιο τρόπο, όπως με τις εφαρμογές παραγωγής, χρησιμοποιούν το λογισμικό DESS για να μοντελοποιήσουν τις επιχειρηματικές τους διεργασίες και να αξιολογήσουν τη συμπεριφορά του συστήματος υπηρεσιών υπό διαφορετικές συνθήκες· να πραγματοποιήσουν ανάλυση σεναρίων "what-if" προκειμένου να εντοπίσουν καλύτερο τρόπο παροχής των υπηρεσιών τους. Μερικά παραδείγματα είναι:

- Οι Τραπεζικές και χρηματοοικονομικές υπηρεσίες.
- Η υγειονομική περίθαλψη και τα νοσοκομεία.
- Η αποθήκευση και μεταφορά.
- Ο δημόσιος τομέας.

Η μοντελοποίηση των λειτουργιών εξυπηρέτησης είναι διαφορετική από εκείνη των εργασιών παραγωγής, όπως περιγράφεται παρακάτω:

1) **Ροή διεργασίας**: Μια διαδικασία παραγωγής συνδέεται πάντα με φυσικές ροές υλικών/συνιστωσών και επομένως μπορούν εύκολα να αναγνωριστούν. Αυτό μπορεί να μην ισχύει για πολλές εφαρμογές υπηρεσιών, όπου οι επιχειρηματικές δραστηριότητες βασίζονται σε πληροφορίες και ενεργοποιούνται από ένα εξωτερικό ή εσωτερικό συμβάν, όπως ένα γραπτό ή προφορικό αίτημα. Η τρέχουσα λύση είναι να χρησιμοποιηθεί ένα εργαλείο χαρτογράφησης της επιχειρηματικής διεργασίας για να την καταγράψει και, στη συνέχεια, να μετατρέψει το μοντέλο διεργασίας σε μοντέλο προσομοίωσης διακριτών συμβάντων.

2) **Δεδομένα σχετικά με τη διεργασία, όπως ο χρόνος επεξεργασίας**: Σε μια εταιρεία παραγωγής, οι βιομηχανικοί μηχανικοί είναι υπεύθυνοι για τη μελέτη του χρόνου, τον καθορισμό του χρόνου επεξεργασίας και την εξισορρόπηση της ροής. Οι περισσότερες εταιρείες παροχής υπηρεσιών δεν προσλαμβάνουν βιομηχανικούς μηχανικούς ή δεν έχουν ισοδύναμη θέση σε οργανισμούς. Ως αποτέλεσμα, πολλά από τα δεδομένα που σχετίζονται με τη διεργασία δεν είναι άμεσα διαθέσιμα.

3) **Εργαζόμενοι της γνώσης**: Σε πολλές εταιρείες παροχής υπηρεσιών, οι εργαζόμενοι εργάζονται κυρίως με πληροφορίες ή αναπτύσσουν και χρησιμοποιούν γνώσεις. Είναι εργαζόμενοι της γνώσης (knowledge workers), ένας όρος που επινοήθηκε από τον Peter Drucker. Ένας εργαζόμενος της γνώσης τείνει να έχει προσωπικά κίνητρα, να εργάζεται διαδραστικά και να λαμβάνει συνεχώς αποφάσεις. Ο τρόπος αναπαράστασης των εργαζομένων της γνώσης και της διαδικασίας λήψης αποφάσεων στην προσομοίωση διακριτών συμβάντων παραμένει αντικείμενο μελέτης.

Στη μεταβιομηχανική οικονομία, ο τομέας των υπηρεσιών αποτελεί περισσότερο από το ήμισυ της Αμερικανικής οικονομίας. Από τα μέσα της δεκαετίας του 1990, ο τομέας έχει προκαλέσει σχεδόν το σύνολο των αυξήσεων της απασχόληση στην οικονομία των ΗΠΑ. Εκτιμάται ότι οι εργαζόμενοι της γνώσης ξεπερνούν όλους τους άλλους εργαζόμενους στη Βόρεια Αμερική τουλάχιστον με ένα περιθώριο τεσσάρα προς ένα. Έτσι, υπάρχουν μεγάλες προοπτικές για τις τεχνολογίες προσομοίωσης διακριτών συμβάντων στον τομέα των υπηρεσιών. Ωστόσο, απαιτείται νέα προσέγγιση και νέες τεχνικές για τη μοντελοποίηση και την προσομοίωση των εργαζομένων της γνώσης και της διαδικασίας λήψης αποφάσεών από αυτούς.

### 1.4.3 Νέες ευκαιρίες για DESS

Το μεταβαλλόμενο επιχειρηματικό περιβάλλον και οι τεχνολογικές εξελίξεις δημιούργησαν άλλες ευκαιρίες για τις τεχνολογίες προσομοίωσης διακριτών συμβάντων. Συγκεκριμένα, θα επισημάνουμε τους ακόλουθους δύο τομείς:

1) **Συστήματα Επιχειρηματικής Νοημοσύνης** (Business Intelligence systems): Καθ' όλη τη δεκαετία του 1990 τα συστήματα ERP[2] (Enterprise Resource Planning) είχαν βρεθεί στο επίκεντρο της ηλεκτρονικής επιχείρησης. Οι εταιρείες ξόδευαν πολλούς πόρους και κατέβαλλαν μεγάλη προσπάθεια στην υλοποίηση συστημάτων ERP. Τώρα πολλές εταιρείες διαθέτουν μια ισχυρή IT υποδομή με υψηλό βαθμό ενσωμάτωσης πληροφοριών. Από την άποψη της προσομοίωσης διακριτών συμβάντων, τώρα είναι πολύ πιο εύκολο να ληφθούν τα δεδομένα για να δοθεί ένα μοντέλο προσομοίωσης από ό,τι στο παρελθόν, καθώς τα δεδομένα είναι άμεσα διαθέσιμα από το σύστημα ERP. Η εστίαση της διοίκησης έχει μετατοπιστεί από τη λήψη πληροφοριών στην ευφυή χρήση πληροφοριών για τη βελτίωση της απόδοσης της επιχείρησης. Το λογισμικό Επιχειρηματικής Νοημοσύνης βοηθάει τις εταιρείες να έχουν μια πιο ολοκληρωμένη γνώση των παραγόντων που επηρεάζουν την επιχείρησή τους, όπως οι μετρήσεις των πωλήσεων, η παραγωγή και οι εσωτερικές λειτουργίες. Ωστόσο, για να παρθούν καλύτερες επιχειρηματικές αποφάσεις, πρέπει να εξεταστεί ο τρόπος ανάπτυξης πόρων στις ευκαιρίες που εντοπίζονται. Η προσομοίωση διακριτών συμβάντων είναι μια ιδανική πλατφόρμα για την υποστήριξη των μάνατζερ στη λήψη αποφάσεων σχετικά με την ανάπτυξη πόρων. Επομένως, ένα πλήρες σύστημα Επιχειρηματικής Νοημοσύνης πρέπει να περιλαμβάνει τόσο την δυνατότητα ανάλυσης δεδομένων όσο και μια τεχνολογία πρόβλεψης. Η ικανότητα ανάλυσης δεδομένων συλλέγει και αναλύει μεγάλες ποσότητες μη δομημένων δεδομένων, όπως μετρήσεις παραγωγής, στατιστικές πωλήσεων, αναφορές συμμετοχής και αριθμοί φθοράς των πελατών με έμφαση στην πλήρη γνώση των παραγόντων που επηρεάζουν τις επιχειρήσεις. Η τεχνολογία πρόβλεψης επιτρέπει στους μάνατζερ να αξιολογούν διαφορετικές επιλογές προκειμένου να λαμβάνουν σωστές επιχειρηματικές αποφάσεις.

2) **Εκπαίδευση με βάση την προσομοίωση**: Όταν μια εταιρεία αποφασίσει να αναθέσει την παραγωγή σε τρίτους, μια σημαντική επίπτωση είναι ο τρόπος με τον οποίο θα διατηρηθεί μακροπρόθεσμα η τεχνογνωσία της, υπό την προϋπόθεση ότι η εταιρεία εξακολουθεί να θέλει να σχεδιάζει και να αναπτύσσει τα δικά της προϊόντα. Η διάβρωση της τεχνογνωσίας είναι

2 Ο **εταιρικός προγραμματισμός πόρων** ορίζεται ως η ικανότητα παράδοσης μιας ολοκληρωμένης σειράς επιχειρηματικών εφαρμογών.

πρόκληση όχι μόνο για τις εταιρείες αλλά και για τα εκπαιδευτικά ιδρύματα στις βιομηχανικές χώρες. Μια πιθανή λύση είναι η δημιουργία της προσομοίωσης ενός περιβάλλον παραγωγής για στελέχη, διευθυντές, μηχανικούς και φοιτητές ώστε να βιώσουν και να μάθουν πώς να διαχειρίζονται τις λειτουργίες παραγωγής και εφοδιασμού. Η προσομοίωση διακριτών συμβάντων είναι μια ιδανική πλατφόρμα για μια τέτοια εφαρμογή. Επιπλέον, υπάρχει πληθώρα περιπτώσεων προσομοίωσης και μοντέλων που μπορούν να υιοθετηθούν για την εκπαίδευση μηχανικών και την επιχειρηματική εκπαίδευση. Λαμβάνοντας υπόψη τις τρέχουσες εξελίξεις στις τεχνολογίες διαδικτύου και τηλεπικοινωνιών, η διαδικασία παρασκευής και εφοδιασμού θα γίνει πλήρως αυτοματοποιημένη.

## 1.5 Σχέδιο προσομοίωσης προσανατολισμένης στη διεργασία

Μια μεγάλη κατηγορία DESS αποτελείται από περιβάλλοντα, όπου οι πόροι πρέπει να μοιράζονται μεταξύ πολλών χρηστών. Σε τέτοια περιβάλλοντα, είναι συχνά βολικό να σκεφτόμαστε για τις "οντότητες" (π.χ. πελάτες) ότι υποβάλλονται σε μια διεργασία καθώς ρέουν μέσω του DESS. Αυτή η διεργασία είναι μια ακολουθία συμβάντων που διαχωρίζονται από χρονικά διαστήματα. Κατά τη διάρκεια ενός τέτοιου χρονικού διαστήματος, μια οντότητα είτε εξυπηρετείται σε κάποιο πόρο είτε περιμένει να εξυπηρετηθεί. Στο σχέδιο προσομοίωσης προσανατολισμένης στη διαδικασία, η συμπεριφορά της DESS περιγράφεται μέσω πολλών τέτοιων διεργασιών, μία για κάθε τύπο οντότητας που ενδιαφέρει.

Για παράδειγμα, το απλό σύστημα ουράς ενός εξυπηρετητή της προηγούμενης ενότητας μπορεί να ειδωθεί ως εξής. Υπάρχει ένας μοναδικός τύπος οντότητας σε αυτό το σύστημα (ο πελάτης) και ένας μοναδικός πόρος (ο εξυπηρετητής). Κάθε οντότητα σε αυτό το σύστημα υποβάλλεται στην ακόλουθη διεργασία:

1. Αφικνείται.
2. Εισέρχεται στην ουρά.
3. Ζητάει εξυπηρέτηση από τον εξυπηρετητή. Αν ο εξυπηρετητής είναι αδρανής, η οντότητα "καταλαμβάνει" αυτόν τον πόρο, διαφορετικά παραμένει στην ουρά έως ότου ο εξυπηρετητής γίνει αδρανής.
4. Μόλις καταλάβει τον εξυπηρετητή, παραμένει στην εξυπηρέτηση για κάποιο χρονικό διάστημα που αντιστοιχεί σε χρόνο εξυπηρέτησης.
5. Όταν ολοκληρωθεί η εξυπηρέτηση, "απελευθερώνει" τον εξυπηρετητή.
6. Εξέρχεται από το σύστημα.

Δείτε ότι τα βήματα 1, 2, 5 και 6 αντιπροσωπεύουν στιγμιαίες ενέργειες. Στο βήμα 3, αν ο εξυπηρετητής δεν είναι αδρανής, η οντότητα αναγκάζεται να παραμείνει στην ουρά για ένα χρονικό διάστημα που εξαρτάται από την κατάσταση του συστήματος, δηλαδή τον αριθμό άλλων οντοτήτων που βρίσκονται ήδη στην ουρά και τους αντίστοιχους χρόνους εξυπηρέτησης, καθώς και το χρόνο που απομένει για την οντότητα που βρίσκεται σε εξυπηρέτηση. Ομοίως, στο βήμα 4 η οντότητα αντιμετωπίζει μια καθυστέρηση χρόνου που αντιστοιχεί στον χρόνο εξυπηρέτησής της. Είναι σημαντικό να παρατηρήσουμε, ωστόσο, ότι ενώ στο βήμα 3 ο χρόνος, που η οντότητα περνάει στην ουρά, εξαρτάται από την κατάσταση, στο βήμα 4 η καθυστέρηση εξαρτάται μόνο από το χρόνο εξυπηρέτησης ο οποίος παρέχεται από μια γεννήτρια τυχαίων αριθμών.

Γενικά, αναπαριστάνουμε μια διεργασία ως μια ακολουθία λειτουργιών. Υπάρχουν δύο τύποι λειτουργιών:

1. **Λειτουργίες λογικής**: Στιγμιαίες ενέργειες που υπέστει η οντότητα που ενεργοποιεί αυτήν τη λειτουργία στη διεργασία της. Για παράδειγμα, ο έλεγχος μιας συνθήκης όπως "είναι ο εξυπηρετητής αδρανής;" ή ενημέρωση μιας δομής δεδομένων, όπως η καταγραφή της ώρας άφιξης της οντότητας.

2. **Λειτουργίες χρονικής καθυστέρησης**: Η οντότητα παραμένει σ' αυτήν τη λειτουργία για κάποια χρονική περίοδο. Υπάρχουν δύο τύποι λειτουργιών χρονικής καθυστέρησης:

   (α) *Καθορισμένος χρόνος*: Η καθυστέρηση είναι καθορισμένη, συνήθως από έναν αριθμό που λαμβάνεται από τη γεννήτρια τυχαίων αριθμών που χρησιμοποιείται στην προσομοίωση. Για παράδειγμα, ο χρόνος εξυπηρέτησης που εξαρτάται μόνο από κάποια προκαθορισμένη κατανομή του χρόνου εξυπηρέτησης.

   (β) *Απροσδιόριστος χρόνος*: Η καθυστέρηση εξαρτάται από την κατάσταση του συστήματος. Για παράδειγμα, ο χρόνος που περνάει σε ουρά αναμονή έως ότου η οντότητα μπορεί να καταλάβει έναν εξυπηρετητή.

Το σχέδιο[3] προσομοίωσης προσανατολισμένης στη διεργασία είναι ιδιαίτερα κατάλληλο για συστήματα ουράς, όπου είναι φυσικό να θεωρούμε τους πελάτες ως οντότητες που ρέουν μέσω ενός δικτύου διασυνδεδεμένων ουρών και εξυπηρετητών. Από πρακτικής άποψης, ο κώδικας υπολογιστών που απαιτείται για την προσομοίωση ενός τέτοιου συστήματος μέσω του σχεδίου προσανατολισμένου στη διεργασία είναι γενικά απλούστερος, απαιτώντας σημαντικά λιγότερες σειρές. Από την άλλη πλευρά, το σχέδιο χρονοπρογραμματισμού

3 Σχέδιο (scheme) είναι ένα συστηματικό πλάνο για την εφαρμογή μιας συγκεκριμένης ιδέας

συμβάντων, εξακολουθεί να παρέχει έναν πολύ γενικό τρόπο μοντελοποίησης οποιουδήποτε DES, που συμπεριλαμβάνει χαρακτηριστικά που το σχέδιο με προσανατολισμό στη διεργασία δεν είναι εξοπλισμένο για να χειριστεί. Οι περισσότερες διαθέσιμες στο εμπόριο γλώσσες προσομοίωσης είναι κυρίως προσανατολισμένες στη διεργασία. Πολλές από αυτές, ωστόσο, περιέχουν επίσης τη δυνατότητα κατασκευής μοντέλων προσομοίωσης χρησιμοποιώντας το γενικότερο σχέδιο χρονοπρογραμματισμού.

Συνοπτικά, οι κύριες συνιστώσες ενός σχεδίου προσομοίωσης προσανατολισμένης στη διεργασία είναι οι ακόλουθες:

1. **Οντότητες**: αντικείμενα που ζητούν εξυπηρέτηση (π.χ. εξαρτήματα σε ένα σύστημα παραγωγής, εργασίες σε ένα σύστημα υπολογιστή, πακέτα σε ένα δίκτυο επικοινωνίας, οχήματα σε ένα σύστημα μεταφοράς). Κάθε τύπος οντότητας χαρακτηρίζεται από μια συγκεκριμένη διεργασία στην οποία υποβάλλεται στο DES στο οποίο εισέρχεται. Για παράδειγμα, σε ένα υπολογιστικό σύστημα μπορεί να υπάρχουν δύο διαφορετικοί τύποι εργασιών που ακολουθούν διαφορετικές διεργασίες.

2. **Χαρακτηριστικά**: πληροφορίες που χαρακτηρίζουν μια συγκεκριμένη οντότητα οποιουδήποτε τύπου. Έτσι, συνήθως επισυνάπτουμε μια μοναδική εγγραφή σε κάθε οντότητα που αποτελείται από τα χαρακτηριστικά αυτής της οντότητας (π.χ., για ένα εξάρτημα σε ένα σύστημα παραγωγής, τα χαρακτηριστικά μπορεί να είναι η ώρα άφιξης του εξαρτήματος στο σύστημα, ο τύπος του και η "ημερομηνία λήξης", δηλαδή, η χρονική στιγμή κατά την οποία απαιτείται η έξοδός του από το σύστημα).

3. **Λειτουργίες διεργασίας**: οι στιγμιαίες ενέργειες ή οι καθυστερήσεις που αντιμετωπίζουν οι οντότητες όπως περιγράφεται παραπάνω.

4. **Πόροι**: αντικείμενα που εξυπηρετούν οντότητες (π.χ. μηχανήματα σε ένα σύστημα παραγωγής, επεξεργαστές σε ένα υπολογιστικό σύστημα, διακόπτες σε ένα τηλεφωνικό δίκτυο). Οι χρονικές καθυστερήσεις που υφίσταται μια οντότητα οφείλονται είτε στο ότι περιμένει να της διατεθεί ένας συγκεκριμένος πόρος, είτε στο ότι εξυπηρετείται σε αυτόν τον πόρο.

5. **Ουρές**: σύνολα οντοτήτων με κάποια κοινά χαρακτηριστικά, συνήθως το γεγονός ότι όλες περιμένουν να χρησιμοποιήσουν ένα συγκεκριμένο πόρο. Μια οντότητα που ρέει μέσω του συστήματος βρίσκεται πάντα σε κάποια ουρά εκτός και αν εξυπηρετείται από έναν πόρο.

## 1.6 Γλώσσες προσομοίωσης συστημάτων διακριτών-συμβάντων

Χρησιμοποιώντας είτε το χρονοπρογραμματισμό συμβάντων είτε το σχέδιο με προσανατολισμό στη διεργασία, είναι δυνατόν να δημιουργήσουμε ένα προσομοιωτή για ένα συγκεκριμένο σύστημα διακριτών συμβάντων, χρησιμοποιώντας τυπικές γλώσσες υπολογιστών, όπως η C++. Από την άλλη πλευρά, οργανώνοντας προσεκτικά τις κοινές συνιστώσες μιας προσομοίωσης (π.χ. γεννήτρια τυχαίων αριθμών, συλλογή δεδομένων και επεξεργασία) είναι δυνατή η ανάπτυξη μιας γλώσσας προσομοίωσης που παρέχει δομικά στοιχεία προσαρμοσμένα σε μοντέλα διακριτών συμβάντων. Υπάρχουν πολλές τέτοιες γλώσσες προσομοίωσης διαθέσιμες στο εμπόριο, εκ των οποίων κάποιες εξετάζονται παρακάτω εν συντομία. Οι περισσότερες από αυτές έχουν ενσωματωμένες εκτεταμένες δυνατότητες για παραγωγή τυχαίων αριθμών με βάση μια ποικιλία κατανομών πιθανότητας και παρέχουν τυπικές αναφορές εξόδου για κοινές μετρήσεις επίδοσης, όπως χρήση πόρων και στατιστικές μήκους ουράς. Από τα μέσα της δεκαετίας του 1990, σχεδόν όλο το λογισμικό προσομοίωσης διακριτών συμβάντων που χρησιμοποιείται, βασίζεται στον αντικειμενοστραφή προγραμματισμό και είναι εξοπλισμένο με γραφικές διεπαφές χρήστη που διευκολύνουν σε μεγάλο βαθμό τη δημιουργία μοντέλων καθώς και την απεικόνιση της εξέλιξης μιας διαδρομής. Το ζωντάνεμα (animation) επιτρέπει σε κάποιον να παρακολουθεί οντότητες που κινούνται εντός ενός συστήματος καθώς συγκρατούνται σε ουρές ή εξυπηρετούνται σε πόρους. Είναι ιδιαίτερα χρήσιμο για την παροχή πρόσθετων πληροφοριών σχετικά με την καθοδηγούμενη από συμβάντα δυναμική ενός συστήματος και τον εντοπισμό σφαλμάτων ενός μοντέλου. Μπορεί να χρησιμοποιηθεί, για παράδειγμα, για γρήγορη αναγνώριση κακού σχεδιασμού παρατηρώντας ότι ο αριθμός οντοτήτων σε κάποια ουρά αυξάνεται συνεχώς. Δεν είναι, ωστόσο, υποκατάστατο της σωστής ανάλυσης ενός στοχαστικού συστήματος διακριτών συμβάντων. Στην πραγματικότητα, συχνά οδηγεί κάποιον στην παγίδα του να πιστεύει ότι μετά την παρατήρηση ενός συστήματος, αποκτά, σε σύντομο χρονικό διάστημα, πλήρη κατανόηση της συμπεριφοράς του. Αυτό αγνοεί, ωστόσο, το γεγονός ότι συμβάντα που δεν έχουν ακόμη λάβει χώρα μπορούν να επηρεάσουν δραστικά το μέλλον του συστήματος.

**GPSS**

Το GPSS (General Purpose Simulation System) αναπτύχθηκε αρχικά από τον Gordon στην IBM στις αρχές της δεκαετίας του 1960. Από τότε έχει εξελιχθεί, με πιο πρόσφατη την έκδοση, GPSS/H, διαθέσιμη μέσω της εταιρείας Minuteman Software. Το GPSS/H είναι μια

γλώσσα βασισμένη σε ενότητες, με την οποία μπορεί κανείς να δημιουργήσει ένα μοντέλο προσομοίωσης προσανατολισμένο στη διεργασία. Οι οντότητες αναφέρονται ως "συναλλαγές" και τα χαρακτηριστικά ως "παράμετροι". Χρησιμοποιώντας τυπικές γραφικές αναπαραστάσεις, μπορεί κανείς να ορίσει με ευχέρεια ένα μοντέλο σε σχέση με ένα διάγραμμα μπλοκ για τα περισσότερα συστήματα που ενδιαφέρουν.

**SIMAN/ARENA**

Η SIMAN (SIMulation ANalysis) αναπτύχθηκε από τον Pegden στις αρχές της δεκαετίας του 1980. Η Arena είναι μια πιο πρόσφατη έκδοση που επιτρέπει σε κάποιον να δημιουργεί γραφικά μοντέλα και να ζωντανεύει την εκτέλεση μιας προσομοίωσης (Pegden et al. 1990) και (Kelton et al. 2007). Προς το παρόν είναι διαθέσιμο μέσω της εταιρείας Rockwell Automation. Η Arena επιτρέπει σε κάποιον να χρησιμοποιήσει το σχέδιο με προσανατολισμό στη διεργασία, το σχέδιο χρονοπρογραμματισμού συμβάντων ή έναν συνδυασμό και των δύο.

**SIMSCRIPT**

Το SIMSCRIPT, όπως το Arena, διαθέτει δυνατότητες σχεδίου με προσανατολισμό στη διεργασία και σχεδίου χρονοπρογραμματισμού συμβάντων. Το SIMSCRIPT αναπτύχθηκε από τον Markowitz στην εταιρεία Rand Corporation στις αρχές της δεκαετίας του 1960. Η πιο πρόσφατη έκδοση, SIMSCRIPT III (Rice et al. 2005), είναι διαθέσιμη μέσω της CACI Products Company. Η γενικότητα και το στυλ ελεύθερης μορφής του SIMSCRIPT το καθιστούν ιδιαίτερα ελκυστικό για τη μοντελοποίηση σύνθετων συστημάτων που δε χρειάζεται να χαρακτηρίζονται αποκλειστικά από μια δομή ουράς.

**SLAM**

Η SLAM (Simulation Language for Alternative Modeling), όπως η Arena και το SIMSCRIPT, παρέχει δυνατότητες σχεδίου με προσανατολισμό στη διεργασία και σχεδίου χρονοπρογραμματισμού συμβάντων. Το SLAM αναπτύχθηκε από τους Pegden και Pritsker στις αρχές της δεκαετίας του 1970 (Pritsker 1995) και είναι διαθέσιμο μέσω της Pritsker Corporation. Στη SLAM, η διεργασία μοντελοποίησης συνήθως περιλαμβάνει τον ορισμό δικτύου του συστήματος, αποτελούμενου από "κόμβους" και "διακλαδώσεις", που μπορεί να γίνει γραφικά. Παρέχεται, επίσης, ζωντάνεμα και γραφική αναπαράσταση των αναφορών εξόδου.

**EXTEND**

Το Extend είναι ένα αντικειμενοστραφές πακέτο προσομοίωσης με εκτεταμένες βιβλιοθήκες "αντικειμένων" για διάφορες εφαρμογές (Krahl 2002). Επιτρέπει στον χρήστη να "επεκτείνει" αυτές τις βιβλιοθήκες δημιουργώντας νέα αντικείμενα (δηλαδή, ενότητες δημιουργίας μοντέλων) μέσω απλών προτύπων που βασίζονται σε C. Παρέχει επίσης τη δυνατότητα κατασκευής ιεραρχικών μοντέλων και περιλαμβάνει μια ποικιλία εργαλείων εκτύπωσης σχεδίων και γραφικής εξόδου, καθώς και κίνησης. Διατίθεται από την εταιρεία Imagine That, Inc.

**SIMEVENTS**

Το SimEvents έχει σχεδιαστεί για την προσομοίωση των συστημάτων διακριτών συμβάντων, αλλά είναι ενσωματωμένο στο MATLAB και λειτουργεί εντός του Simulink, ενός παραδοσιακού προσομοιωτή που καθοδηγείται από το χρόνο, έτσι ώστε να είναι εξοπλισμένο με λειτουργικότητα που επιτρέπει την αποτελεσματική συνύπαρξη συνιστωσών που καθοδηγούνται από το χρόνο και από συμβάντα σε υβριδικά συστήματα (Cassandras et al. 2006). Επιπλέον, αυτός ο σχεδιασμός επιτρέπει στο SimEvents να επωφεληθεί από μια πλούσια συλλογή εργαλείων οπτικοποίησης, επεξεργασίας δεδομένων και υπολογισμού τόσο στο Simulink όσο και στο MATLAB, ενώ λειτουργεί ως καθαρός προσομοιωτής συστημάτων διακριτών συμβάντων, αν δεν εμπλέκονται συνιστώσες συστήματος καθοδηγούμενες από το χρόνο. Διατίθεται ως μέρος της σουίτας MATLAB από την εταιρεία The MathWorks, Inc.

Όπως ήδη αναφέρθηκε, πρόκειται για μια σύντομη και μερική έκθεση του λογισμικού προσομοίωσης συστημάτων διακριτών συμβάντων. Υπάρχουν πολλές επί πλέον γλώσσες προσομοίωσης (π.χ. GASP, SIMPAS, SIMULA, DEMOS, SIM++) και ορισμένες ειδικά προσανατολισμένες σε συγκεκριμένους τύπους συστημάτων διακριτών συμβάντων, όπως συστήματα παραγωγής και δίκτυα υπολογιστών. Βιβλία που παρέχουν λεπτομερείς περιγραφές και συγκρίσεις γλωσσών προσομοίωσης είναι τα εξής: Kreutzer 1986, Law and Kelton 1991, Robinson 2004. Είναι σαφές ότι πρόκειται για ένα ταχέως εξελισσόμενο πεδίο με νέες εκδόσεις λογισμικού προσομοίωσης να βρίσκονται σε εξέλιξη.

## 1.7 Εννοιολογική μοντελοποίηση

Η **εννοιολογική μοντελοποίηση** (Conceptual Modeling) είναι μία από τις πιο σημαντικές πτυχές ενός έργου προσομοίωσης. Περιλαμβάνει την αφηρημένη άποψη ενός συστήματος του πραγματικού κόσμου, προσδιορίζοντας τι πρέπει να μοντελοποιηθεί και πώς. Παρόλη τη σημασία της, η εννοιολογική μοντελοποίηση μόνο πρόσφατα κέρδισε δημοτικότητα στη

βιβλιογραφία της προσομοίωσης συστημάτων διακριτών συμβάντων. Κατά συνέπεια, ο ορισμός της εννοιολογικής μοντελοποίησης εξακολουθεί να εξελίσσεται και να ερμηνεύεται ελαφρώς διαφορετικά από διάφορους συγγραφείς. Ωστόσο υπάρχει μια κοινή συμφωνία ότι θα πρέπει να είναι ανεξάρτητη (η εννοιολογική μοντελοποίηση) από οποιοδήποτε υπόδειγμα υλοποίησης ή λύση λογισμικού και ότι θα πρέπει να ξεκινά νωρίς στον κύκλο-ζωής του έργου προσομοίωσης και να επανεξετάζεται επαναληπτικά. Ένα εννοιολογικό μοντέλο είναι ένα εργαλείο επικοινωνίας μεταξύ όλων των μερών σε ένα έργο προσομοίωσης. Επιπλέον, μεταξύ άλλων χαρακτηριστικών, παρέχει τη βάση τεκμηρίωσης του μοντέλου, καθοδηγεί την ανάπτυξη ενός μοντέλου υπολογιστή, παρέχει καθοδήγηση για πειράματα και είναι βοήθημα για την επαλήθευση και επικύρωση του μοντέλου. Από την άλλη πλευρά, μελετάται η εννοιολογική μοντελοποίηση μοντέλων μεγάλης κλίμακας και ως εκ τούτου η επαναχρησιμοποίηση των μοντέλων προσομοίωσης αντιμετωπίζεται ως το μεγαλύτερο κέρδος από τη δομημένη και την τυποποιημένη εννοιολογική μοντελοποίηση.

Ο Robinson (2012) δίνει ένα σύντομο εκπαιδευτικό υλικό για την εννοιολογική μοντελοποίηση. Ζητήματα της εννοιολογικής μοντελοποίησης και περαιτέρω ερευνητικές απαιτήσεις για την εννοιολογική μοντελοποίηση συνοψίστηκαν από τον Robinson (2006). Μεταξύ άλλων, ο Robinson προσδιορίζει τις ακόλουθες βασικές απαιτήσεις για έρευνα:

- συναίνεση σχετικά με τον ορισμό της εννοιολογικής μοντελοποίησης,
- τον προσδιορισμό των απαιτήσεων της εννοιολογικής μοντελοποίησης,
- την ανάπτυξη μεθόδων για το σχεδιασμό εννοιολογικών μοντέλων,
- τον ορισμό των πλαισίων μοντελοποίησης και
- πορεία προς πρότυπες μεθόδους αναπαράστασης και επικοινωνίας εννοιολογικών μοντέλων.

Μέχρι σήμερα δεν υπάρχει γενικά αποδεκτός ορισμός του εννοιολογικού μοντέλου και της σχετιζόμενης μοντελοποίησης εργασιών. (Zeigler 1987) είναι η πρώτη εργασία που δίνει σημασία σε διαφορετικά αφαιρετικά επίπεδα των μοντέλων προσομοίωσης, προσδιορίζοντας πέντε στοιχεία:

- το πραγματικό σύστημα,
- το πειραματικό πλαίσιο,
- το βασικό μοντέλο,
- το περικομμένο μοντέλο και
- το μοντέλο του υπολογιστή.

Το *πειραματικό πλαίσιο* δηλώνει τις συνθήκες υπό τις οποίες γίνεται η παρατήρηση του πραγματικού συστήματος (μοτίβα εισόδου και εξόδου). Το *βασικό μοντέλο* είναι μια υποθετική, πλήρης αναπαράσταση του πραγματικού κόσμου, η οποία προφανώς δεν είναι γνωστή. Το *περικομμένο μοντέλο* είναι μια απλοποιημένη έκδοση του βασικού μοντέλου που μπορεί να αντικατοπτρίζει όλα τα μοτίβα εισόδου και εξόδου, και το *μοντέλο υπολογιστή* δηλώνει την υλοποίηση του περικομμένου μοντέλου. Σύμφωνα με την ορολογία του Zeigler, το περικομμένο μοντέλο είναι ανεξάρτητο από το μοντέλο υπολογιστή και αναπαριστάνει αυτό που τώρα γίνεται κατανοητό ως εννοιολογικό μοντέλο.

Ο Balci (1994) αναλύει τα πρώτα στάδια μιας μελέτης προσομοίωσης και συνεχώς επαναπροσδιορίζει επαναλαμβανόμενα βήματα που περιλαμβάνουν:

- τη διαμόρφωση μοντέλου,
- την αναπαράσταση μοντέλου και
- τον προγραμματισμό.

Ο Nance (1994) διακρίνει:

- το μοντέλο που υπάρχει στο μυαλό του σχεδιαστή μοντέλων,
- το εννοιολογικό μοντέλο και
- την αναπαράστασή του για επικοινωνία (το επικοινωνιακό μοντέλο).

Στον αμυντικό τομέα, ο Pace (1999) ερμηνεύει το εννοιολογικό μοντέλο ως *τον ορισμό του τι πρέπει να μοντελοποιηθεί και πώς*, παρέχοντας πληροφορίες για παραδοχές, αλγόριθμους, χαρακτηριστικά, σχέσεις και δεδομένα. Περαιτέρω, ο Lacy και άλλοι (Lacy et al. 2001) διακρίνουν μεταξύ:

- ενός προβλήματος προσανατολισμένου στον τομέα που περιγράφει τις απαιτήσεις του μοντέλου και
- ενός μοντέλου προσανατολισμένου στο σχεδιασμό που χρησιμοποιείται ως βάση για την υλοποίηση μοντέλου.

Πιο πρόσφατα, οι Balci και Ormsby (2007) συζητούν την εννοιολογική μοντελοποίηση για προσομοιώσεις μεγάλης κλίμακας. Με αυτόν τον τρόπο, εντόπισαν τρία σημαντικά επίπεδα αφαίρεσης, από τα οποία διέρχεται ένα μοντέλο προσομοίωσης στον κύκλο ζωής του:

- το εννοιολογικό μοντέλο προσομοίωσης,
- ο σχεδιασμός του μοντέλου προσομοίωσης,
- η υλοποίηση του μοντέλου προσομοίωσης.

Υποστηρίζουν ότι η υλοποίηση επιτυγχάνεται χρησιμοποιώντας γλώσσες προγραμματισμού όπως η Java, η C++ ή η C#. Η μετάβαση από το επίπεδο σχεδίασης στο επίπεδο υλοποίησης πραγματοποιείται με τη χρήση κοσμοθεώρησης διακριτών-συμβάντων, π.χ. χρονοπρογραμματισμός συμβάντων, τριφασική προσέγγιση ή αλληλεπίδραση διαδικασιών. Ως εκ τούτου, το μοντέλο σχεδίασης είναι μια περιγραφή των συνιστωσών του μοντέλου υλοποίησης, χρησιμοποιώντας αντικειμενοστραφή ή διαδικαστικά υποδείγματα. Το εννοιολογικό μοντέλο αναπαριστάνει το υψηλότερο επίπεδο αφαίρεσης και είναι το μοντέλο που μορφοποιήθηκε στο το μυαλό του σχεδιαστή μοντέλων. Σημειώνουμε ότι η δουλειά τους βασίζεται σε μοντέλα μεγάλης κλίμακας του αμυντικού τομέα. Ο αριθμός των σχεδιαστών μοντέλων και των ειδικών του τομέα, που συμμετέχουν σε αυτές τις μελέτες, είναι συνήθως μεγάλος. Επομένως οι απαιτήσεις για την εννοιολογική μοντελοποίηση διαφέρουν από τα προσανατολισμένα στις επιχειρήσεις μοντέλα που εστιάζονται στην τεκμηρίωση, την επικοινωνία και την επαναχρησιμοποίηση.

Οι Birta and Arbez (2007) δηλώνουν ότι η εννοιολογική μοντελοποίηση ασχολείται με την ανάπτυξη μιας ουσιαστικής αναπαράστασης του πραγματικού συστήματος. Περαιτέρω, ορίζουν δύο βασικές απαιτήσεις για ένα εννοιολογικό μοντέλο:

- πρέπει να είναι αρκετά διαφανές για να χρησιμοποιείται ως μέσο για συζητήσεις όλων των ενδιαφερομένων στο έργο, και
- πρέπει να είναι αρκετά διαφανές ώστε να μπορεί να χρησιμεύσει ως προδιαγραφή για το πρόγραμμα υπολογιστή.

Οι Karagöz και Demirörs (2010) δηλώνουν ότι η εννοιολογική μοντελοποίηση είναι ένα εργαλείο που παρέχει μια σαφή κατανόηση του τομέα-στόχου ή του προβλήματος. Ως εκ τούτου, είναι μια απλοποιημένη αναπαράσταση του πραγματικού συστήματος που περιλαμβάνει δομικά χαρακτηριστικά και χαρακτηριστικά συμπεριφοράς.

Ο πιο αναγνωρισμένος ορισμός ενός εννοιολογικού μοντέλου δίνεται από το Robinson. Ο Robinson υποστηρίζει ότι ένα εννοιολογικό μοντέλο προέρχεται από την κατανόηση της προβληματικής κατάστασης, η οποία δεν είναι μέρος του εννοιολογικού μοντέλου. Επομένως είναι μερική περιγραφή του πραγματικού κόσμου που επαρκεί για την αντιμετώπιση της προβληματικής κατάστασης. Επιπλέον, αποτελείται από τέσσερες κύριες συνιστώσες:

- αντικειμενικοί σκοποί,
- είσοδοι (πειραματικοί παράγοντες), έξοδοι (απαντήσεις), και
- το περιεχόμενο του μοντέλου.

Το περιεχόμενο διαιρείται περαιτέρω στην εμβέλεια του μοντέλου και το επίπεδο λεπτομέρειας. Κατά τον προσδιορισμό του περιεχομένου του μοντέλου πρέπει να γίνουν υποθέσεις και απλοποιήσεις. Αυτό οδηγεί στον ακόλουθο ορισμό ενός εννοιολογικού μοντέλου (Robinson 2008):

> *Το εννοιολογικό μοντέλο είναι μια περιγραφή, όχι ειδικά σε σχέση με λογισμικό, του μοντέλου προσομοίωσης, περιγράφοντας τους αντικειμενικούς σκοπούς, τις εισόδους, τις εξόδους, το περιεχόμενο, τις παραδοχές και τις απλοποιήσεις του μοντέλου.*

Το έργο της εννοιολογικής μοντελοποίησης σχετίζεται, επομένως, με την κατανόηση της προβληματικής κατάστασης και την ταυτοποίηση και προσδιορισμό των συνιστωσών ενός εννοιολογικού μοντέλου. Επιπλέον, στο άρθρο (Robinson 2011) ο σχεδιασμός του μοντέλου προστέθηκε ως τεχνούργημα στον κύκλο-ζωής του έργου.

Σημειώνουμε ότι ο ορισμός του Robinson είναι εμπνευσμένος από διάφορες κλίμακες και τομείς προβλημάτων. Ενώ των Balci και Ormsby βασίζεται σε στρατιωτικές εφαρμογές μεγάλης κλίμακας, που συνήθως περιλαμβάνουν εκατοντάδες προγραμματιστές και εμπειρογνώμονες, κωδικοποίηση από την αρχή και κύκλο-ζωής μακρών έργων, του Robinson αποβλέπει σε μοντέλα προσανατολισμένα στις επιχειρήσεις. Αυτά τα μοντέλα έχουν συνήθως αρκετά μικρό χρόνο ανάπτυξης, κατασκευάζονται χρησιμοποιώντας εμπορικά οπτικά εργαλεία λογισμικού και η επαναχρησιμοποίηση δεν αποτελεί μείζον ζήτημα. Επομένως ο Robinson ενδιαφέρεται περισσότερο για την αφαίρεση ενός πραγματικού συστήματος και των εμπλεκόμενων συνιστωσών, όπου η άποψη των Balci και Ormsby έχει πιο περιγραφικό χαρακτήρα.

Η κοινή αντίληψη είναι ότι ένα εννοιολογικό μοντέλο πρέπει να είναι ανεξάρτητο από το λογισμικό και ότι πρέπει να γίνει μια σαφής διάκριση μεταξύ της εννοιολογικής μοντελοποίησης και της σχεδίασης μοντέλου. Ωστόσο, κατά τη συζήτηση των υπαρχόντων πλαισίων για την εννοιολογική μοντελοποίηση, θα καταστεί σαφές ότι πιστεύουμε ότι το περιεχόμενο ενός μοντέλου δεν αντιπροσωπεύεται επαρκώς από υπάρχουσες προσεγγίσεις. Επιπλέον, συμφωνούμε ότι ένας σαφής διαχωρισμός μεταξύ των εννοιολογικών μοντέλων και των σχεδιαστικών μοντέλων πρέπει πάντα να διατηρείται προς αποφυγήν της οδήγησης της μοντελοποίησης από υποδείγματα υλοποίησης, όπως οι κοσμοθεωρήσεις συστημάτων διακριτών συμβάντων.

## 1.8 Τεκμηρίωση του εννοιολογικού μοντέλου

Όπως αναφέρθηκε παραπάνω, το εννοιολογικό μοντέλο δεν εκφράζεται πάντα ρητά, αλλά μπορεί να παραμείνει στο μυαλό του μοντελοποιητή. Γι' αυτό το λόγο, είναι καλή πρακτική να τεκμηριώνουμε το εννοιολογικό μοντέλο και με αυτόν τον τρόπο να παράσχουμε ένα μέσο επικοινωνίας μεταξύ όλων των μερών σε μια μελέτη προσομοίωσης. Δεν υπάρχουν καθορισμένα πρότυπα για την τεκμηρίωση εννοιολογικών μοντέλων προσομοίωσης συστημάτων διακριτών συμβάντων, αλλά έχει προταθεί μια σειρά προσεγγίσεων, που συμπεριλαμβάνουν:

- Λίστα συνιστωσών
- Διάγραμμα ροής διεργασίας
- Διάγραμμα κύκλου δραστηριοτήτων
- Διάγραμμα ροής
- Λίστα υποθέσεων και απλουστεύσεων
- Ενοποιημένη γλώσσα μοντελοποίησης (UML)
- Δίχτυα Petri
- Προδιαγραφή συνθηκών

Η τεκμηρίωση ενός εννοιολογικού μοντέλου πρέπει να παραμένει απλή και να επικεντρώνεται στον προσδιορισμό του τι πρόκειται να μοντελοποιηθεί και τι δεν πρέπει να μοντελοποιηθεί. Δεν υπάρχει ανάγκη για περίπλοκη τεκμηρίωση, επειδή λεπτομερείς αποφάσεις, σχετικά με τον τρόπο μοντελοποίησης, δε λαμβάνονται κατά την εννοιολογική μοντελοποίηση. Αυτές οι αποφάσεις λαμβάνονται κατά τη σχεδίαση του μοντέλου βάσει του εννοιολογικού μοντέλου. Για παράδειγμα, στην εννοιολογική μοντελοποίηση λαμβάνεται η απόφαση να μοντελοποιηθεί ένα πρόγραμμα πτήσεων σε αεροδρόμιο. Στο σχεδιασμό του μοντέλου καθορίζεται ο τρόπος με τον οποίο πρόκειται να μοντελοποιηθεί αυτό το πρόγραμμα πτήσεων. Ως εκ τούτου, η τεκμηρίωση του εννοιολογικού μοντέλου χρειάζεται μόνο να δηλώσει ότι το πρόγραμμα πτήσεων πρέπει να μοντελοποιηθεί, ενώ η τεκμηρίωση του σχεδιασμού μοντέλου χρειάζεται να παράσχει λεπτομέρειες σχετικά με τον τρόπο μοντελοποίησης του προγράμματος πτήσεων.

## 1.9 Διάγραμμα ροής διεργασίας

Σε αυτήν την προσέγγιση το εννοιολογικό μοντέλο αναπαριστάνεται ως μια ροή διεργασιών ή ένας χάρτης διεργασιών, που δείχνει κάθε συνιστώσα του συστήματος σε μια ακολουθία και περιλαμβάνει κάποια περιγραφή λεπτομερειών του μοντέλου. Οι διεργασίες μπορούν να εμφανίζονται ως πλαίσια και οι ουρές ως κύκλοι. Το παρακάτω σχήμα είναι ένα παράδειγμα διαγράμματος ροής διεργασίας για μία ουρά ενός εξυπηρετητή.


**1.6** Διάγραμμα ροής διεργασίας για μία ουρά ενός εξυπηρετητή

Σε αυτό το διάγραμμα, η λεπτομέρεια των συνιστωσών δηλώνεται εντός παρενθέσεων. Είναι μια αρκετά απλή προσέγγιση και η οπτικοποιημένη αναπαράσταση είναι χρήσιμη για την εμφάνιση της ροής διεργασίας. Δεδομένου ότι πολλά πακέτα προσομοίωσης χρησιμοποιούν παρόμοια αναπαράσταση, αυτή η προσέγγιση είναι πολύ χρήσιμη. Ωστόσο είναι ακόμα δύσκολο να καταγράψουμε πιο περίπλοκη λογική. Υπάρχει λογισμικό ειδικά για τη καταγραφή των διεργασιών, για παράδειγμα, το ARIS Express.

## 1.10 Πλαίσια μοντελοποίησης

Ένα **πλαίσιο μοντελοποίησης** (modeling frames) υπερβαίνει την ιδέα καθοδήγησης αρχών και μεθόδων απλούστευσης ενός μοντέλου, παρέχοντας ένα συγκεκριμένο σύνολο βημάτων που καθοδηγούν ένα σχεδιαστή μοντέλου στην ανάπτυξη ενός εννοιολογικού μοντέλου. Έχουν γίνει κάποιες προσπάθειες για την παροχή τέτοιων πλαισίων από τον Shannon (1975) που περιγράφει τέσσερα βήματα:

- προσδιορισμός του σκοπού του μοντέλου,
- προδιαγραφές των συνιστωσών του μοντέλου,
- προσδιορισμός των παραμέτρων και των μεταβλητών που σχετίζονται με τις συνιστώσες, και
- προσδιορισμός των σχέσεων μεταξύ των συνιστωσών, των παραμέτρων και των μεταβλητών.

Τόσο ο Nance όσο και ο Pace έχουν επινοήσει πλαίσια που σχετίζονται κυρίως με την ανάπτυξη μοντέλων μεγάλης κλίμακας στον στρατιωτικό τομέα. Ο Nance (1994) περιγράφει την κωνική μεθοδολογία. Πρόκειται για μια αντικειμενοστραφή, ιεραρχική γλώσσα προδιαγραφών που αναπτύσσει τον ορισμό μοντέλου από πάνω προς τα κάτω και τις προδιαγραφές μοντέλου από κάτω προς τα πάνω. Οι Balci και Nance (1985) επικεντρώνονται σε μια διαδικασία μορφοποίησης προβλημάτων. Οι Arthur και Nance (2007) προσδιορίζουν τη δυνατότητα υιοθέτησης προσεγγίσεων **τεχνολογίας απαιτήσεων λογισμικού** (software requirements engineering - SRE) για την ανάπτυξη μοντέλων προσομοίωσης. Υπάρχουν ελάχιστες ενδείξεις ότι η SRE πράγματι υιοθετήθηκε από σχεδιαστές μοντέλων προσομοίωσης.

Ο Pace (1999) διερευνά μια προσέγγιση τεσσάρων σταδίων για την ανάπτυξη ενός εννοιολογικού μοντέλου, παρόμοια με εκείνη του Shannon:

- συλλογή έγκυρων πληροφοριών σχετικών με το πεδίο του προβλήματος,
- προσδιορισμός των οντοτήτων και διαδικασιών που πρέπει να αναπαρασταθούν,
- ταυτοποίηση των στοιχείων προσομοίωσης, και
- ταυτοποίηση σχέσεων μεταξύ των στοιχείων προσομοίωσης.

Προσδιορίζει, επίσης, έξι κριτήρια για τον προσδιορισμό των στοιχείων που θα συμπεριληφθούν στο εννοιολογικό μοντέλο. Αυτά τα κριτήρια εστιάζουν στην αντιστοιχία μεταξύ αντικειμένων του πραγματικού κόσμου και των αντικειμένων της προσομοίωσης.

Στο πεδίο που μας ενδιαφέρει - η προσομοίωση για τη μοντελοποίηση λειτουργικών συστημάτων - υπάρχει πολύ περιορισμένη εργασία πάνω στα εννοιολογικά πλαίσια μοντελοποίησης. Οι Brooks και Tobias (1996) προτείνουν ένα πλαίσιο για την εννοιολογική μοντελοποίηση, αλλά δεν επεκτείνουν περαιτέρω την ιδέα. Οι πρόσφατες εργασίες των Guru και Savory (2004), van der Zee και van der Vorst (2005) προτείνουν πιο λεπτομερή εννοιολογικά πλαίσια μοντελοποίησης. Οι Guru και Savory προτείνουν ένα σύνολο προτύπων μοντελοποίησης χρήσιμων για τη μοντελοποίηση φυσικών συστημάτων ασφάλειας. Εν τω μεταξύ, οι van der Zee και van der Vorst προτείνουν ένα πλαίσιο προσομοίωσης αλυσίδας εφοδιασμού. Και οι δύο στοχεύουν σε αντικειμενοστραφή υλοποίηση ενός μοντέλου προσομοίωσης που βασίζεται στον υπολογιστή. Εν τω μεταξύ, ο Kotiadis (2007) εξετάζει τις ιδέες της εταιρείας Soft Operational Research, και συγκεκριμένα τη Soft Systems Methodology (SSM), για την υποβοήθηση της διαδικασίας εννοιολογικής μοντελοποίησης. Χρησιμοποιεί τη SSM για να κατανοήσει ένα περίπλοκο σύστημα υγειονομικής περίθαλψης

και στη συνέχεια παράγει το εννοιολογικό μοντέλο προσομοίωσης από το "σκόπιμο μοντέλο δραστηριότητας" του SSM.

# 2. 2. Λογισμικό ανοιχτού κώδικα για την DESS

## 2.1 Εισαγωγή

Εμπορικά διαθέσιμο λογισμικό για την DESS υπάρχει εδώ και αρκετές δεκαετίες και έχει συμβάλει σημαντικά στη διάδοση της DESS στην ακαδημαϊκή και βιομηχανική κοινότητα. Η ισχύς αυτών των πακέτων έγκειται στο γεγονός ότι παρέχουν στον χρήστη γραφικά εργαλεία για μοντελοποίηση, εντοπισμό σφαλμάτων και πειραματισμό, με προηγμένες οπτικές δυνατότητες που ενισχύουν την ανάπτυξη μοντέλων. Από την άλλη πλευρά, η έλλειψη πρόσβασης στον πηγαίο κώδικα και η έλλειψη γνώσης του τρόπου λειτουργίας αυτών των εργαλείων, καθιστούν πιο δύσκολη την προσαρμογή τους. Ένας άλλος παράγοντας που εμποδίζει την προσαρμοστικότητα είναι το γεγονός ότι το εμπορικά διαθέσιμο λογισμικό συχνά στερείται διάρθρωσης ενοτήτων (modularity), καθώς έρχεται ως ένα μονολιθικό προϊόν. Η επαναχρησιμοποίηση μπορεί, επίσης, να παρεμποδίζεται από τη χρήση εργαλείων εμπορικά διαθέσιμου λογισμικού για την DESS. Το γεγονός ότι διαφορετικοί οργανισμοί χρησιμοποιούν διαφορετικά εργαλεία, σε μια αγορά όπου δεν ξεχωρίζει κάποιο από αυτά (Wiedemann, 2002), καθιστά πολύ πιο δύσκολη την ανταλλαγή μοντέλων και συνιστωσών. Το υψηλό κόστος άδειας χρήσης των εμπορικών πακέτων λογισμικού είναι συχνά ο πιο σημαντικός παράγοντας για την απροθυμία των οργανισμών να υιοθετήσουν την DESS, ιδίως οι μικρομεσαίες επιχειρήσεις. Το κόστος ιδιοκτησίας μπορεί να κυμαίνεται από μερικές εκατοντάδες έως πολλές χιλιάδες ευρώ. Αλλά ακόμη και για μεγάλες πολυεθνικές τα πακέτα DESS προκαλούν πρόσθετα έξοδα που σχετίζονται με παράγοντες όπως η εκπαίδευση, η προσθήκη plug-ins και οι αναβαθμίσεις. Το άρθρο (Schriber et al 2012) παρουσιάζει πολλά παραδείγματα όπου διαφορετικά εργαλεία προσομοίωσης επεξεργάζονται συμβάντα με διαφορετική ακολουθία, με αποτέλεσμα μερικές φορές διαφορετικά αποτελέσματα.

Ο ανοιχτός κώδικας είναι προσβάσιμος από έναν προχωρημένο χρήστη να τον ελέγξει και να τον επεξεργαστή. Φυσικά, για να είναι έγκυρο το παραπάνω επιχείρημα, ο κώδικας πρέπει να είναι ευανάγνωστος και κατανοητός. Υπάρχει η δυνατότητα δημιουργίας ενός αποθετηρίου ανοιχτών συνιστωσών DESS που ένας τρίτος χρήστης μπορεί να λάβει και να υιοθετήσει για τις ανάγκες του. Η απουσία αποκλειστικών ζητημάτων αδειοδότησης θα μπορούσε ενδεχομένως να κάνει ένα τέτοιο αποθετήριο καθολικό και επεκτάσιμο, εάν υπήρχε μια ενεργή κοινότητα ανοιχτού κώδικα DESS. Ωστόσο, για να επιτευχθεί αυτό, θα ήταν απαραίτητο ένα πρότυπο τεκμηρίωσης. Τέλος, καθώς η προσομοίωση είναι μια υπολογιστικά

περίπλοκη τεχνική και μια μηχανή προσομοίωσης ανοιχτού κώδικα δεν επιβάλλει πολυπλοκότητες αδειοδότησης κατά τη χρησιμοποίηση σε ένα σύμπλεγμα υπολογιστών, η διαθεσιμότητα της προσομοίωσης ανοιχτού κώδικα θα έπρεπε να προωθεί το υπολογιστικό νέφος (Taylor et al 2014).

## 2.2 Επισκόπηση των εργαλείων

Η Java και η C++ φαίνεται να είναι οι πιο δημοφιλείς γλώσσες προγραμματισμού σε εργαλεία DESS, με την πρώτη να χρησιμοποιείται σε 20 εργαλεία και τη δεύτερη σε 19. Σημειώνουμε ότι ορισμένα εργαλεία ενδέχεται να χρησιμοποιούν περισσότερες από μία γλώσσες, για παράδειγμα το SimKit που παρέχει υλοποιήσεις τόσο σε C++ όσο και σε Java. Θεωρούμε ότι η επίδοση αυτών των δύο γλωσσών ώθησε τους συγγραφείς να τις χρησιμοποιήσουν στα πλαίσια της DESS. Η Python χρησιμοποιείται σε τρία εργαλεία, η C# σε δύο, η JavaScript και η Scala σε ένα η καθεμία. Επίσης το Root-sim είναι γραμμένο στη C. Γενικά η χρήση της C στην DESS μπορεί να μοιάζει περίεργη, καθώς δεν υποστηρίζει την αντικειμενοστρέφια και η DESS είναι ένα κλασικό παράδειγμα όπου η αντικειμενοστρέφια υιοθετείται διαισθητικά. Είναι ενδεικτικό ότι η γλώσσα προσομοίωσης Simula, βασισμένη στην ALGOL, ήταν ιστορικά η πρώτη που εισήγαγε την έννοια της κλάσεως. Παρ' όλα αυτά, το Root-sim εστιάζεται παράλληλα εκεί όπου χρησιμοποιείται η γλώσσα C.

Χρησιμοποιούνται συχνότερα διαφορετικές εκδόσεις της GPL (General Public Licence) και της LGPL (Lesser General Public Licence). Υπάρχει ένα θέμα με την άδεια GPL, καθώς οι περιορισμοί, που επιβάλλει, ενδέχεται να αποθαρρύνουν τις εταιρείες να το υιοθετήσουν για λύσεις σε ανοιχτό κώδικα. Πράγματι τέτοια εργαλεία δεν θα ήταν δυνατόν να τροποποιηθούν και να διανεμηθούν εκ νέου σε ιδιόκτητη μορφή. Οι άδειες τύπου BSD (Berkeley Software Distribution) χρησιμοποιούνται από τρία εργαλεία, ενώ οι άδειες τύπου MIT χρησιμοποιούνται από δύο εργαλεία. APL σημαίνει "Academic Public License", που χρησιμοποιείται στο OMNeT++ και CCA σημαίνει "Creative Commons Attribution" που χρησιμοποιείται στο Over-Sim. Κάποια εργαλεία, όπως το PARSEC και το SIMCAN, χρησιμοποιούν πολλούς τύπους αδειών, που βασίζονται σε εργαλεία ανοιχτού κώδικα τα οποία αξιοποιούν.

Περισσότερα από τα μισά από τα σαράντα τέσσερα εργαλεία DESS έχουν γενικό χαρακτήρα. Γενικά είναι καλό να έχουμε γενικά εργαλεία τα οποία ενδεχομένως μπορούν να επεκταθούν σε συγκεκριμένους τομείς. Υποστηρίζεται ότι τα περιβάλλοντα DESS συγκεκριμένων τομέων μπορούν να διευκολύνουν την ανάπτυξη μοντέλων και ίσως είναι μια ευκαιρία για την DESS

να παρέχει τέτοιες λύσεις. Τα εργαλεία, που εστιάζονται στους τομείς που μας ενδιαφέρουν, είναι λιγοστά, δηλαδή δύο στην παραγωγή και ένα στη διαχείριση της εφοδιαστικής αλυσίδας. Υπολογιστές και δίκτυα επικοινωνίας, από την άλλη πλευρά, φαίνεται να είναι αρκετά ενεργά με πέντε εργαλεία αφιερωμένα σε αυτόν τον τομέα.

Δεκατέσσερα εργαλεία δεν παρέχουν καμία τεκμηρίωση οποιασδήποτε μορφής. Όσον αφορά την επικοινωνία μεταξύ των μελών του εργαλείου, η κατάσταση είναι χειρότερη καθώς τριάντα ένα από τα εργαλεία δεν παρέχουν αποκλειστικά μέσα για αυτήν (την επικοινωνία). Οι λίστες αλληλογραφίας είναι το πιο κοινό μέσο επικοινωνίας που χρησιμοποιείται. Επίσης, σε περισσότερα από τα μισά από τα εργαλεία δεν είναι δυνατόν να βρεθεί κάποιο σύστημα ελέγχου της έκδοσης για τη διαχείριση διαφορετικών εκδόσεων. Τα παραπάνω δίνουν τη συνολική εικόνα ότι πολλά εργαλεία παραμελούν να χρησιμοποιήσουν δημοφιλείς πρακτικές ανοιχτού κώδικα, γεγονός που καθιστά πιο δύσκολη την προσέλκυση προσοχής σε αυτά τα εργαλεία.

Το **OMNeT++** είναι ένα δημοφιλές εργαλείο DESS ανοιχτού κώδικα του οποίου η κύρια εστίαση είναι η δημιουργία προσομοιώσεων δικτύου επικοινωνίας. Το εργαλείο είναι διαθέσιμο στο κοινό από τον Σεπτέμβριο του 1997 και διατηρεί μια αρκετά ενεργή κοινωνία χρηστών που αποτελείται από ανώνυμους χρήστες και πολλά εγγεγραμμένα πανεπιστήμια. Ένα ειδικό εργαστήριο προγραμματιστών του OMNeT++ πραγματοποιείται κάθε χρόνο από το 2008. Το OMNeT++ χρησιμοποιείται ευρέως, συνήθως για την προσομοίωση πρωτοκόλλων επικοινωνίας. Η εκπαίδευση στην προσομοίωση είναι ένας άλλος τομέας στον οποίο εστιάζεται το εργαλείο και υπήρξαν επίσης εφαρμογές σε πολλά εκπαιδευτικά ιδρύματα. Ένας άλλος δείκτης της επιτυχίας του OMNeT++, είναι η ύπαρξη άλλων εργαλείων ανοιχτού κώδικα που είναι χτισμένα πάνω σ' αυτό, όπως το OverSim ή το SIMCAN.

Το OMNeT++ προσφέρει προηγμένες δυνατότητες τόσο για προγραμματιστές όσο και για χρήστες. Για την πρώτη κατηγορία παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που βασίζεται στην Eclipse. Στη δεύτερη κατηγορία, προσφέρει χρόνο εκτέλεσης GUI, όπου ο χρήστης μπορεί να μεταφέρει και να αποθέσει ενότητες OMNeT++ και να τις συνδέσει με ένα μοντέλο. Το εργαλείο προσφέρει πολλά βοηθήματα εκπαίδευσης και έγγραφα υποστήριξης, ενώ οι επικοινωνίες διευκολύνονται μέσω ενός wiki και μιας λίστας αλληλογραφίας, τα οποία χρησιμοποιούνται ενεργά από την κοινωνία.

Η τελευταία έκδοση του OMNeT++ είναι η 6.0.

Το **NS-3** είναι επίσης ένα πολύ ενεργό εργαλείο ανοιχτού κώδικα με πολύ εντυπωσιακή δομή. Ο κώδικας λειτουργεί σε πολλές αποθήκες που χρησιμοποιούν το εργαλείο Mercurial, για κατανεμημένο έλεγχο εκδόσεων, και ο κώδικας λαμβάνει συχνές ενημερώσεις (http://code.nsnam.org/). Η επικοινωνία παρέχεται μέσω καναλιών wiki και irc, ενώ η τεκμηρίωση παρέχεται σε διάφορες μορφές. Το NS-3 επικεντρώνεται επίσης στον τομέα των δικτύων υπολογιστών και δεν κατέστη δυνατό να βρεθούν εφαρμογές που το επεκτείνουν σε άλλους τομείς. Προσφέρει επίσης plug-ins για μια γραφική διεπαφή, αλλά δε φαίνεται να είναι τόσο ευθυγραμμισμένη με την κατάσταση ανάπτυξης του εργαλείου όσο η GUI του OMNeT++. Η άδεια χρήσης του λογισμικού είναι η GPL. Τόσο το NS-3 όσο και το OMNeT+ + μπορούν να δώσουν πληροφορίες για το πώς ένα εργαλείο DESS ανοιχτού κώδικα μπορεί να γίνει επιτυχές και να προσελκύσει προγραμματιστές.

Η τελευταία έκδοση του NS είναι η 3.33.

Το **SimPy** είναι ένα εργαλείο που υλοποιεί τη διαδικασία αλληλεπίδρασης worldview, χρησιμοποιώντας τη γεννήτρια συναρτήσεων της Python. Το SimPy ενημερώνεται ενεργά τα τελευταία χρόνια και τον Οκτώβριο του 2013 κυκλοφόρησε το SimPy3. Ένας αρνητικός παράγοντας είναι ότι το SimPy3 έχει διαφορετική σύνταξη από το SimPy2, πράγμα που σημαίνει ότι το εργαλείο δε διατήρησε τη συμβατότητα προς τα πίσω. Ωστόσο το SimPy3 φαίνεται να κάνει καθαρότερη χρήση των γεννητριών Python και λαμβάνει καθημερινές ενημερώσεις. Η λίστα αλληλογραφίας του SimPy συμπληρώνεται, επίσης, καθημερινά με ερωτήσεις και απαντήσεις σχετικά με το εργαλείο. Το εργαλείο δεν έχει GUI και φαίνεται ότι απευθύνεται κυρίως σε προγραμματιστές. Η τεκμηρίωση του SimPy3 είναι επαρκής, προσφέροντας γνώση για τις ενότητες και πολλά παραδείγματα. Η νέα έκδοση χρησιμοποιεί την MIT άδεια.

Το γεγονός ότι το SimPy βασίζεται στην Python έχει πλεονεκτήματα και μειονεκτήματα. Η Python έχει το πλεονέκτημα να είναι πιο ελκυστική για ερευνητές που δεν είναι απαραίτητα ειδικοί επί του λογισμικού. Παρέχει επίσης μέσα για την ανάπτυξη κώδικα που ακολουθεί αυτονόητη και καθαρή σύνταξη. Τα παραπάνω χαρακτηριστικά μπορεί να την κάνουν ιδανική ως βάση ενός εργαλείου DESS ανοιχτού κώδικα σε βιομηχανικά περιβάλλοντα, που θα επιτύχει πραγματικά στη διαμόρφωση μιας κοινότητας προγραμματιστών, μοντελιστών και χρηστών. Επιπλέον, υπάρχουν πολλά δημοφιλή επιστημονικά πακέτα γραμμένα στην Python, όπως το SciPy ή το scikit-learn, οπότε μια βιβλιοθήκη βασισμένη στην Python έχει τη δυνατότητα να ενσωματωθεί σε αυτά. Ωστόσο η Python, ως γλώσσα δέσμης ενεργειών,

είναι εξ ορισμού πιο αργή από τις στατικές γλώσσες όπως η C++ ή η Java, αν και έχουν γίνει προσπάθειες βελτίωσης της επίδοσης της Python γενικά (Lavrijsen 2012) και του SimPy συγκεκριμένα (Bahouth et al, 2007).

Η τελευταία έκδοση του SimPy είναι η 4.0.2.

Το **JaamSim**, που αναπτύχθηκε από την καναδική εταιρεία Ausenco, κυκλοφόρησε το 2013 ως ανοιχτός κώδικας, ωστόσο η διαδικασία ανάπτυξής του ξεκίνησε πριν από μια δεκαετία. Από όλα τα εργαλεία DESS ανοιχτού κώδικα που είδαμε, το JaamSim είναι αυτό με το πιο εντυπωσιακό 3D περιβάλλον εργασίας για το χρήστη, που μπορεί να ανταγωνιστεί το εμπορικά διαθέσιμο λογισμικό DESS. Το εργαλείο λαμβάνει καθημερινές ενημερώσεις και διατηρεί ένα φόρουμ με διάφορα θέματα. Επιπλέον, κάνει το διαχωρισμό μεταξύ εγχειριδίων "χρήσης" και "προγραμματισμού", παρέχοντας διαφορετικά έγγραφα για κάθε περίπτωση, δείχνοντας έτσι μια εξέταση διαφορετικών επιπέδων δυνητικών χρηστών. Ένα άλλο πλεονέκτημα είναι ότι το JaamSim ως λογισμικό μπορεί να ληφθεί ως αυτόνομο εκτελέσιμο αρχείο (περίπου 6-7 MB για διαφορετικές διανομές που σχετίζονται με τα Windows, το Mac OS X και το Linux). Το γεγονός ότι ένας μη ειδικός χρήστης μπορεί απλά να κατεβάσει και να δοκιμάσει το λογισμικό μέσα σε λίγα λεπτά είναι κάτι που δεν συναντάται συχνά σε εργαλεία ανοιχτού κώδικα και ειδικότερα στον τομέα DESS.

Το εργαλείο έχει μια γενετική αρχιτεκτονική, επιτρέποντας στους προγραμματιστές να προσθέσουν νέα αντικείμενα ("παλέτες" στην ορολογία του JaamSim), ενώ η εταιρεία προσφέρει βίντεο επίδειξης της χρήσης του για την προσομοίωση αλυσίδων εφοδιασμού. Είναι το μόνο εργαλείο που βρήκαμε να είναι σαφώς καθοδηγούμενο από τη βιομηχανία (δηλαδή το ανέπτυξε και το εκμεταλλεύεται μια εμπορική εταιρεία συμβούλων) και αυτό μπορεί να έχει οδηγήσει σε πιο συνεπή κίνητρα και χρηματοδότηση. Το JaamSim έχει άδεια χρήσης GPL, επομένως ένα έργο που βασίζεται σε αυτό είναι υποχρεωμένο να χρησιμοποιεί την ίδια άδεια.

Η τελευταία έκδοση του JaamSim είναι η 2021-02.

Το **JAPROSIM** είναι επίσης ένα πρόσφατο εργαλείο. Αναπτύχθηκε και συντηρείται από ερευνητικά ινστιτούτα στην Αλγερία. Το JAPROSIM υλοποιήθηκε στην Java και προσφέρει επίσης ένα GUI που βασίζεται στη βιβλιοθήκη Java Swing. Η διεπαφή του αποτελείται από ένα πλαίσιο πειραματισμού όπου ο χρήστης μπορεί να παραμετροποιήσει το μοντέλο που έχει προγραμματίσει και την εκτέλεση της προσομοίωσης, και τα παράθυρα όπου τα αποτελέσματα εμφανίζονται σε γραφήματα. Τα ίχνη της προσομοίωσης είναι επίσης

προσβάσιμα. Το εργαλείο μπορεί επίσης να χρησιμοποιηθεί για το συνδυασμό DESS και CS[4]. Η άδεια χρήσης του JAPROSIM είναι LGPLv3. Τα μειονεκτήματα του JAPROSIM είναι ότι δεν μπορεί να βρεθεί λεπτομερής τεκμηρίωση και δεν υπάρχουν κανάλια επικοινωνίας μεταξύ προγραμματιστών ή χρηστών.

Η τελευταία ενημέρωση του JAPROSIM έγινε στις 10/03/2014.

Το **DESMO-J** είναι ένα εργαλείο ανοιχτού κώδικα που υποστηρίζει το είδος μοντελοποίησης που είναι προσανατολισμένη στη διαδικασία αλλά και το είδος μοντελοποίησης που είναι προσανατολισμένη στα συμβάντα. Αναπτύχθηκε από το τμήμα πληροφορικής του Πανεπιστημίου του Αμβούργου. Προσφέρει έτοιμες προς χρήση κλάσεις για κοινές συνιστώσες μοντέλου και μια διεπαφή για το χρήστη που διατίθεται σε 2-D και 3-D εκδόσεις. Όσον αφορά την τεκμηρίωση, παρέχεται διαδικτυακός οδηγός και τεκμηρίωση των API. Το DESMO-J διαθέτει άδεια επί τη βάσει της άδειας Apache έκδοση 2.0. Από το αρχείο καταγραφής ιστορικού, μπορούμε να δούμε ότι υπάρχουν ενημερώσεις με σχεδόν μηνιαία συχνότητα, που δείχνουν μια υπάρχουσα αλλά όχι πολύ ενεργή πρόοδο. Το εργαλείο έχει χρησιμοποιηθεί από ερευνητές και στον τομέα της παραγωγής. Ένα μειονέκτημα είναι ότι το εργαλείο φαίνεται να μην έχει ενεργή κοινότητα, καθώς το φόρουμ DESMO-J στο SourceFourge περιέχει μόνο δύο θέματα και δεν υπάρχουν νέες καταχωρήσεις από το 2011. Το ίδιο ισχύει για τη λίστα αλληλογραφίας που υπάρχει αλλά είναι κενή.

Η τελευταία έκδοση του DESMO-J είναι η 2.5.1e.

Σύμφωνα με τον ιστότοπό του (https://facsim.org/), ο στόχος του εργαλείου **Facsimile** είναι "να αναπτύξει και να διατηρήσει μια υψηλής ποιότητας 3-D βιβλιοθήκη προσομοίωσης διακριτών γεγονότων που μπορεί να χρησιμοποιηθεί για έργα βιομηχανικής προσομοίωσης σε τεχνολογικό ή/και κατασκευαστικό περιβάλλον". Το Facsimile χρησιμοποιεί τη Scala ως βάση του που είναι μια γλώσσα δέσμης ενεργειών τύπου Java. Κυκλοφορεί με την άδεια LGPLv3. Επίσης υπάρχει μια ενότητα τεκμηρίωσης στον ιστότοπο, αλλά δε διαθέτει παραδείγματα για να δείξει τη χρήση του εργαλείου. Ο ιστότοπος του εργαλείου προσφέρει ένα φόρουμ, αλλά δεν περιέχει μηνύματα. Τέλος, δε βρέθηκαν επιστημονικές δημοσιεύσεις που να χρησιμοποιούν το εργαλείο.

4 Η **συνεχής προσομοίωση** (Continuous Simulation) ασχολείται με τη μοντελοποίηση φυσικών συμβάντων (διεργασίες, συμπεριφορές, συνθήκες) που μπορούν να περιγραφούν από κάποιο σύνολο συνεχώς μεταβαλλόμενων εξαρτημένων μεταβλητών. Αυτές με τη σειρά τους ενσωματώνονται σε διαφορικές εξισώσεις ή εξισώσεις διαφοράς που περιγράφουν τη φυσική διεργασία.

Το **SharpSim** είναι μια DESS βιβλιοθήκη που αναπτύχθηκε στη C#. Κάποιος μπορεί να δημιουργήσει δικά του μοντέλα προσομοίωσης με το SharpSim, γράφοντας κώδικα στη C#. Το SharpSim υλοποιεί τη worldview συμβάντων DESS και διαθέτει άδεια βάσει του GPLv2. Χρησιμοποιήθηκε κυρίως από ερευνητές, οι οποίοι είναι και οι συγγραφείς του εργαλείου, για την προσομοίωση αποθηκών μονάδων φορτίου. Το SharpSim φιλοξενείται στο CodePlex που ήταν ο δωρεάν ιστότοπος φιλοξενίας έργων ανοιχτού κώδικα της Microsoft, ο οποίος διήρκεσε από το 2006 έως το 2017. Ο ιστότοπος βρίσκεται σε λειτουργία αρχειοθέτησης τα τελευταία 3 χρόνια και μετά την 1η Ιουλίου 2021 θα κλείσει.

Το **URURAU** έχει χρησιμοποιηθεί από ερευνητές, οι οποίοι είναι επίσης συγγραφείς του εργαλείου, για την αξιολόγηση των εκπομπών CO που παράγονται από ένα σύστημα logistics. Οι ερευνητές συγκρίνουν επίσης τα αποτελέσματά τους με αυτά ενός μοντέλου του ίδιου συστήματος που κατασκευάστηκε στο Arena ένα εμπορικά διαθέσιμο λογισμικό για DESS. Σπάνια βρίσκουμε προγραμματιστές ενός εργαλείου DESS ανοιχτού κώδικα που παρουσιάζουν σύγκριση των αποτελεσμάτων των μοντέλων τους με αντίστοιχα από εμπορικά διαθέσιμο λογισμικό. Το URURAU παρέχει ένα γραφικό περιβάλλον, όπου ο χρήστης μπορεί να κατασκευάσει ένα μοντέλο γραφικά. Διαπιστώσαμε ότι το εργαλείο δέχονταν σχόλια με μηνιαία συχνότητα και υπάρχουν πέντε βίντεο, ένα για την εγκατάσταση του λογισμικού, δύο που παρέχουν μια επισκόπηση των ενοτήτων του URURAU και δύο που παρουσιάζουν παραδείγματα μοντέλων.

Το **PowerDEVS** είναι ένα εργαλείο που υλοποιεί τον φορμαλισμό DEVS. Διανέμεται με δύο άδειες, δηλαδή AFL και GPLv2. Το εργαλείο συνεχίζει να είναι ενεργό, αλλά στον ιστότοπό του δεν υπάρχει τεκμηρίωση για τρίτους χρήστες.

# 3. 3. Το λογισμικό προσομοίωσης ανοιχτού κώδικα Jaamsim

## 3.1 Εισαγωγή

Το JaamSim (King και Harrison 2013) είναι ένα δωρεάν πακέτο προσομοίωσης ανοιχτού κώδικα που αναπτύχθηκε από την εταιρεία μηχανικών Ausenco. Το JaamSim απευθύνεται σε ένα ευρύ κοινό επαγγελματιών, ερευνητών και φοιτητών και περιλαμβάνει μια σύγχρονη GUI που είναι συγκρίσιμη με εκείνη που παρέχεται από εμπορικά διαθέσιμο λογισμικό προσομοίωσης. Το λογισμικό είναι γραμμένο στη γλώσσα προγραμματισμού Java και μπορεί να τηλεφορτωθεί από τον Github[5]. Το εκτελέσιμο αρχείο, το εγχειρίδιο χρήστη, το εγχειρίδιο προγραμματισμού και τα παραδείγματα είναι διαθέσιμα από τους συντάκτες κατόπιν αιτήματος.

Το βασικό χαρακτηριστικό, που κάνει το JaamSim διαφορετικό από το εμπορικά διαθέσιμο λογισμικό προσομοίωσης, είναι ότι επιτρέπει στους χρήστες να δημιουργούν τις δικές τους παλέτες αντικειμένων υψηλού επιπέδου για νέες εφαρμογές. Τα "υψηλού επιπέδου" αντικείμενα είναι εκείνα που προσομοιώνουν τα σημαντικά αντικείμενα για μια δεδομένη κλάση μοντέλου. Για έναν προσομοιωτή κυκλοφορίας, τα αντικείμενα υψηλού επιπέδου θα ήταν οι διάφοροι τύποι οχημάτων, οδών και διασταυρώσεων. Αντίθετα, τα αντικείμενα "χαμηλού επιπέδου" είναι πιο θεμελιώδη και έχουν ευρύτερη εφαρμογή, για παράδειγμα ουρές και εξυπηρετητές. Τα σύνθετα μοντέλα μπορούν να κατασκευαστούν πολύ πιο εύκολα από αντικείμενα υψηλού επιπέδου παρά από αντικείμενα χαμηλού επιπέδου.

Νέα αντικείμενα που έχουν προγραμματιστεί από τον χρήστη έχουν αυτόματα γραφικά 3D, μπορούν να μεταφερθούν και αποτεθούν, να έχουν εισόδους επεξεργάσιμες μέσω επεξεργαστή εισόδων και οι έξοδοί τους να εμφανίζονται στον προβολέα εξόδου. Προγραμματίστηκε σε στάνταρ Java χρησιμοποιώντας σύγχρονα εργαλεία ανάπτυξης όπως το Eclipse. Σε αντίθεση με το εμπορικά διαθέσιμο λογισμικό προσομοίωσης, το JaamSim δεν περιορίζεται από τις ιδιοσυγκρασίες και το περιορισμένο σύνολο δυνατοτήτων μιας ιδιόκτητης γλώσσας προγραμματισμού ή γλώσσας δέσμης ενεργειών.

5 Το git είναι ένα σύστημα διαχείρισης εκδόσεων αρχείων. Το GitHub είναι ένας εξυπηρετητής που φιλοξενεί αποθετήρια και μια εύχρηστη διεπαφή ιστού. Υποστηρίζει όλες τις εντολές του git και παρέχει μερικές επιπλέον δυνατότητες.

Το JaamSim δεν είναι το πρώτο προσφερόμενο λογισμικό προσομοίωσης ανοιχτού κώδικα στην Java. Προηγούμενες μηχανές προσομοίωσης ανοιχτού κώδικα στην Java περιλαμβάνουν τις: DEUS, DSOL, JavaSim, JiST, Jsim, JSL, SimJava, Simkit, SSJ και Tortuga. Αν και αυτά τα πακέτα προσφέρουν πολλά χρήσιμα χαρακτηριστικά που τα καθιστούν καλές επιλογές για ορισμένες εφαρμογές, το καθένα παρέχει μόνο μια βιβλιοθήκη εργαλείων προσομοίωσης και ο χρήστης επαφίεται να προετοιμάσει το μοντέλο του από Java κώδικα. Χωρίς GUI, αυτά τα πακέτα απευθύνονται μόνο σε άτομα που διαθέτουν χρόνο και έχουν δεξιότητες προγραμματισμού απαραίτητες για την κατασκευή μοντέλων με αυτόν τον τρόπο.

Εξαίρεση στα παραπάνω αποτελεί το Simkit, το οποίο παρέχει τον οπτικό δημιουργό γράφων συμβάντων VisSim. Ωστόσο αυτή η GUI περιορίζεται στην κατασκευή μοντέλων από στοιχεία γραφικών συμβάντων χαμηλού επιπέδου, που είναι μόνο ένα μέρος της λειτουργικότητας που απαιτείται για το λογισμικό ώστε να έχει ευρεία απήχηση. Επίσης από τα παραπάνω μόνο τα Jsim, DSOL και Tortuga έχουν ακόμα ενεργή homepage.

Το JaamSim περιλαμβάνει μια σύγχρονη GUI η οποία είναι συγκρίσιμη με το εμπορικά διαθέσιμο λογισμικό προσομοίωσης. Το JaamSim είναι μια εξαιρετικά ελαφριά εφαρμογή που αποτελείται από ένα εκτελέσιμο αρχείο μεγέθους 11Mb, που μπορεί να αντιγραφεί απευθείας στον υπολογιστή του χρήστη. Δεν απαιτείται ειδικό πρόγραμμα εγκατάστασης. Το JaamSim εκτελείται στους περισσότερους σύγχρονους υπολογιστές που υποστηρίζουν τα OpenGL γραφικά, έκδοση 3.0 ή μεταγενέστερη, συμπεριλαμβανομένων των φορητών υπολογιστών με επεξεργαστές Intel Core i5 και i7 series που βασίζονται σε ενσωματωμένα γραφικά ("Sandybridge" δεύτερης γενιάς και νεότερες εκδόσεις).

## 3.2 Περιγραφή της γραφικής διεπαφής χρήστη

Ακολουθούν τα εργαλεία που παρέχει η GUI του JaamSim και είναι απαραίτητα για την κατασκευή μοντέλων:

- **Πίνακας Ελέγχου** - η κύρια διεπαφή του JaamSim που ελέγχει την εκτέλεση των μοντέλων και παρέχει πρόσβαση στις άλλες συνιστώσες της GUI.
- **Προβολές** - ένα ή περισσότερα παράθυρα που δείχνουν τρισδιάστατες προβολές του μοντέλου.
- **Δημιουργός μοντέλων** - εργαλείο για μεταφορά και απόθεση των συνιστωσών του μοντέλου.
- **Επιλογέας αντικειμένων** - παρέχει πρόσβαση σε κάθε αντικείμενο του μοντέλου.

- **Επεξεργαστής εισόδου** - εμφανίζει τις εισόδους του επιλεγμένου αντικείμενου και επιτρέπει την επεξεργασία τους.
- **Προβολέας εξόδου** - εμφανίζει την τρέχουσα τιμή για τις εξόδους του επιλεγμένου αντικείμενου.
- **Προβολέας ιδιοτήτων** - εργαλείο εντοπισμού σφαλμάτων που εμφανίζει όλες τις εσωτερικές ιδιότητες του επιλεγμένου αντικείμενου.

Παρέχονται οι ακόλουθες παλέτες αντικειμένων εντός του Δημιουργού Μοντέλων (Model Builder):

- **Αντικείμενα γραφικών** - παρέχει 2D και 3D συνιστώσες, όπως γραφήματα, κείμενο, βέλη, κ.α., που χρειάζονται για την οπτικοποίηση και παρακολούθηση μιας προσομοίωσης.
- **Κατανομές πιθανοτήτων** - είναι τυπικές κατανομές πιθανοτήτων, όπως η ομοιόμορφη κατανομή, η εκθετική κατανομή, η κανονική κατανομή, και κατανομές καθορισμένες από το χρήστη.
- **Βασικά αντικείμενα** - εισαγωγή αριθμητικών τιμών άμεσα στην οθόνη προσομοίωσης μοντέλου, χρονοσειρές, εκφράσεις οντότητας, κ.α.

- **Αντικείμενα πόρων** - είναι αντικείμενα που μπορούν να δεσμευτούν και να αποδεσμευτούν από άλλα αντικείμενα σε ένα μοντέλο, όπως ο επεξεργαστής οντότητας. Όταν ένας πόρος δεσμευτεί από ένα αντικείμενο, δεν μπορεί να δεσμευθεί από άλλο αντικείμενο έως ότου αποδεσμευθεί. Τα αντικείμενα πόρων δεν εκτελούν δική τους εργασία.

- **Διεργασία ροής** - παρέχει τα αντικείμενα που είναι απαραίτητα για τη δημιουργία μοντέλων ροής διεργασίας. Σ' αυτά τα μοντέλα, μια οντότητα προσομοίωσης μεταφέρεται από ένα αντικείμενο σε άλλο ακολουθώντας ένα διάγραμμα ροής διεργασίας. Τα ακόλουθα αντικείμενα είναι μερικά από τα είκοσι επτά αντικείμενα της παλέτας: οντότητα προσομοίωσης, ουρά, εξυπηρετητής, διακομιστής και διακλάδωση.

- **Αντικείμενα υπολογισμού** - περιέχονται αντικείμενα για την κατασκευή μοντέλων συνεχούς τύπου και μικτών μοντέλων διακριτών/συνεχών συμβάντων. Ένα συνεχές μοντέλο αλλάζει κατάσταση συνεχώς με την πάροδο του χρόνου. Το JaamSim είναι σε

θέση να μοντελοποιήσει αυτό το είδος συμπεριφοράς πολύ αποτελεσματικά, χρησιμοποιώντας εξόδους, χαρακτηριστικά και εκφράσεις.

- **Ρευστά αντικείμενα** - περιέχονται αντικείμενα για την κατασκευή δυναμικών μοντέλων υδραυλικής ροής που περιλαμβάνουν δεξαμενές, σωλήνες, αντλίες, κ.α. Ο ρυθμός ροής στο σύστημα υπολογίζεται λύνοντας την ασταθή εξίσωση Bernoulli.
- **Αντικείμενα υπομοντέλου** - τα υπομοντέλα παρέχουν έναν τρόπο ομαδοποίησης των αντικειμένων ενός μοντέλου σε λειτουργικές μονάδες και τη δημιουργία μιας ιεραρχικής δομής.

## 3.3 Είσοδοι μοντέλου

Ο ευκολότερος και ταχύτερος τρόπος για να δημιουργήσουμε ένα απλό μοντέλο είναι να μεταφέρουμε και να αποθέσουμε μεμονωμένα αντικείμενα και να χρησιμοποιήσετε τον Επεξεργαστή Εισόδου (Input Editor) για να ορίσουμε τις παραμέτρους τους. Ωστόσο αυτή η μέθοδος καθίσταται λιγότερο ελκυστική για πιο περίπλοκα μοντέλα, όπου τα γραφικά γίνονται μπερδεμένα και οι είσοδοι γίνονται πάρα πολλές. Για ένα πολύ μεγάλο μοντέλο που περιέχει εκατοντάδες αντικείμενα, είναι συνήθως απαραίτητο να δημιουργήσουμε το μοντέλο μέσω ενός αρχείου εισόδου.

Το JaamSim επιτρέπει την κατασκευή μοντέλων είτε μέσω της GUI είτε ενός αρχείου εισόδου. Το αρχείο εισόδου χρησιμοποιεί τη δομή "αντικείμενο" - "χαρακτηριστικό" - "τιμή". Για παράδειγμα, οι ακόλουθες γραμμές εισόδου:

```
Define Ship { LargeShip }

LargeShip Length { 300 m }
```

ορίζουν το αντικείμενο "LargeShip" (ένα στιγμιότυπο του αντικειμένου "Ship") και εκχωρούν την τιμή 300m στην είσοδο "Length". Σημειώνουμε ότι σε όλες τις εισόδους μοντέλου εκχωρούνται μονάδες, όπου απαιτείται, και μετατροπές μονάδων πραγματοποιούνται αυτόματα από το πρόγραμμα.

Οι εσωτερικοί υπολογισμοί γίνονται σε μονάδες SI[6] από την JaamSim. Ο επεξεργαστής εισόδου χρησιμοποιεί την ίδια δομή "αντικείμενο"-"χαρακτηριστικό"-"τιμή" με το αρχείο εισόδου.

6 Συντομογραφία του γαλλικού Système International.

## 3.4 Τα αντικείμενα Process Flow

Η παλέτα Process Flow περιέχει όλα τα αντικείμενα που απαιτούνται για τη δημιουργία μοντέλων τύπου διαδικασίας ροής. Αυτά τα μοντέλα χαρακτηρίζονται από μια παθητική οντότητα που περνά από το ένα αντικείμενο στο άλλο ακολουθώντας ένα διάγραμμα ροής διεργασίας. Αυτοί οι τύποι μοντέλων χρησιμοποιούνται συχνά για την προσομοίωση μιας διαδικασίας παραγωγής όπου οι οντότητες αναπαριστάνουν μέρη που μετακινούνται μεταξύ των σταθμών επεξεργασίας. Τα ακόλουθα αντικείμενα παρέχονται στην παλέτα διαδικασίας ροής.

### 3.4.1SimEntity

Το αντικείμενο SimEntity (με εικόνα  ) είναι η βασική οντότητα που περνά μέσα από ένα μοντέλο τύπου ροής διεργασίας. Το κύριο χαρακτηριστικό της SimEntity είναι ότι η κατάστασή της μπορεί να εκχωρηθεί σε διάφορα στάδια της διαδικασίας ροής. Τα αντικείμενα διαδικασίας ροής που μπορούν να δεχτούν μια SimEntity, όπως ο Server και η Queue, μπορούν να εκχωρήσουν μια κατάσταση στη ληφθείσα SimEntity χρησιμοποιώντας το χαρακτηριστικό StateAssignment. Για παράδειγμα, αν η είσοδος της StateAssignment μιας ουράς έχει οριστεί ως "Waiting", τότε η κατάσταση για κάθε SimEntity ληφθείσα από την ουρά θα οριστεί σε "Waiting" όταν φτάνει στην ουρά. Η SimEntity θα παραμείνει σε αυτήν την κατάσταση έως ότου οριστεί σε μια νέα κατάσταση από ένα επόμενο αντικείμενο. Η πρόσβαση στο χρόνο παραμονής μιας SimEntity σε κάθε κατάσταση είναι δυνατή χρησιμοποιώντας την έξοδο StateTimes. Για παράδειγμα, ο συνολικός χρόνος παραμονής της SimEntity στην κατάσταση "Waiting" δίνεται από τη `StateTimes("Waiting")`. Σε αντίθεση με άλλα αντικείμενα, τα δεδομένα της StateTimes για μια SimEntity δε διαγράφονται στο τέλος της περιόδου αρχικοποίησης.

### 3.4.2 EntityGenerator

Το αντικείμενο EntityGenerator (με εικόνα  ) δημιουργεί μια σειρά οντοτήτων που μεταβιβάζονται στο επόμενο αντικείμενο σε μια διεργασία. Η είσοδος PrototypeEntity προσδιορίζει την οντότητα από την οποία θα δημιουργηθεί το αντίγραφο. Αυτή η οντότητα μπορεί να είναι είτε ένα συγκεκριμένο αντικείμενο είτε μια έκφραση που επιστρέφει ένα αντικείμενο. Τα αντίγραφα διατηρούν τόσο την εικόνα του πρωτότυπου όσο και τις τιμές όλων των εισόδων του.

### 3.4.3 EntitySink

Το αντικείμενο EntitySink (με εικόνα  ) καταστρέφει τις εισερχόμενες οντότητες. Συνήθως χρησιμοποιείται ως το τελευταίο αντικείμενο του μοντέλου.

### 3.4.4 Server

Το αντικείμενο Server (με εικόνα  ) επεξεργάζεται μια εισερχόμενη οντότητα και στη συνέχεια τη μεταβιβάζει στο επόμενο αντικείμενο. Οι οντότητες που περιμένουν να υποβληθούν σε επεξεργασία κρατούνται από ένα αντικείμενο ουράς που προσδιορίζεται από το χαρακτηριστικό WaitQueue. Όλες οι οντότητες που λαμβάνονται από τον εξυπηρετητή περνούν πρώτα από αυτό την ουρά, ακόμη και αν ο εξυπηρετητής είναι αδρανής.

Κάθε φορά που μια οντότητα προστίθεται σε μια ουρά, ειδοποιούνται όλοι οι εξυπηρετητές και άλλοι τύποι αντικειμένων που όρισαν αυτήν την ουρά ως WaitQueue. Το πρώτο διαθέσιμο αντικείμενο θα αφαιρέσει την οντότητα από την ουρά για επεξεργασία.

Η ταχύτητα επεξεργασίας των οντοτήτων καθορίζεται από το χαρακτηριστικό ServiceTime. Αυτή η είσοδος έχει τις μονάδες χρόνου και μπορεί να είναι μια σταθερή τιμή, ένα αντικείμενο που επιστρέφει έναν αριθμό με μονάδες χρόνου (π.χ. TimeSeries ή Probability Distribution) ή μια παράσταση που επιστρέφει έναν τέτοιο αριθμό.

Ο εξυπηρετητής μπορεί να σταματήσει, υπό διάφορες συνθήκες, χρησιμοποιώντας το χαρακτηριστικό OperatingThresholdList, το οποίο καθορίζει μια λίστα αντικειμένων κατωφλίου όπως SignalThreshold, TimeSeriesThreshold ή ExpressionThreshold. Όλα τα καθορισμένα αντικείμενα κατωφλίου πρέπει να είναι ανοιχτά για να λειτουργήσει ο εξυπηρετητής. Ωστόσο, αν ένα κατώφλι κλείσει ενώ ο εξυπηρετητής επεξεργάζεται μια οντότητα, θα ολοκληρώσει τις εργασίες του με αυτήν την οντότητα πριν από τη διακοπή της περαιτέρω λειτουργίας. Παρόλο που ένας εξυπηρετητής ειδοποιείται αυτόματα όταν μια οντότητα φτάσει στην ουρά του, δεν ειδοποιείται αυτόματα όταν αλλάζει η είσοδος Match. Το χαρακτηριστικό WatchList παρέχει έναν τρόπο ειδοποίησης του εξυπηρετητή για αυτού του είδους τις αλλαγές.

### 3.4.5 Queue

Ένα αντικείμενο Queue (με εικόνα  ) ορίζει μια τοποθεσία όπου οντότητες προσομοίωσης περιμένουν την επεξεργασία τους από άλλες οντότητες.

Σε αντίθεση με πολλά άλλα αντικείμενα αυτής της παλέτας, μια οντότητα που λαμβάνεται από μια ουρά δεν μεταφέρεται αυτόματα στο επόμενο αντικείμενο. Πρέπει να περιμένει στην ουρά μέχρι να αφαιρεθεί από κάποιο άλλο αντικείμενο. Οι ουρές χρησιμοποιούνται με αυτόν τον τρόπο από τα αντικείμενα Server, Seize, EntityGate, Assemble, Combine, Pack, Unpack, AddTo και RemoveFrom. Κάθε φορά που μια οντότητα προστίθεται σε μια ουρά, όλα τα αντικείμενα που χρησιμοποιούν αυτήν την ουρά ειδοποιούνται ότι μια νέα οντότητα είναι διαθέσιμη. Το πρώτο διαθέσιμο αντικείμενο θα αφαιρέσει την οντότητα από την ουρά και θα ξεκινήσει την επεξεργασία της.

Οι οντότητες, που βρίσκονται σε ουρά, μπορούν να διαταχτούν με μια (προαιρετική) προτεραιότητα που καθορίζεται από το χαρακτηριστικό Priority. Στις περισσότερες περιπτώσεις, η προτεραιότητα καθορίζεται από μια έκφραση που υπολογίζεται όταν η οντότητα φτάνει για πρώτη φορά στην ουρά. Η προτεραιότητα είναι ακέραια τιμή και οι δεκαδικές τιμές περικόπτονται.

Οι οντότητες με τις ίδιες τιμές προτεραιότητας μπορούν να υποβληθούν, εξ ορισμού, σε first-in-first-out (FIFO) επεξεργασία ή σε last-in-first-out (LIFO) επεξεργασία.

Τέλος, μια οντότητας που βρίσκεται σε ουρά μπορεί να εξοπλιστεί με μια προαιρετική ταυτότητα χρησιμοποιώντας το χαρακτηριστικό Match. Τα αντικείμενα που χρησιμοποιούν ουρές, όπως ένας εξυπηρετητής, μπορούν να ζητήσουν από την ουρά τους να παράσχει στην πρώτη οντότητα μια καθορισμένη τιμή για το χαρακτηριστικό Match. Όπως με το χαρακτηριστικό Priority, στις περισσότερες περιπτώσεις η τιμή Match καθορίζεται από μια έκφραση που υπολογίζεται όταν η οντότητα φτάνει για πρώτη φορά στην ουρά. Η μεταβλητή Match αποθηκεύεται ως συμβολοσειρά, αλλά θα δεχτεί και έναν ακέραιο χωρίς διάσταση ή μια οντότητα που θα μετατραπεί σε συμβολοσειρά. Μια δεκαδική τιμή περικόπτεται.

### 3.4.6 EntityConveyor

Το αντικείμενο EntityConveyor (με εικόνα ) μετακινεί μια εισερχόμενη οντότητα κατά μήκος μιας διαδρομής με σταθερή ταχύτητα και, στη συνέχεια, τη μεταβιβάζει στο επόμενο αντικείμενο.

Ο χρόνος της διαδρομής για το EntityConveyor καθορίζεται από το χαρακτηριστικό TravelTime. Εάν καθοριστεί ένας μεταβλητός χρόνος διαδρομής μέσω μιας έκφρασης, η ταχύτητα του μεταφορέα ενημερώνεται κάθε φορά που μια οντότητα προστίθεται στον μεταφορέα ή μια οντότητα φτάνει στο τέλος του μεταφορέα.

### 3.4.7 Seize

Το αντικείμενο Seize (με εικόνα  ) χορηγεί μία ή περισσότερες μονάδες ενός καθορισμένου συνόλου πόρων, όταν λάβει μια εισερχόμενη οντότητα. Αν κάποιος από τους πόρους δε διαθέτει επαρκείς μονάδες, η ληφθείσα οντότητα κατευθύνεται προς ένα αντικείμενο ουρά που προσδιορίζεται από το χαρακτηριστικό WaitQueue. Όλες οι οντότητες που λαμβάνονται από το αντικείμενο Seize περνούν πρώτα από αυτό το αντικείμενο ουρά, ακόμη και αν οι πόροι διαθέτουν επαρκείς μονάδες.

Οι οντότητες μπορούν να σταλούν απευθείας στην ουρά που καθορίζεται από το χαρακτηριστικό WaitQueue. Κάθε φορά που μια οντότητα προστίθεται στην ουρά, ειδοποιούνται όλα τα αντικείμενα Seize που ορίζουν αυτήν την ουρά ως WaitQueue. Το πρώτο μπλοκ Seize που διαθέτει επαρκείς μονάδες πόρων, θα αφαιρέσει την οντότητα από την ουρά για επεξεργασία.

Οι οντότητες που περιμένουν τους ίδιους πόρους σε περισσότερα από ένα αντικείμενα Seize υποβάλλονται σε επεξεργασία με τη σειρά προτεραιότητας που τους έχει εκχωρηθεί από την είσοδο του χαρακτηριστικού Priority της ουράς τους. Αν οι οντότητες σε δύο ή περισσότερες ουρές έχουν την ίδια τιμή προτεραιότητας, τότε επιλέγεται αυτή με τη μεγαλύτερη αναμονή.

Όλοι οι καθορισμένοι πόροι για την επιλεγμένη οντότητα πρέπει να είναι διαθέσιμοι πριν καταληφθούν. Όταν όλοι είναι διαθέσιμοι, τότε καταλαμβάνονται ταυτόχρονα.

### 3.4.8 Release

Το αντικείμενο Release (με εικόνα  ) καταργεί μία ή περισσότερες μονάδες ενός καθορισμένου συνόλου πόρων αφού λάβει μια εισερχόμενη οντότητα. Όλοι οι πόροι απελευθερώνονται ταυτόχρονα.

Κατά την απελευθέρωση των πόρων, οι οντότητες που περιμένουν αυτούς τους πόρους υποβάλλονται σε επεξεργασία με τη σειρά προτεραιότητας που τους έχει δοθεί από την είσοδο του χαρακτηριστικού Priority της ουράς τους. Αν οι οντότητες σε δύο ή περισσότερες ουρές έχουν την ίδια τιμή προτεραιότητας, τότε επιλέγεται αυτή με τη μεγαλύτερη αναμονή.

### 3.4.9 Assign

Το αντικείμενο Assign (με εικόνα  ) εκτελεί μία ή περισσότερες εκχωρήσεις σε ιδιότητες, κάθε φορά που λαμβάνει μια εισερχόμενη οντότητα. Μόλις οι εκχωρήσεις

εκτελεστούν, η ληφθείσα οντότητα μεταβιβάζεται στο επόμενο αντικείμενο χωρίς καθυστέρηση.

Το αντικείμενο Assign είναι το μόνο μέρος όπου μπορεί να τροποποιηθεί η τιμή μιας ιδιότητας. Το αντικείμενο Assign μπορεί να τροποποιήσει την τιμή οποιασδήποτε ιδιότητας στο μοντέλο: τις δικές του ιδιότητες, τις ιδιότητες της ληφθείσας οντότητας ή άλλες ιδιότητες του αντικειμένου.

Οι εκχωρήσεις που θα εκτελεστούν καθορίζονται από την είσοδο του χαρακτηριστικού AttributeAssignmentList. Κάθε εκχώρηση έχει την ακόλουθη μορφή:

```
{ <left-hand side expression> = <right-hand side expression> }
```

Η δεξιά πλευρά κάθε εκχώρησης είναι μια έκφραση που πρέπει να υπολογιστεί. Η αριστερή πλευρά είναι μια έκφραση που προσδιορίζει την ιδιότητα της οποίας η τιμή πρόκειται να τροποποιηθεί. Για παράδειγμα, αν το αντικείμενο Logariasmos έχει μια αριθμητική ιδιότητα metritis χωρίς διάσταση, τότε η ακόλουθη είσοδος του χαρακτηριστικού AttributeAssignmentList προκαλεί την αύξηση της ιδιότητας κατά 1:

```
{ 'this.metritis = this.metritis + 1' }
```

Η ιδιότητα obj του αντικειμένου Assign μπορεί να χρησιμοποιηθεί για την προσπέλαση των ιδιοτήτων της ληφθείσας οντότητας. Για παράδειγμα, αν η ληφθείσα οντότητα έχει μια ιδιότητα mhnyma της οποίας η τιμή είναι μια συμβολοσειρά, τότε η ακόλουθη είσοδος του χαρακτηριστικού AttributeAssignmentList έχει ως συνέπεια η ιδιότητα αυτή να λάβει την τιμή `'To dema exei apostalei'`:

```
{ 'this.obj.mhnyma = "To dema exei apostalei"' }
```

Εάν ένα άλλο αντικείμενο, ας πούμε `Aytokinitodromos`, έχει μια ιδιότητα `diadromi` της οποίας η τιμή είναι ένας αριθμός, τότε η ακόλουθη είσοδος του χαρακτηριστικού AttributeAssignmentList προκαλεί την αύξηση αυτής της ιδιότητας κατά 1 χιλιόμετρο:

```
{ '[Aytokinitodromos].diadromi = [Aytokinitodromos].diadromi + 1[km]' }
```

Οι θέσεις καταχωρήσεις σε έναν πίνακα ή map μπορούν επίσης να τροποποιηθούν. Για παράδειγμα, η ακόλουθη είσοδος εκχωρεί την τιμή 1 στη δεύτερη θέση καταχώρισης σε μια ιδιότητα ArrayAttrib που είναι πίνακας και την τιμή 2 στο κλειδί "abc" της ιδιότητας MapAttrib που είναι map:

```
{ 'this.ArrayAttrib(2) = 1' } { 'this.MapAttrib("abc") = 2' }
```

Επιπλέον, ο αριθμοδείκτης ενός πίνακα ή το κλειδί ενός map μιας εκχώρησης, μπορεί να είναι μια υπολογισμένη ποσότητα. Για παράδειγμα, η ακόλουθη είσοδος εκχωρεί την τιμή 1 στον αριθμοδείκτη της ArrayAttrib που δίνεται από την ιδιότητα N της οντότητας που υποβάλλεται σε επεξεργασία:

```
{ 'this.ArrayAttrib(this.obj.N) = 1' }
```

Μια θέση καταχώρισης σε έναν πίνακα δύο αριθμοδεικτών (ένας πίνακας πινάκων) μπορεί να οριστεί καθορίζοντας δύο ξεχωριστούς αριθμοδείκτες. Για παράδειγμα, αν η ιδιότητα MatrixAttrib είναι πίνακας δύο αριθμοδεικτών {{0,0}, {0,0}}, τότε η ακόλουθη είσοδος εκχωρεί την τιμή 1 στη δεύτερη θέση καταχώρησης του πρώτου ένθετου πίνακα:

```
{ 'this.MatrixAttrib(1)(2) = 1' }
```

Ένας απεριόριστος αριθμός εκχωρήσεων μπορεί να εκτελεστεί από ένα αντικείμενο Assign. Οι εκχωρήσεις υποβάλλονται σε επεξεργασία με τη σειρά με την οποία εμφανίζονται στην είσοδο AttributeAssignmentList.

### 3.4.10 Branch

Το αντικείμενο Branch (με εικόνα  ) κατευθύνει μια εισερχόμενη οντότητα σε έναν προορισμό που επιλέγεται από μια λίστα εναλλακτικών προορισμών.

Η τιμή του χαρακτηριστικού Choice καθορίζει τον προορισμό: 1 = πρώτη διακλάδωση, 2 = δεύτερη διακλάδωση, κ.λπ. Η είσοδος του χαρακτηριστικού Choice μπορεί να είναι μια σταθερά, μια DiscreteDistribution, μια TimeSeries ή μια Expression. Η χρήση μιας Expression επιτρέπει την επιλογή να γίνεται με βάση τα χαρακτηριστικά της εισερχόμενης οντότητας.

### 3.4.11 Combine

Το αντικείμενο Combine (με εικόνα  ) παίρνει μια οντότητα από πολλές ουρές και μεταβιβάζει μία οντότητα στο επόμενο αντικείμενο.

Αν η είσοδος του χαρακτηριστικού MatchRequired είναι TRUE, τότε κάθε οντότητα πρέπει να έχει την ίδια τιμή ταιριάσματος που υπολογίζεται από την είσοδο του χαρακτηριστικού Match της ουράς της.

Οι οντότητες συνδυάζονται όταν ο αριθμός που καθορίζεται από την είσοδο του χαρακτηριστικού NumberRequired είναι διαθέσιμος σε κάθε ουρά. Όταν εντοπίζεται

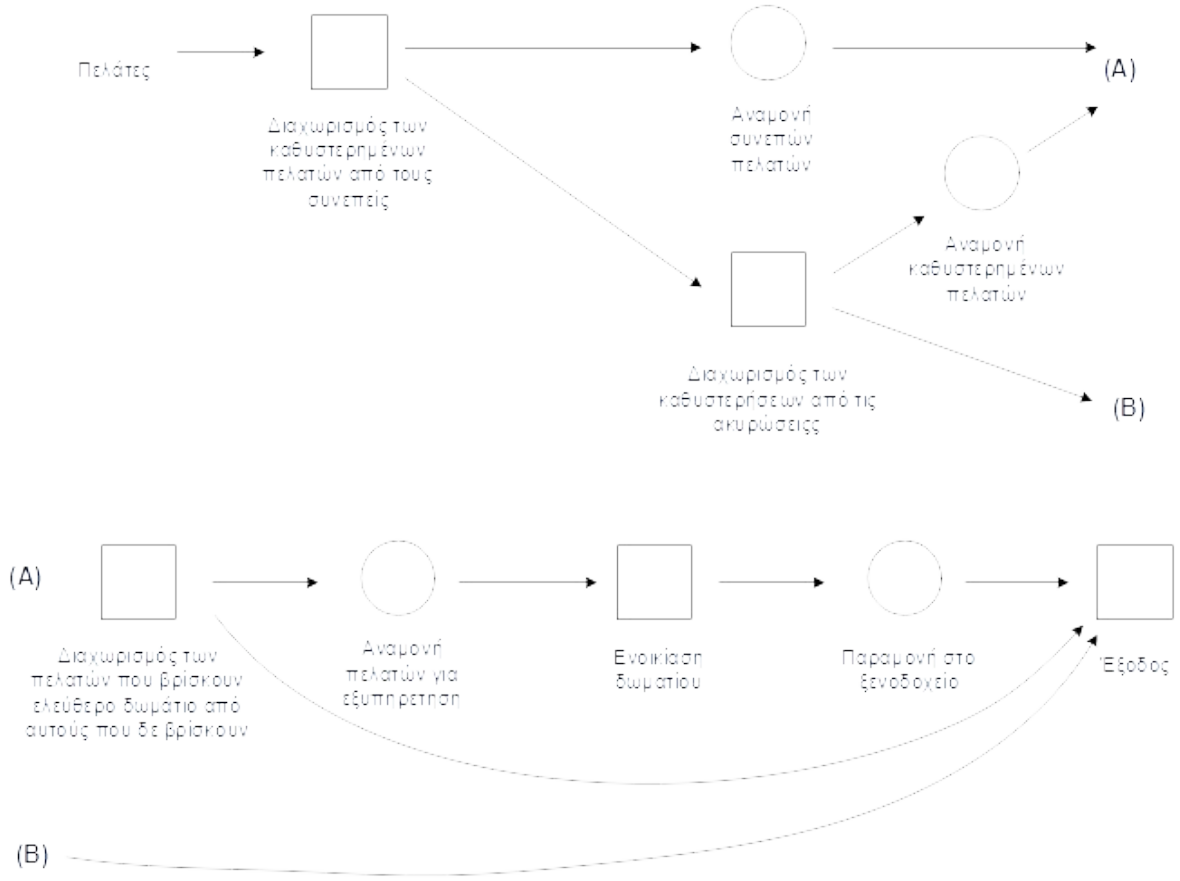
σύμπτωση, η οντότητα στην πρώτη ουρά μεταβιβάζεται στο αντικείμενο που καθορίζεται από το χαρακτηριστικό NextComponent, ενώ οι οντότητες από τις άλλες ουρές καταστρέφονται.

### 3.4.12 EntityLogger

Το αντικείμενο **EntityLogger**  καταγράφει τις εξόδους, τα χαρακτηριστικά και τα δεδομένα κατάστασης για κάθε οντότητα που λαμβάνει. Το αρχείο καταγραφής εξόδου δημιουργείται αυτόματα όταν ξεκινά η εκτέλεση της προσομοίωσης. Το αρχείο εξόδου ονομάζεται <όνομα αρχείου διαμόρφωσης>-<όνομα EntityLogger>**.**log για να διασφαλιστεί ότι είναι μοναδικό για την εκτέλεση της προσομοίωσης. Για παράδειγμα, εάν το αρχείο διαμόρφωσης ονομάζεται "hotel.cfg" και το όνομα του EntityLogger είναι ArExypiretisewn, τότε το όνομα του αρχείου καταγραφής θα είναι "hotel-ArExypiretisewn.log". Ένα προϋπάρχον αρχείο με αυτό το όνομα θα αντικατασταθεί μόλις ξεκινήσει η εκτέλεση της προσομοίωσης.

# 4. 4. Το εννοιολογικό μοντέλο ενός ξενοδοχείου

Σε αυτό το κεφάλαιο θα παρουσιάσουμε το εννοιολογικό μοντέλο ενός ξενοδοχείου και στο επόμενο κεφάλαιο θα παρουσιάσουμε το μοντέλο προσομοίωσης. Η αναπαράσταση του εννοιολογικού μοντέλου θα γίνει με τη χρήση του διαγράμματος ροής διεργασίας. Όπως ήδη είπαμε, ένα διάγραμμα ροής διεργασιών δείχνει κάθε συνιστώσα του συστήματος σε μια ακολουθία. Οι διεργασίες εμφανίζονται ως πλαίσια και οι ουρές ως κύκλοι. Το παρακάτω σχήμα απεικονίζει το διαγράμματος ροής διεργασίας για ένα ξενοδοχείο.




**Σχήμα 4.7 Δ**ιαγράμματος ροής διεργασίας για ένα ξενοδοχείο

# 5. 5. Περιγραφή του μοντέλου προσομοίωσης

## 5.1 Δημιουργία των αντικειμένων του μοντέλου

Μπορούμε να προσθέσουμε αντικείμενα στο μοντέλο μας σύροντάς και αποθέτοντάς τα από το Model Builder στο ViewWindow.

Βασικοί τρόποι ελέγχου στο ViewWindow:

| **Ενέργεια του ποντικιού** | **Αποτέλεσμα** |
|---|---|
| Αριστερό κλικ και σύρσιμο | Μετακινεί το παράθυρο προβολής |
| Αριστερό κλικ | Επιλέγει το σημείο ενδιαφέροντος |
| Control + αριστερό κλικ και σύρσιμο | Μετακινεί το επιλεγμένο αντικείμενο |
| Shift + αριστερό κλικ και σύρσιμο | Μεγέθυνση/σμίκρυνση |

Αρχικά θα προσθέσουμε το αντικείμενο τύπου SimEntity (βρίσκεται στη λίστα Process Flow) στο μοντέλο μας για να αναπαραστήσουμε έναν πελάτη στο σύστημά μας.

Αν η ετικέτα (το όνομα) της νέας SimEntity δεν προβάλλεται, τότε πατάμε το δεξί κουμπί του ποντικιού πάνω στο αντικείμενο και επιλέγουμε "Show Label". Διπλοπατάμε πάνω στην ετικέτα και πληκτρολογούμε το όνομα "Pelatis".

Μπορούμε να αλλάξουμε την εικόνα του αντικειμένου πατώντας το δεξί κουμπί του ποντικιού πάνω στο αντικείμενο και επιλέγοντας "Change Graphics". Στο παράθυρο που εμφανίζεται επιλέγουμε την εικόνα που θέλουμε. Εμείς διαλέξαμε τη μορφή ενός ταξιδιώτη που σέρνει τη βαλίτσα του.

### 5.1.1 Αντικείμενα

Τα επόμενα αντικείμενα που ορίσαμε είναι τα εξής:

**EntityGenerator**, με ετικέτα ErxomosPelatwn για τη δημιουργία μιας ακολουθίας πελατών. Στον Input Editor, δώσαμε τις εξής τιμές στα χαρακτηριστικά:

NextComponent: AssignTimiDiax
InterArrivalTime: GenerationOfPelates
PrototypeEntity: Pelatis

**Assign**, με ετικέτα AssignTimiDiax για την εκχώρηση ιδιότητας της οποίας η τιμή διαχωρίζει τους πελάτες σε συνεπείς και καθυστερημένους. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: Kathysterisi?
AttributeAssignmentList: { 'this.obj.TyposPelatis = [TimiDiaxorismoy].Value' }

**Branch**, με ετικέτα Kathysterisi? για να κατευθύνει έναν εισερχόμενο πελάτη στην ουρά των συνεπών ή στην ουρά των καθυστερημένων. Η επιλογή της ουράς γίνεται με βάση την τιμή της ιδιότητας this.obj.TyposPelatis. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: Metaforeas  Synepeis
Choice: this.obj.TyposPelatis

Αν η τιμή του χαρακτηριστικού Choice είναι 1, τότε προχωράει προς τη συνιστώσα Metaforeas, αν είναι 2, τότε προχωράει προς τη συνιστώσα Synepeis.

**Queue**, με ετικέτα Synepeis για τους πελάτες που φτάνουν στην ώρα τους. Οι τιμές των χαρακτηριστικών είναι:

Priority: 0
MaxPerLine:2

**Assign** με ετικέτα AssignMeresKathisterisis για την εκχώρηση ιδιότητας που η τιμή της διαχωρίζει τους πελάτες ανάλογα με τις ημέρες καθυστέρησης. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: LockDown?
AttributeAssignmentList: { 'this.obj.TyposPelatis = ([MeresKathisterisis].Value > 10[s] ? 2 : 1)' }

Δηλαδή, αν ο πελάτης καθυστερήσει περισσότερες από 10 ημέρες (στο μοντέλο seconds), τότε η ιδιότητα this.obj.TyposPelatis παίρνει την τιμή 2, διαφορετικά παίρνει την τιμή 1.

**Branch**, με ετικέτα LockDown? για να κατευθύνει έναν εισερχόμενο πελάτη στην ουρά καθυστερημένων, αν πρόκειται για λίγες ημέρες καθυστέρησης (this.obj.TyposPelatis=1), για παράδειγμα λόγω κακών καιρικών συνθηκών, ή στην έξοδο από το σύστημα αν, για παράδειγμα, πρόκειται για lock down (this.obj.TyposPelatis=2). Οι τιμές των χαρακτηριστικών είναι:

NextComponent: Kathysterimenoi   Metaforeas1ProsExodo
AttributeAssignmentList: this.obj.TyposPelatis

Αν η τιμή του χαρακτηριστικού AttributeAssignmentList είναι 1, τότε προχωράει προς τη συνιστώσα AttributeAssignmentList, αν είναι 2, τότε προχωράει προς τη συνιστώσα Metaforeas1ProsExodo.

**Queue**, με ετικέτα Kathysterimenoi για τους, λίγων ημερών, καθυστερημένους. Οι τιμές των χαρακτηριστικών είναι:

Priority: 3
MaxPerLine:2

Η προτεραιότητα 3 είναι χαμηλότερη από τις προτεραιότητες 2,1,0, δηλαδή 3 < 2 < 1 < 0.

**Combine**, με ετικέτα Synenwsi για να συνενώσει τους συνεπείς με τους καθυστερημένους. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: AssignArDwmatiwn
ServiceTime: 0.1 sec
WaitQueueList: Kathysterimenoi  Synepeis
NumberRequired: {1.0} {GenSynepwn}
RetainAll: TRUE

Η τιμή TRUE του χαρακτηριστικού RateinAll σημαίνει ότι όλες οι αντίστοιχες οντότητες μεταβιβάζονται στην επόμενη συνιστώσα.

**Assign**, με ετικέτα AssignArDwmatiwn για την εκχώρηση ιδιότητας που η τιμή της διαχωρίζει τους πελάτες σε αυτούς που βρίσκουν δωμάτιο και σε αυτούς που δε βρίσκουν. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: Dwmatio?
AttributeAssignmentList: { 'this.obj.TyposPelatis = ([ArithmosDwmatiwn].AvailableUnits > 0 ? 1 : 2)' }

Δηλαδή, αν υπάρχουν διαθέσιμα δωμάτια, τότε η ιδιότητα this.obj.TyposPelatis παίρνει την τιμή 1, διαφορετικά παίρνει την τιμή 2.

**Branch**, με ετικέτα Dwmatio? για να κατευθύνει τους πελάτες στην ουρά των αναμενόντων δωμάτιο ή στην έξοδο από το σύστημα αν δεν υπάρχει ελεύθερο δωμάτιο. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: MetaforeasEisodoy  Metaforeas2ProsExodo
AttributeAssignmentList: this.obj.TyposPelatis

**Queue**, με ετικέτα AnamenontesDwmatio για τους πελάτες που περιμένουν να εξυπηρετηθούν εφόσον υπάρχει διαθέσιμος receptionist και έχει αδειάσει δωμάτιο. Οι τιμές των χαρακτηριστικών είναι:

Priority: 0
MaxPerLine: 3

**Seize**, με ετικέτα Dektos για να εξυπηρετήσει έναν πελάτη εφόσον υπάρχει διαθέσιμος receptionist και ελεύθερο δωμάτιο. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: EleytherwsiReceptionist

WaitQueue: AnamenontesDwmatio
ResourceList: ArithmosDwmatiwn  ArithmosReceptionists

**Release**, με ετικέτα EleytherwsiReceptionist που απελευθερώνει έναν υπάλληλο. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: Diamoni
ResourceList: ArithmosReceptionists

**Release**, με ετικέτα EleytherwsiDwmatiou που απελευθερώνει ένα δωμάτιο. Οι τιμές των χαρακτηριστικών είναι:

NextComponent: Anaxwrisi
ResourceList: ArithmosDwmatiwn

**EntitySink**, με ετικέτα Anaxwrisi για την έξοδο των οντοτήτων από το σύστημα. Οι οντότητες αυτές καταστρέφονται.

Σε διάφορα σημεία παρεμβάλαμε αντικείμενα τύπου EntityConveyor ώστε να μπορούμε να βλέπουμε τους πελάτες που διατρέχουν αυτήν τη διαδρομή.

Επίσης σε διάφορα αντικείμενα τοποθετήσαμε το αντικείμενο Text, ώστε να βλέπουμε τον αριθμό των πελατών που διέρχονται από το αντικείμενο.

Τα παραπάνω αντικείμενα αναπαριστάνουν τα διαφορετικά τμήματα του συστήματος που συναντά ένας Πελάτης.

### 5.1.2 Κατανομές πιθανοτήτων

Στη συνέχεια, ορίσαμε τα ακόλουθα αντικείμενα κατανομής πιθανότητας:

ExponentialDistribution, με ετικέτα GenerationOfPelates για τη δημιουργία των πελατών του ξενοδοχείου. Οι τιμές που παράγονται χρησιμοποιούνται από το αντικείμενο EntityGenerator με όνομα το ExomosPelatwn.

DiscreteDistribution, με ετικέτα TimiDiaxorismoy που παράγει μια ακολουθία με τους αριθμούς 1 και 2 με πιθανότητες 0.5 και 0.5 αντίστοιχα. Οι αριθμοί αυτοί δηλώνουν τη διαδρομή που πρέπει να ακολουθήσει ένας πελάτης που προσέρχεται καθυστερημένος (1), π.χ. λόγω κυκλοφορίας, ή αναβάλει την παραμονή του στο ξενοδοχείο (1), π.χ. λόγω covid-19, και ένας πελάτης που προσέρχεται στην ώρα του (2). Το αντικείμενο Assign με το όνομα AssignTimiDiax εκχωρεί τις παραπάνω τιμές στην ιδιότητα this.obj.TyposPelatis. Ο διαχωρισμός των πελατών, βάσει τις τιμής της ιδιότητας this.obj.TyposPelatis, γίνεται στο

αντικείμενο Kathysterisi?, όπου οι πελάτες με την τιμή 1 ακολουθούν τη διαδρομή των καθυστερημένων και οι πελάτες με την τιμή 2 ακολουθούν τη διαδρομή των συνεπών.

UniformDistribution, με ετικέτα MeresKathisterisis που παράγει τιμές μεταξύ 4 και 15, οι οποίες αντιπροσωπεύουν το χρόνο καθυστέρησης των πελατών. Το αντικείμενο Assign με το όνομα AssignMeresKathisterisis ελέγχει τις παραπάνω τιμές και εκχωρεί στην ιδιότητα this.obj.TyposPelatis την τιμή 2 για τους πελάτες με καθυστέρηση μεγαλύτερη των 10 s και την τιμή 1 για τους πελάτες με καθυστέρηση μικρότερη η ίση των 10 s. Ο διαχωρισμός γίνεται στο αντικείμενο LockDown? , όπου οι πελάτες με την τιμή 1 ακολουθούν τη διαδρομή των καθυστερημένων και οι πελάτες με την τιμή 2 ακολουθούν τη διαδρομή προς την έξοδο.

DiscreteDistribution, με ετικέτα GenSynepwn που παράγει μια ακολουθία με τους αριθμούς 1, 2, 3, 4, 5, 6 με πιθανότητες 0.3, 0.3, 0.1, 0.1, 0.1 και 0.1 αντίστοιχα. Ο αριθμός αυτός δηλώνει τον αριθμό των συνεπών πελατών που θα εισέλθουν στο ξενοδοχείο για να εξυπηρετηθούν. Οι συνεπείς πελάτες εισέρχονται στο ξενοδοχείο μαζί με έναν καθυστερημένο. Η συνένωση γίνεται από το αντικείμενο Synenwsi.

DiscreteDistribution, με ετικέτα HmeresDiamonis που παράγει μια ακολουθία με τους αριθμούς 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, όλοι με πιθανότητα 0.1. Ο αριθμός αυτός δηλώνει τον αριθμό των ημερών ενοικίασης των δωματίων από τους πελάτες. Οι αριθμοί αυτοί χρησιμοποιούνται από τον μεταφορέα Diamoni.

### 5.1.3 Αντικείμενα εξυπηρέτησης

Το αντικείμενο Assign με το όνομα AssignArDwmatiwn ελέγχει τον αριθμό των διαθέσιμων δωματίων και εκχωρεί στην ιδιότητα this.obj.TyposPelatis την τιμή 1 όταν υπάρχει άδειο δωμάτιο ή την τιμή 2 όταν δεν έχει αδειάσει δωμάτιο. Ο διαχωρισμός γίνεται στο αντικείμενο Dwmatio? , όπου οι πελάτες με την τιμή 1 ακολουθούν τη διαδρομή MetaforeasEisodoy και οι πελάτες με την τιμή 2 ακολουθούν τη διαδρομή προς την έξοδο.

Το αντικείμενο Resource με το όνομα ArithmosReceptionists παρέχει των αριθμό των receptionist στο αντικείμενο Seize με το όνομα Dektos. Το αντικείμενο Resource με το όνομα ArithmosDwmatiwn παρέχει των αριθμό των άδειων δωματίων, επίσης, στο αντικείμενο Dektos. Το αντικείμενο Dektos κάθε φορά που χρησιμοποιεί έναν receptionist μειώνει κατά ένα τον αριθμό τους. Το ίδιο κάνει και με τον αριθμό των δωματίων, κάθε φορά που νοικιάζει ένα δωμάτιο.

Το αντικείμενο Release με το όνομα EleytherwsiReceptionist αυξάνει τον αριθμό των receptionist κατά ένα, αφού εξυπηρετήσει έναν πελάτη.

Το αντικείμενο Release με το όνομα EleytherwsiDwmatiou αυξάνει τον αριθμό των δωματίων κατά ένα, αφού το αφήσει ο πελάτης.

Το αντικείμενο EntitySink με το όνομα Anaxwrisi αποσύρει τους πελάτες.

### 5.1.4 Αποθήκευση δεδομένων

Για την αποθήκευση δεδομένων, χρησιμοποιήσαμε τρία αντικείμενα ExpressionLogger με τα ονόματα DedomenaSynepwn για τις εξής τιμές:

[ErxomosPelatwn].NumberProcessed

[Synepeis].NumberProcessed

[Synepeis].QueueLengthMaximum

[Synepeis].QueueLength

DedomenaKathysterimenwn για τις εξής τιμές:

[ErxomosPelatwn].NumberProcessed

[Kathysterimenoi].NumberProcessed

[Kathysterimenoi].QueueLengthMaximum

[Kathysterimenoi].QueueLength

και DedomenaAnamenontwn για τις εξής τιμές:

[ErxomosPelatwn].NumberProcessed

[AnamenontesDwmatio].NumberProcessed

[AnamenontesDwmatio].QueueLengthMaximum

[AnamenontesDwmatio].QueueLength

Τα αρχεία εξόδου έχουν, αντιστοίχως, τα εξής ονόματα:

Hotels-DedomenaSynepwn.log

Hotels-DedomenaKathysterimenwn.log

Hotels-DedomenaAnemenontwn.log

Εκτελούμε δύο φορές το μοντέλο. Αφού τερματίσει η πρώτη φορά, μετονομάζουμε τα αρχεία αυτά ως εξής:

Hotels-DedomenaSynepwn1.log

Hotels-DedomenaKathysterimenwn1.log

Hotels-DedomenaAnemenontwn1.log

και ξαναεκτελούμε το μοντέλο. Όταν τερματίσει μετονομάζουμε τα νεοδημιουργηθέντα αρχεία αυτά ως εξής:

Hotels-DedomenaSynepwn2.log

Hotels-DedomenaKathysterimenwn2.log

Hotels-DedomenaAnemenontwn2.log

## 5.2 Τελικά βήματα

Ορίζουμε το χρόνο προσομοίωσης ως εξής:

- Επιλέγουμε Simulation στον Object Selector.
- Πραγματοποιούμε τις ακόλουθες αλλαγές στον Input Editor:

| **Χαρακτηριστικό** | **Τιμή** |
|---|---|
| RunDuration | 10 min |

Δίνουμε τίτλο στο μοντέλο:

- **Object Selector** -> **Graphics Objects** -> **Overlay Text** -> **Title** στο πλαίσιο Format πληκτρολογούμε 'Hotel'.

## 5.3 Εκτέλεση της προσομοίωσης

Η γραμμή εργαλείων περιέχει στοιχεία ελέγχου της εκτέλεση της προσομοίωσης:

| **Στοιχείο γραμμής εργαλείων** | **Τι κάνει** |
|---|---|
| | Εκκίνηση, παύση και συνέχιση την εκτέλεση της προσομοίωσης. |
| | Τερματίζει την προσομοίωση, απαλείφει τις δημιουργηθείσες οντότητες, θέτει το χρόνο προσομοίωσης στο μηδέν. |
| | Όταν ενεργοποιηθεί, η προσομοίωση εκτελείται σε ένα σταθερό πολλαπλάσιο της ώρας του ρολογιού. |
| | Ο χρόνος αυτόματης παύσης της προσομοίωσης. |

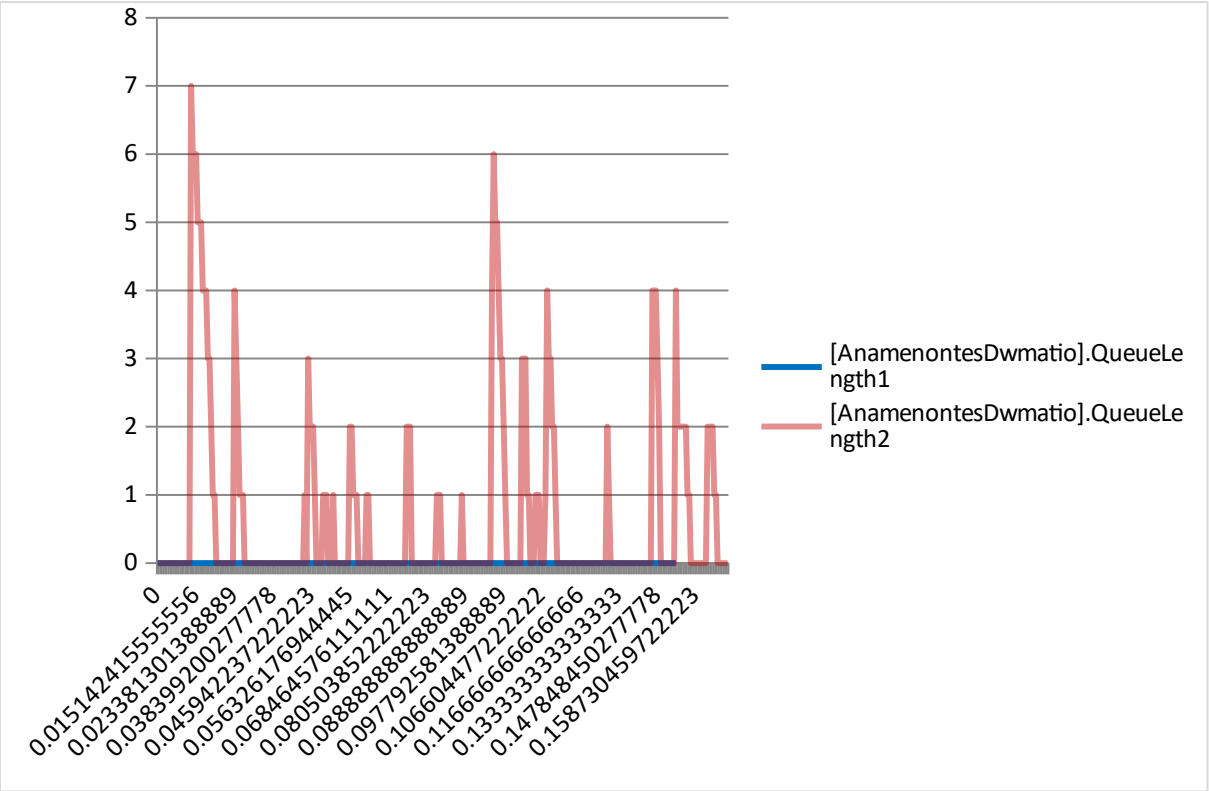
# 6. Αποτελέσματα

Πραγματοποιήσαμε δύο εκτελέσεις του μοντέλου. Στην πρώτη ο αριθμός των receptionist ήταν 2 και ο αριθμός των δωματίων 10. Στην δεύτερη ο αριθμός των receptionist ήταν 1 και ο αριθμός των δωματίων 3. Αυτό που θέλαμε να δούμε είναι η διαφορά στην εξυπηρέτηση πελατών, όπως αυτή θα καταγράφονταν ανά 20 s στο μέγεθος της ουράς αναμενόντων για δωμάτιο. Οι καταχωρήσεις έγιναν στα αρχεία Hotels-DedomenaAnamenontwn1.log και Hotels-DedomenaAnamenontwn2.log.

- Τα δύο αυτά αρχεία εισάγαμε σε φύλλο του Excel, ως εξής:
- Ενεργοποιούμε την καρτέλα "Δεδομένα".
- Ενεργοποιούμε την υποκαρτέλα "Λήψη εξωτερικών δεδομένων".
- Επιλέγουμε "Από κείμενο".
- Επιλέγουμε το αρχείο "Hotels-DedomenaAnamenontwn1.log".
- Πατάμε το κουμπί "επόμενο" δύο φορές και μετά "Τέλος".
- Διαλέγουμε το κελί "A1" που θα είναι η πάνω αριστερή γωνία του πίνακα.

Το ίδιο κάνουμε με το αρχείο "Hotels-DedomenaAnamenontwn2.log", αλλά επιλέγουμε το κελί "G1" ως την πάνω αριστερή γωνία του πίνακα. Στη συνέχεια επιλέγουμε τη στήλη QueueLength και από τους δύο πίνακες και δημιουργούμε ένα γράφημα σε ξεχωριστεί σελίδα.

Τα δεδομένα της στήλης QueueLength, από κάθε αρχείο, απεικονίζονται στο γράφημα που ακολουθεί.




Παρατηρούμε ότι στην πρώτη εκτέλεση το μήκος της ουράς είναι 0 σε κάθε χρονική στιγμή καταγραφής. Στη δεύτερη εκτέλεση το μήκος της ουράς αυξομειώνεται ανάλογα με τον αριθμό των πελατών και τους πόρους εξυπηρέτησης.

Παρατηρήστε ότι ο αριθμός των πελατών που εξυπηρετήθηκαν στην πρώτη εκτέλεση είναι 176, ενώ στη δεύτερη (λιγότερα δωμάτια) είναι 137.

Μπορούμε να πειραματιστούμε με πάρα πολλούς τρόπους, για παράδειγμα στην κατανομή, που χρησιμοποιείται από τη γεννήτρια πελατών, μπορούμε να αλλάξουμε την τιμή των χαρακτηριστικών RandomSeed και Mean. Έτσι μπορούμε να δούμε τη συμπεριφορά του μοντέλου, όταν αυξηθεί ή μειωθεί ο ρυθμός προσέλευσης των πελατών.

# 7. Συμπέρασμα

Σε αυτήν την εργασία συζητήσαμε την ανάγκη για σαφείς δομές ελέγχου στα εννοιολογικά μοντέλα. Καθώς η εννοιολογική μοντελοποίηση δηλώνει τη διαδικασία αφαίρεσης από τον πραγματικό κόσμο και της δημιουργίας ενός απλοποιημένου μοντέλου, οι συνιστώσες και η ατομική συμπεριφορά δεν είναι οι μόνες έννοιες που πρέπει να προσδιοριστούν και να τεκμηριωθούν. Οι στρατηγικές και οι πολιτικές ελέγχου πρέπει να περιλαμβάνονται ρητά σε οποιονδήποτε ορισμό του εννοιολογικού μοντέλου.

Αν αποκτήσουμε το σωστό εννοιολογικό μοντέλο, τότε θα είναι πιο απλό και το υπόλοιπο έργο προσομοίωσης, παρέχοντας τις σωστές πληροφορίες στη σωστή χρονική κλίμακα.

Παρουσιάσαμε το λογισμικό JaamSim και παρείχαμε μια επισκόπηση πολλών από τα χαρακτηριστικά του. Στη συνέχεια το χρησιμοποιήσαμε για την προσομοίωση ενός ξενοδοχείου, τα αποτελέσματα της οποίας παρουσιάσαμε στην προηγούμενη ενότητα

# Βιβλιογραφία


Arthur, J.D. and R.E. Nance, "Investigating the use of software requirements engineering techniques in simulation modeling", *Journal of simulation, vol* 1, nr 3, 2007, pp. 159-174.

Bahouth, A., et al, "Revisiting the Issue of Performance Enhancement of Discrete Event Simulation Software", Simulation Symposium, 2007. ANSS '07. 40th Annual.

Balci, O., "Validation, verification and testing techniques throughout the life cycle of a simulation study", Annals of Operations Research, vol 53, nr 1, 1994, pp. 121-173.

Balci, O. and R.E. Nance, "Formulated problem verification as an explicit requirement of model credibility", *Simulation, vol* 45, nr 2, 1985, pp. 76-86.

Balci, O. and W.F. Ormsby, "Conceptual modeling for designing large-scale simulations", Journal of Simulation, vol 1, nr 3, 2007, pp. 175-186.

Birta, L.G. and G. Arbez, "Exploring Dynamic System Behaviour", in *Modelling and Simulation*, Springer, 2007.

Brooks, R.J. and A.M. Tobias, "A framework for choosing the best model structure in mathematical and computer modeling", In *Proceedings of the 6th Annual Conference AI, Simulation, and Planning in High Autonomy Systems*, 1996, pp. 53-60.

Cassandras, C.G., M.I. Clune, and P.J. Mosterman, "Discrete Event and Hybrid System Simulation with SimEvents", Proceedings of 2006 Intl. Workshop on Discrete Event Systems, 2006, pp. 386–387.

Dosi, C. et al., "Computational Simulation as an organizational prototyping tool", Proceedings of the Design Society: International Conference on Engineering Design, 2019.

Gordon, G., *A General Purpose Systems Simulation Program*, In *Proceedings of the Eastern Joint Computer Conference,* 1961, Washington, D.C.

Guru, A. and P. Savory, "A template-based conceptual modeling infrastructure for simulation of physical security systems". In *Proceedings of the 2004 Winter Simulation Conference*, 2004, pp. 866-873.

Harrison, H., H. King, and P.-L. Paour, "Jaamsim: Release 2016-14", Zenodo.

JaamSim - User Manual, January 2021, JaamSim Software Inc.

Karagöz, N. and O. Demirörs, "Conceptual modeling notations and techniques", in Conceptual Modeling for Discrete-Event Simulation, CRC Press, 2010, pp. 179-209.

Kelton, W.D., R.P. Sadowski, and D.T. Sturrock, *Simulation with Arena,* 4th ed., McGraw Hill, New York, 2007.

King, D.H. and H.S. Harrison, "Open-source simulation software 'jaamsim'", Proceedings of the 2013 Winter Simulation Conference, pp. 2163-2171.

Kotiadis, K., "Using soft systems methodology to determine the simulation study objectives", *Journal of simulation, vol* 1, nr 3, 2007, pp. 215-222.

Krahl, D., "The Extend Simulation Environment", Proceedings of 2002 Winter Simulation Conference, 2002, pp. 205–213.

Kreutzer, W., "Computer system modelling and simulation", ACM SIGMETRICS Performance Evaluation Review, 1979, Vol. 8, No. 1-2, pp. 9-35.

Kreutzer, W., *System Simulation Programming Styles and Languages*, Addison-Wesley, Reading, MA, 1986.

Lacy, L.W., et al., "Developing a consensus perspective on conceptual models for simulation systems", in Proceedings of the 2001 Spring Simulation Interoperability Workshop, 2001.

Lavrijsen, W., "Optimizing python-based ROOT I/O with PyPy's tracing just-in-time compiler", Journal of Physics: Conference Series, Vol. 396, 2012.

Law, A.M., and W.D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1991.

Nance, R., "The conical methodology and the evolution of simulation model development", Annals of Operations Research, vol 53, nr 1, 1994, pp. 1-45.

Nance, R. and J.D. Arthur, "Software requirements engineering: exploring the role in simulation model development", in Proceedings of the Third Operational Research Society Simulation Workshop (SW06), 2006, pp. 117-127.

Pace, D.K., "Development and documentation of a simulation conceptual model", in Proceedings of the 1999, Fall Simulation Interoperability Workshop, 1999.

Pegden, C.D., R.E. Shannon, and R.P. Sadowski, *Introduction to Simulation Using SIMAN*, McGraw-Hill, New York, 1990.

Pegden, C.D., & D.A. Davis, "Arena: a SIMAN/Cinema-Based Hierarchical Modeling System", In *Proceedings of the 1992 Winter Simulation Conference,* Piscataway, NJ, IEEE, pp. 390-399.

Pritsker, A.A.B., *Introduction to Simulation and SLAM II*, 4th ed., 1995, Wiley, New York.

Rice, S.V., A. Marjanski, H.M. Markowitz, and S.M. Bailey, "The SIMSCRIPT III Programming Language for Modular Object-Oriented Simulation", Proceedings of 2005 Winter Simulation Conference, 2005, pp. 621–630.

Robinson, S., *Simulation: The Practice of Model Development and Use*, John Wiley & Sons, Ltd, 2004.

Robinson, S., "Conceptual modeling for simulation: issues and research requirements", in Proceedings of the 38th Conference on Winter Simulation, WSC ’06, Winter Simulation Conference, 2006, pp. 792–800.

Robinson, S., "Conceptual Modeling for Simulation Part I: Definition and Requirements", Journal of the Operational Research Society, 2008, vol 59, nr 3, p. 278-290.

Robinson, S., "Choosing the right model: conceptual modeling for simulation", in Proceedings of the 2011 Winter Simulation Conference (WSC), 2011, pp. 1423-1435.

Robinson, S., "Tutorial: choosing what to model - conceptual modeling for simulation", in Proceedings of the 2012 Winter Simulation Conference (WSC), 2012, pp. 1-12.

Schriber T, D. Brunner and J. Smith, "How discrete-event simulation software works and why it matters", Proceedings of the Winter Simulation Conference, Berlin, Germany, 2012, pp 1-15.

Shannon, R.E., *Systems Simulation*, Prentice-Hall, 1975.

Taylor SJE, T. Kiss, G. Terstyanszky, P. Kacsuk and N. Fantini, "Cloud computing for simulation in manufacturing and engineering: Introducing the CloudSME simulation platform", Proceedings of the 2014 Annual Simulation Symposium, Society for Computer Simulation International, San Diego, CA, USA.

van der Zee, D.J. and J.G.A.J. van der Vorst. 2005. A modeling framework for supply chain simulation: Opportunities for improved decision making", *Decision sciences*, vol 36, nr 1, 2005, pp. 65-95.

Weidemann, D. et al., "Product-service system development with discrete event simulation", in IEEE systems conference (SysCon), 2015.

Zeigler, B.P., "Hierarchical, modular discrete-event modelling in an object-oriented environment", Simulation vol 49, nr 5. 1987, pp. 219-230.