ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

# ΣΥΣΤΗΜΑ ΕΚΤΕΛΕΣΗΣ ΡΟΩΝ ΕΡΓΑΣΙΑΣ ΒΑΣΙΣΜΕΝΟ ΣΕ ΠΡΑΚΤΟΡΕΣ

## ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**ΑΝΤΩΝΙΟΣ Κ. ΣΜΑΡΔΑΣ**

ΗΡΑΚΛΕΙΟ, ΙΑΝΟΥΑΡΙΟΣ 2003

# Σύστημα Εκτέλεσης Ροών Εργασίας Βασισμένο σε Πράκτορες

Αντώνιος Κ. Σμαρδάς

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

## ΠΕΡΙΛΗΨΗ

Οι ρυθμοί ανάπτυξης που γνωρίζει τα τελευταία χρόνια το Διαδίκτυο είναι ενδεικτικοί της τάσης διασύνδεσης υπολογιστών και της παροχής πρόσβασης σε ευρείες ομάδες πληθυσμού σε συνεχώς περισσότερες και πλουσιότερες πηγές πληροφοριών. Ο χώρος αυτός φαντάζει ως ιδανικός για την ανάπτυξη πιλοτικών συστημάτων που θα επιχειρήσουν να εκμεταλλευτούν την ύπαρξη πληθώρας διαθέσιμων πόρων κατανεμημένων στο δίκτυο για την παροχή νέων υπηρεσιών. Η ανάπτυξη των Ψηφιακών Βιβλιοθηκών συνηγορεί στην παραπάνω άποψη. Η χρήση μίας Ψηφιακής Βιβλιοθήκης παρέχει τη δυνατότητα καθολικής πρόσβασης στα στοιχεία μίας βιβλιοθήκης, τα οποία είναι πλέον σε ψηφιακή μορφή, από οποιοδήποτε υπολογιστή έχει πρόσβαση στο δίκτυο.

Το έργο ARION φιλοδοξεί να παρέχει μία νέα γενιά υπηρεσιών Ψηφιακής Βιβλιοθήκης για την αναζήτηση κι ανάκτηση επιστημονικών ψηφιακών συλλογών. Ο ARION πέρα από τη βασική λειτουργικότητα που παρέχει μία Ψηφιακή Βιβλιοθήκη υποστηρίζει τη δυνατότητα δυναμικής ανεύρεσης πόρων για την περίπτωση όπου τα δεδομένα δεν είναι άμεσα διαθέσιμα, αλλά μπορούν να παραχθούν δυναμικά μέσω της εκτέλεσης μίας ροής εργασίας (workflow) που περιλαμβάνει υπάρχοντα προγράμματα και δεδομένα.

Σκοπός της παρούσας εργασίας είναι η ανάπυξη του runtime συστήματος του ARION το οποίο υποστηρίζει την εκτέλεση ροών εργασίας για τη δυναμική παραγωγή πληροφοριών. Το σύστημα επιτρέπει δύο διαφορετικές καταστάσεις εκτέλεσης ροών εργασίας (αλληλεπιδραστική - αυτοματοποιημένη) και παρουσιάζει χαρακτηριστικά που προσδίδουν περαιτέρω αξία σε αυτό, όπως υποστήριξη backtracking (δυνατότητα επιστροφής σε προηγούμενο βήμα μίας ροής εργασίας κατά τη διάρκεια της εκτέλεσής

της), ανοχή σε λάθη/αποτυχίες, υψηλό ποσοστό διαθεσιμότητας, καθώς και υποστήριξη Java Applets ως συστατικό στοιχείο μίας ροής εργασίας. Η εκτέλεση των ροών εργασίας βασίζεται στη χρήση τεχνολογίας κινούμενων πρακτόρων. Οι χρήστες αλληλεπιδρούν με το runtime σύστημα μέσω ενός απλού web interface που έχει υλοποιηθεί.

Επόπτης: Αικατερίνη Χούστη
Καθηγήτρια Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

# Agent-based Runtime System for Workflow Execution

Antonios K. Smardas

Master of Science Thesis

Department of Computer Science
University of Crete
Greece

## ABSTRACT

The rapid development of the Internet over the last years clearly indicates a trend towards connecting computers that provide vast amount of people with access to information sources that keep growing bigger and richer. This area seems ideal for the development of prototype systems that take advantage of the existence of such resources that are distributed over the network for the development of value-added services. The development of Digital Libraries (DL) supports the above idea. A DL provides global access to the elements of the library, now in a digital form, from any computer connected to the Internet.

The ARION project has the ambition to provide a new generation of DL Services for the search and retrieval of scientific collections. Apart from the basic functionality provided by any DL, ARION provides special support for the case where data are not directly available but can be produced dynamically, through a workflow involving several existing data sets and programs.

The aim of this work is the development of the runtime system of ARION that supports the execution of workflows for the facilitation of data production. The system supports two execution modes (interactive and automated mode) and has several features that enhance its value, such as the support for backtracking, fault tolerance, high availability as well as support for the integration of Java Applets in a workflow. The execution of workflows is agent-based. The users may interact with the system through a simple web interface that has been developed.

Supervisor: Catherine Houstis

Professor of Computer Science

University of Crete

# Ευχαριστίες

Στο τέλος της πορείας των σπουδών μου στο Πανεπιστήμιο Κρήτης νιώθω ότι αποκόμισα πολλά από αυτό τόσο σε ό,τι αφορά ευκαιρίες μελλοντικής επαγγελματικής σταδιοδρομίας όσο και σε ανθρώπινο επίπεδο.

Θα ήθελα ιδιαίτερα να ευχαριστήσω την οικογένεια μου για τη συνολική βοήθεια που μου παρείχε στα χρόνια των σπουδών μου. Χωρίς τη βοήθειά της σίγουρα δε θα τα κατάφερνα ως εδώ.

Επίσης, ευχαριστώ τους καθηγητές μου του Τμήματος Επιστήμης Υπολογιστών για τις γνώσεις που μου παρείχαν όλα αυτά τα χρόνια. Ιδιαιτέρως ευχαριστώ τους επόπτες καθηγητές μου Αικατερίνη Χούστη και Σπύρο Λάλη για τις πολύτιμες συμβουλές τους και τις υποδείξεις τους καθόλη τη διάρκεια της μεταπτυχιακής εργασίας μου. Ευχαριστώ τους καθηγητές μου Δημήτρη Πλεξουσάκη και Βασίλη Χριστοφίδη για τη συμμετοχή τους στην επιτροπή εξέτασης της εργασίας μου και τις εποικοδομητικές παρατηρήσεις τους. Ευχαριστώ και τους καθηγητές μου Μανόλη Μαραζάκη και Ευάγγελο Μαρκάτο για τη βοήθεια τους στην εμβάθυνση σε θέματα κατανεμημένων συστημάτων.

Θέλω ακόμα να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών και το Ίδρυμα Τεχνολογίας και Έρευνας για την υλικοτεχνική υποδομή που μου παρείχαν χάρη στην οποία ήταν εφικτή η πραγματοποίηση της εργασίας μου. Ευχαριστώ θερμά τους συνεργάτες μου σε αυτήν την εργασία: Κυριάκο Κρητικό, Μάριο Πιτικάκη, Χαράλαμπο Γκίκα, Γιώργο Βασιλάκη, Λευτέρη Σιδηρουργό, Θεόδωρο Γεροστάθη και τον καθηγητή Μανόλη Βάβαλη.

Επιπλέον, θα ήθελα να ευχαριστήσω αρκετούς σημαντικούς φίλους για τη συνολικότερη βοήθεια που μου παρείχαν τα χρόνια των σπουδών μου: Χρήστο Παπαχρήστο, Νίκο Γιανναδάκη, Μανόλη Σταματογιαννάκη, Μανόλη Δελάκη, Γιώργο Ζαχαριουδάκη, Νίκο Ριζόπουλο, Ματίνα Καβουρίδου, Εύα Καλυβιαννάκη, Λουκά Νίκαινα, Νίκο Μπέρδο, Σωτήρη Τουρτούνη, Ευάγγελο Αντωνιάδη, Στέλιο Μαυρομιχάλη, Σωκράτη Δημητριάδη. Ελπίζω να έχουν όλοι τους την εξέλιξη που επιθυμούν.

# ΠΕΡΙΕΧΟΜΕΝΑ


**Ευχαριστίες** ..... 9
**ΠΕΡΙΕΧΟΜΕΝΑ** ..... 11
**Πίνακας Σχημάτων** ..... 13
**Πίνακας Εικόνων** ..... 13
1.1 Οργάνωση Αναφοράς ..... 16
**Κεφάλαιο 2ο: Συστήματα Ροών Εργασίας** ..... 18
2.1 Ορισμός Ροής Εργασίας ..... 18
2.2 Περιγραφή Ροής Εργασίας ..... 18
2.3 Διαστάσεις μίας Ροής Εργασίας ..... 19
2.4 Ωφέλειες από τη χρήση Ροών Εργασίας ..... 20
2.5 Συστήματα Διαχείρισης Ροών Εργασίας ..... 21
2.6 Υλοποιημένα Συστήματα Διαχείρισης Ροών Εργασίας ..... 22
2.7 Γλώσσες και Μετα-γλώσσες Ορισμού Ροών Εργασίας ..... 25
2.8 Διαλειτουργικότητα συστημάτων ροών εργασίας ..... 37
**Κεφάλαιο 3ο: ARION** ..... 41
3.1 Εισαγωγή ..... 41
3.2 Οι στόχοι του ARION ..... 43
3.3 Λίστα συμμετεχόντων ..... 44
3.4 Καινοτομία ..... 45
**Κεφάλαιο 4ο: Τεχνολογία κινούμενων πρακτόρων** ..... 53
4.1 Εισαγωγή ..... 53
4.2 Τι είναι ένας κινούμενος πράκτορας; ..... 53
4.3 Γιατί κινούμενοι πράκτορες; ..... 55
4.4 Πλατφόρμες κινούμενων πρακτόρων ..... 57
**Κεφάλαιο 5ο: Σχετική Εργασία** ..... 68
5.1 Service-Oriented Middleware ..... 68
5.2 Message Oriented Middleware ..... 70
5.3 Distributed Event Systems ..... 70
5.4 Virtual Shared Memory Middleware ..... 71
5.5 Peer-to-Peer Middleware ..... 71
**Κεφάλαιο 6ο: Το runtime σύστημα του ARION** ..... 73
6.1 Εισαγωγή ..... 73
6.2 Αρχιτεκτονική ..... 73
6.3 Εκτέλεση Ροών Εργασίας ..... 94
6.4 Χαρακτηρισικά runtime συστήματος ..... 103
6.5 Διαχείριση συστήματος ..... 118
6.6 Στοιχεία υλοποίησης ..... 123
**Κεφάλαιο 7ο: Σενάρια Χρήσης** ..... 124
7.1 Πρώτο σενάριο: Αλληλεπιδραστική εκτέλεση ..... 124
7.2 Δεύτερο σενάριο: Προγραμματισμένη εκτέλεση ..... 128
7.3 Τρίτο σενάριο: Χρήση Java Applet ..... 130
**Κεφάλαιο 8ο: Συμπεράσματα,Συζήτηση– Μελλοντικές Κατευθύνσεις** ..... 132
8.1 Ανακεφαλαίωση ..... 132
8.2 Μελλοντικές Επεκτάσεις ..... 133

# Πίνακας Σχημάτων

**Σχήμα 3.1. - Αρχιτεκτονική για κατανεμημένες επιστημονικές πηγές.....52**
**Σχήμα 6.1. - Η αρχιτεκτονική του runtime συστήματος του ARION.....74**
**Σχήμα 6.2. - Αίτηση για χρήση πράκτορα από έναν εκτελεστή εργασίας.....80**
**Σχήμα 6.3. - Μηχανισμός επικοινωνίας .....81**
**Σχήμα 6.4. - Μηχανή εκτέλεσης ροών εργασίας .....84**
**Σχήμα 6.5. - Μεταβάσεις καταστάσεων σε μία εργασία.....86**
**Σχήμα 6.6. - Παράδειγμα ροής εργασίας με τις συνδέσεις του control flow.....87**
**Σχήμα 6.7. - Παράδειγμα ροής εργασίας με control flow και backtracking.....88**
**Σχήμα 6.8. - Pool εκτελεστών εργασίας .....90**
**Σχήμα 6.9. - Ουρά ενεργειών .....91**
**Σχήμα 6.10. - Μεταφορά αρχείου με χρήση πρακτόρων .....93**
**Σχήμα 6.11. - Group μηχανών εκτέλεσης ροών εργασίας .....102**
**Σχήμα 6.12. - Δενδρικό σχήμα ροής εργασίας .....105**
**Σχήμα 6.13. - Η αρχιτεκτονική των δακτυλίων .....107**
**Σχήμα 6.14a. - Ανάνηψη έπειτα από αποτυχία – Α' φάση.....110**
**Σχήμα 6.14b. - Ανάνηψη έπειτα από αποτυχία – Β' φάση.....110**
**Σχήμα 6.14c. - Ανάνηψη έπειτα από αποτυχία – Γ' φάση .....111**
**Σχήμα 6.14d. - Ανάνηψη έπειτα από αποτυχία – Δ' φάση.....112**
**Σχήμα 6.15. - Σχήμα πρωτεύοντος – εφεδρικού server .....115**
**Σχήμα 6.16. - Επικοινωνία Java Applet με το runtime σύστημα .....116**
**Σχήμα 7.1. - Παράδειγμα ροής εργασίας σε αλληλεπιδραστική εκτέλεση .....124**
**Σχήμα 7.2. - Παράδειγμα ροής εργασίας σε αυτοματοποιημένη εκτέλεση.....128**
**Σχήμα 7.3. - Παράδειγμα ροής εργασίας που ενσωματώνει ένα Java Applet ....131**

# Πίνακας Εικόνων

**Εικόνα 4.1. - Το περιβάλλον του Grasshopper.....49**
**Εικόνα 6.1. - Λίστα ροών εργασίας.....86**
**Εικόνα 6.2. - Τρέχουσα κατάσταση ροής εργασίας.....87**
**Εικόνα 6.3. - Ιστορικό εκτέλεσης ροής εργασίας .....88**
**Εικόνα 6.4. - Διαθέσιμες εντολές του command-line εργαλείου .....109**
**Εικόνα 6.5. - Απαρίθμηση ροών εργασίας στο σύστημα .....110**
**Εικόνα 6.6. - Τρέχουσα κατάσταση μίας ροής εργασίας .....111**
**Εικόνα 6.7. - Εμφάνιση της κατάστασης μίας ροής εργασίας .....112**

# Κεφάλαιο 1ο: Εισαγωγή

Η ταχεία ανάπτυξη των κατανεμημένων συστημάτων και η ολοένα και ευρύτερη αποδοχή που αυτά γνωρίζουν από την ευρύτερη επιστημονική κοινότητα, αλλά και από μεγάλες ομάδες χρηστών καθιστούν πλέον εφικτή και σκόπιμη την κατασκευή συστημάτων που θα εκμεταλλεύονται ήδη υπάρχουσες τεχνολογίες παρέχοντας μια νέα γενιά ηλεκτρονικών υπηρεσιών. Οι νέες value-added υπηρεσίες που δημιουργούνται παρέχουν δυνατότητες αποδοτικότερης πρόσβασης και αξιοποίησης πόρων που προηγουμένως παρέμεναν σε μεγάλο βαθμό ανεκμετάλλευτοι. Οι ευκαιρίες για ανάπτυξη τέτοιων συστημάτων είναι σημαντικές και υπάρχει αρκετός χώρος για πειραματισμό και ανάπτυξη πρωτότυπων συστημάτων που θα αποσκοπούν στην παροχή εξελιγμένων υπηρεσιών μέσω του συνδυασμού και της εξέλιξης υφιστάμενων λύσεων.

Οι ρυθμοί ανάπτυξης που γνωρίζει τα τελευταία χρόνια το διαδίκτυο είναι ενδεικτικοί της τάσης διασύνδεσης υπολογιστών και της παροχής πρόσβασης σε ευρείες ομάδες χρηστών, ειδικών και μη, σε συνεχώς και περισσότερες/πλουσιότερες πηγές πληροφόρησης. Η ανάπτυξη των Ψηφιακών Βιβλιοθηκών είναι ένα στοιχείο που συνηγορεί στην άποψη αυτή. Με την εμφάνιση των Ψηφιακών Βιβλιοθηκών τροποποιείται σημαντικά η συμβατική αντίληψη που υπάρχει για μία βιβλιοθήκη, σύμφωνα με την οποία η πρόσβαση στην πληροφορία/γνώση που διαθέτει η βιβλιοθήκη είναι δυνατή μόνο από το μέρος όπου αυτή βρίσκεται και ότι τα στοιχεία της είναι σε μορφή που δύσκολα αναπαράγονται και μεταφέρονται σε μεγάλες ομάδες πληθυσμού. Με τη χρήση των Ψηφιακών Βιβλιοθηκών παρέχεται η δυνατότητα καθολικής πρόσβασης στα στοιχεία μίας βιβλιοθήκης, τα οποία πλέον είναι σε ψηφιακή μορφή, από έναν οποιοδήποτε υπολογιστή που είναι συνδεδεμένος στο δίκτυο και με αυξημένες δυνατότητες αναζήτησης της επιθυμητής πληροφορίας.

Το έργο ARION φιλοδοξεί να παρέχει μια νέα γενιά υπηρεσιών Ψηφιακής Βιβλιοθήκης για την αναζήτηση και ανάκτηση επιστημονικών ψηφιακών συλλογών που βρίσκονται σε ερευνητικούς και συμβουλευτικούς οργανισμούς. Η σχεδίαση του ARION είναι τέτοια ώστε να υποστηρίζεται αφενός η αναζήτηση και η πρόσβαση σε επιστημονικά αντικείμενα και αφετέρου η λειτουργία ροών εργασίας οι οποίες επεξεργάζονται επιστημονικά δεδομένα, π.χ. δορυφορικές εικόνες, time series κ.α. Ο ARION επιχειρεί να

χρησιμοποιήσει και να αναβαθμίσει τα ήδη υπάρχοντα αποτελέσματα μελετών στην περιοχή. Ο στόχος του ARION είναι να ξεπεράσει την υπάρχουσα έλλειψη προσβασιμότητας στην κληρονομιά επιστημονικών συλλογών δεδομένων και να προσφέρει μία νέα προσέγγιση στην προώθηση της επαναχρησιμοποίησής τους. Συγκεκριμένα, πέρα από την ανάπτυξη μίας Ψηφιακής Βιβλιοθήκης ο ARION επιχειρεί την ανάπτυξη νέων μεθόδων και τεχνικών απαραίτητων για τη διαλειτουργικότητα των δεδομένων και των λειτουργιών της ψηφιακής βιβλιοθήκης.

Στα πλαίσια αυτής της εργασίας υλοποιήθηκε ένα τμήμα του συστήματος ARION που κινείται στην κατεύθυνση της διασύνδεσης ετερογενών συλλογών επιστημονικών συλλογών (δεδομένων και προγραμμάτων). Αντικείμενο της εργασίας ήταν η υλοποίηση του μηχανισμού εκτέλεσης ροών εργασίας. Το σύστημα που υλοποιήθηκε είναι σε θέση να εκτελεί ροές εργασίας προκειμένου να παραχθεί πληροφορία που έχει ζητηθεί από ένα χρήστη. Το σύστημα επιτρέπει την παρακολούθηση καθώς και τη διαχείριση των εκτελούμενων ροών εργασίας. Για την εκτέλεση των ροών εργασίας υιοθετήθηκε μια λύση βασισμένη σε πράκτορες λογισμικού.

Το σύστημα αυτό υλοποιήθηκε σε συνεργασία με παροχείς επιστημονικών συλλογών και βάσει πραγματικών παραδειγμάτων ροών εργασίας που συνδυάζουν ετερογενείς πηγές επιστημονικών συλλογών δεδομένων. Οι σχεδιαστικές επιλογές που έγιναν, π.χ. λειτουργικότητα backtracking που επιτρέπει το calibration παραμέτρων εισόδου επιστημονικών προγραμμάτων, αντικατοπτρίζουν ως ένα βαθμό τις ανάγκες των κυρίων χρηστών του συστήματος που συνήθως είναι μέλη της ακαδημαϊκής-ερευνητικής κοινότητας. Το σύστημα αυτό συγκεντρώνει χαρακτηριστικά που συμβάλλουν στην ποιότητα της παρεχόμενης λειτουργικότητας, όπως εγγύηση της συνέπειας των εκτελούμενων λειτουργιών και αποτελεσμάτων, υψηλό ποσοστό διαθεσιμότητας του συστήματος και ανοχή σε λάθη.

## 1.1 Οργάνωση Αναφοράς

Η οργάνωση της υπόλοιπης αναφοράς της παρούσας εργασίας είναι η ακόλουθη:

**Στο 2$^{ο}$ κεφάλαιο** αναλύεται η έννοια της ροής εργασίας. Επίσης παρουσιάζονται συστήματα ροών εργασίας, καθώς και οι προσπάθειες που αναπτύσσονται για την επίτευξη διαλειτουργικότητας μεταξύ συστημάτων ροών εργασίας διαφορετικών κατασκευαστών.

**Στο 3$^{ο}$ κεφάλαιο** γίνεται μία περιεκτική περιγραφή του συστήματος ARION όπου και εξηγούνται οι στόχοι που θέτει. Παρουσιάζεται η αρχιτεκτονική του ολοκληρωμένου συστήματος και γίνεται μία σύντομη περιγραφή των επιμέρους μερών (components).

**Στο 4$^{ο}$ κεφάλαιο** γίνεται αναφορά στην τεχνολογία κινούμενων πρακτόρων και εξηγούνται οι λόγοι για τους οποίους υιοθετήθηκε η χρήση τους από το runtime σύστημα. Επίσης, γίνεται μια παρουσίαση της πλατφόρμας κινούμενων πρακτόρων Grasshopper η οποία επιλέχθηκε για την παροχή της λειτουργικότητας αυτής, ενώ γίνεται αναφορά και.σε άλλες πλατφόρμες κινούμενων πρακτόρων.

**Στο 5$^{ο}$ κεφάλαιο** εξετάζεται η σχετική εργασία που αναφέρεται στη βιβλιογραφία σε σχέση με το σύστημα που υλοποιήθηκε στην παρούσα εργασία.

**Στο 6$^{ο}$ κεφάλαιο** γίνεται η κύρια παρουσίαση του runtime συστήματος του ARION που αποτελεί και τον κορμό της παρούσας εργασίας. Περιγράφεται η αρχιτεκτονική του συστήματος, ενώ εξηγούνται και οι σχεδιαστικές αποφάσεις που λήφθηκαν για την υλοποίησή του.

**Στο 7$^{ο}$ κεφάλαιο** παρουσιάζονται τρία σενάρια χρήσης του συστήματος, όπου επιχειρούμε να παρουσιάσουμε την παρεχόμενη λειτουργικότητα του runtime συστήματος μέσω παραδειγμάτων χρήσης του.

**Στο 8$^{ο}$ κεφάλαιο** γίνεται μια ανασκόπηση του συστήματος, παρουσιάζονται τα συμπεράσματα της εργασίας, ενώ καταγράφονται και οι μελλοντικές κατευθύνσεις που βλέπουμε να ακολουθούνται πάνω στην εργασία αυτή.

# Κεφάλαιο 2ο: Συστήματα Ροών Εργασίας

## 2.1 Ορισμός Ροής Εργασίας

Σύμφωνα με το Workflow Management Coalition [1], [2] μια ροή εργασίας (workflow) ορίζεται ως ο πλήρης ή μερικός αυτοματισμός ενός επιχειρησιακού μοντέλου κατά τον οποίο έγγραφα, πληροφορίες ή έργα (tasks) μεταφέρονται από τον ένα συμμετέχοντα στον επόμενο για εκτέλεση βάσει ενός συνόλου κανόνων. Μία επιχειρησιακή διαδικασία αποτελείται από ένα σύνολο διαδικασιών ή ενεργειών οι οποίες συλλογικά πραγματοποιούν μία επιχειρησιακή επιδίωξη ή ένα στόχο πολιτικής-τακτικής, συνήθως στο πλαίσιο μίας οργανωτικής δομής οριζόμενης από λειτουργικούς ρόλους και σχέσεις.

Οι διαδικασίες δεν σχετίζονται/εκτελούνται αναγκαστικά σε έναν υπολογιστή. Ένας μεγάλος αριθμός από επιχειρησιακές διαδικασίες μπορεί να εκτελεστεί χωρίς να απαιτείται κανένα απολύτως βήμα εκτέλεσης με τη χρήση υπολογιστή. Επομένως, οι επιχειρησιακές διαδικασίες μπορεί να αποτελούνται από κομμάτια που εκτελούνται από υπολογιστή καθώς κι από κομμάτια που δεν εκτελούνται από υπολογιστή. Ο βαθμός αυτοματισμού της ροής αποτελεί βασικό χαρακτηριστικό των ροών εργασίας. Το μέτρο αυτό υποδηλώνει την ανεξαρτησία μίας ροής εργασίας από την ανθρώπινη παρέμβαση, δηλαδή το κατά πόσο οι ενέργειες μιας ροής εργασίας εκτελούνται από ανθρώπους ή από το σύστημα.

## 2.2 Περιγραφή Ροής Εργασίας

Η περιγραφή (specification) μιας ροής εργασίας απαρτίζεται από τα ακόλουθα στοιχεία:

- Εργασίες προς εκτέλεση (tasks)
- Σειρά εκτέλεσης των εργασιών (control flow)
- Data sets
- Data flow

Μία ροή εργασίας αποτελείται από έναν αριθμό εργασιών κι η συσχέτιση μεταξύ τους ορίζεται από το control flow. Το runtime σύστημα εγγυάται τη διατήρηση αυτών των συσχετίσεων κατά τη διάρκεια εκτέλεσης μίας ροής εργασίας.

Ο ορισμός μίας εργασίας της ροής εργασίας περιλαμβάνει την περιγραφή της απαιτούμενης εισόδου (π.χ. data sets και παράμετροι αρχικοποίησης) καθώς και της παραγόμενης εξόδου της. Μπορεί επίσης να περιλαμβάνει στοιχεία που αφορούν δικαιώματα εκτέλεσης για χρήστες ή ομάδες χρηστών. Επιπλέον στην περίπτωση που η εργασία αντιστοιχεί στην εκτέλεση κάποιου προγράμματος που είναι εγκατεστημένο σε κάποιον υπολογιστή τότε στην περιγραφή της εργασίας περιέχονται στοιχεία που αφορούν τη διεύθυνση του μηχανήματος, τη θέση του προγράμματος στο σύστημα αρχείων κλπ. Ένα data set μπορεί να είναι κάποιο αρχείο με επιστημονικά δεδομένα, εγγραφές σε βάση δεδομένων κλπ.

Το data flow περιγράφει τη μετακίνηση των data sets μεταξύ των διαφόρων εργασιών (π.χ. τη μετακίνηση ενός data set από το μηχάνημα προέλευσης προς ένα μηχάνημα όπου είναι εγκατεστημένο ένα πρόγραμμα που συμμετέχει στη ροή εργασίας και για το οποίο το data set αυτό αποτελεί είσοδο).

## 2.3 Διαστάσεις μίας Ροής Εργασίας

Η ιστορία εκτέλεσης μίας διαδικασίας είναι μία ακολουθία από τριάδες:

- ενέργεια
- χρήστης
- IT πόροι

Μπορούμε να μιλήσουμε για τρεις διαστάσεις που αφορούν μία ροή εργασίας.

### 2.3.1 Η λογική της διαδικασίας

Η πρώτη διάσταση αντιπροσωπεύει τη ‘λογική της διαδικασίας’. Η λογική της διαδικασίας περιγράφει τις ενέργειες που πρέπει να εκτελεστούν και την ακολουθία εκτέλεσής τους. Μια ενέργεια μπορεί να είναι χειρωνακτική (μη υποστηριζόμενη από υπολογιστή) ή αυτοματοποιημένη (υποστηριζόμενη από υπολογιστή). Αν είναι αυτοματοποιημένη μπορεί να είναι ένα πρόγραμμα ή μια άλλη ανάλογη διαδικασία. Η ροή εκτέλεσης από μία ενέργεια σε μία άλλη μπορεί να είναι ακολουθιακή (σειριακή) ή

παράλληλη. Αν είναι ακολουθιακή, τότε μία ενέργεια εκτελείται μετά από την ολοκλήρωση μίας άλλης. Αν είναι παράλληλη, τότε δύο ή περισσότερες ενέργειες εκτελούνται ταυτόχρονα, μειώνοντας με αυτό τον τρόπο το χρόνο εκτέλεσης της επιχειρησιακής διαδικασίας. Επίσης, η ροή εκτέλεσης μπορεί να περιλαμβάνει κάποιες συνθήκες. Αυτό σημαίνει ότι αναλόγως των αποτελεσμάτων μίας ενέργειας η πορεία εκτέλεσης μπορεί να εμφανίσει μία διακλάδωση είτε προς μία κατεύθυνση είτε προς μία άλλη.

### 2.3.2 Η οργανωτική διάσταση

Η δεύτερη διάσταση είναι η οργανωτική. Αυτή περιγράφει την οργανωτική δομή σε σχέση με τμήματα, ρόλους και άτομα. Παρέχει, δηλαδή, την πληροφορία που αφορά το θέμα του ποιος εκτελεί ποια/ες ενέργεια/ες. Εδώ μπορούν να εμφανιστούν περιπτώσεις όπου κάποια ενέργεια εκτελείται όχι απευθείας από κάποιο άτομο, αλλά από κάποια άλλη οντότητα για λογαριασμό όμως του ατόμου αυτού.

### 2.3.3 Η τεχνολογική υποδομή

Η τρίτη διάσταση μίας ροής εργασίας είναι αυτή της τεχνολογικής υποδομής που απαιτείται για την εκτέλεσή της (IT infrastructure). Σε αυτήν περιγράφονται ποιοι IT πόροι χρειάζονται για τη διεκπεραίωση μίας επιχειρησιακής εργασίας, π.χ. τα προγράμματα που εκτελούν μια ιδιαίτερη ενέργεια.

## 2.4 Ωφέλειες από τη χρήση Ροών Εργασίας

Η χρήση ροών εργασίας συνοδεύεται από σημαντικά πλεονεκτήματα στοιχείο το οποίο παρακινεί πολλούς οργανισμούς να υιοθετήσουν τη χρήση τους. Μεταξύ των πλεονεκτημάτων αυτών παρουσιάζουμε τα σημαντικότερα παρακάτω:

- Αύξηση της αποδοτικότητας που οδηγεί σε χαμηλότερο κόστος για τον οργανισμό καθώς και καλύτερη κατανομή του φόρτου εργασίας εντός του οργανισμού.

- Βελτιωμένος έλεγχος των διαδικασιών ο οποίος επιτυγχάνεται από την τυποποίησή τους.

- Βελτιωμένη ικανότητα διαχείρισης διαδικασιών. Τα προβλήματα απόδοσης μπορούν πλέον πολύ πιο εύκολα να εντοπισθούν και να γίνει κατανοητή η προέλευσή τους.

Έτσι, πολλοί οργανισμοί καταφεύγουν στη χρήση ροών εργασίας στην καθημερινή τους πρακτική κυρίως πιεζόμενοι από τη συνεχή ανάγκη για μείωση του κόστους τους και την αύξηση της ποιότητας και των εσωτερικών δυνατοτήτων τους. Με τη χρήση ροών εργασίας λειτουργίες όπως η πελατειακή υποστήριξη κι επεξεργασία παραγγελιών, εσωτερικές διαχειριστικές διαδικασίες κ.α. εκτελούνται αρκετά πιο αποδοτικά και με καλύτερο έλεγχο σε σχέση με τις παραδοσιακές τεχνικές.

## 2.5 Συστήματα Διαχείρισης Ροών Εργασίας

Ένα σύστημα διαχείρισης ροών εργασίας (Workflow Management System ή WfMS) είναι ένα σύστημα που ορίζει, δημιουργεί και διαχειρίζεται την εκτέλεση των ροών εργασίας μέσω της χρήσης εργαλείων λογισμικού που υλοποιούν μία ή περισσότερες μηχανές εκτέλεσης ροών εργασίας (workflow engines). Οι τελευταίες μπορούν να ερμηνεύσουν τον ορισμό μίας ροής εργασίας, να αλληλεπιδράσουν με συμμετέχοντες ροών εργασίας (όπως είναι οι χρήστες ή τα προγράμματα) και να χρησιμοποιήσουν IT πόρους. Αυτού του είδους τα συστήματα έχουν τη δυνατότητα να αξιοποιήσουν την υπάρχουσα πληροφοριακή και επικοινωνιακή υποδομή και να λειτουργήσουν σε ένα περιβάλλον που εκτείνεται από μικρές ομάδες εργασίας σε παγκόσμιες επιχειρήσεις. Παρόλο το εύρος στο οποίο εκτείνονται τα WfMSs έχουν να επιδείξουν συγκεκριμένα κοινά χαρακτηριστικά τα οποία και παρέχουν τη βάση της διαλειτουργικότητας μεταξύ διαφορετικών συστημάτων.

Ένα WFM συστήμα θα πρέπει να παρέχει υποστήριξη δύο βασικών λειτουργιών:

α) Λειτουργίες κατασκευής (build-time), οι οποίες σχετίζονται με τον ορισμό και τη μοντελοποίηση της διαδικασίας ροής εργασίας και των σχετιζόμενων ενεργειών.

β) Λειτουργίες εκτέλεσης (run-time), οι οποίες αφορούν την εκτέλεση των ροών εργασίας και τη διαχείρισή τους.

### 2.5.1 Build-time

Το build-time μέρος ενός συστήματος διαχείρισης ροών εργασίας παρέχει τις λειτουργίες και τις δυνατότητες του ορισμού, της δοκιμασίας και της διαχείρισης της πληροφορίας σχετιζόμενης με ροές εργασίας. Συνήθως, το build-time ενός συστήματος διαχείρισης ροών εργασίας παρέχει ένα γραφικό περιβάλλον εργασίας για τον ορισμό ροών εργασίας, ενώ μπορεί να χρησιμοποιεί και μία πρότυπη γλώσσα ορισμού ροών εργασίας.

### 2.5.2 Run-time

Η κύρια επιδίωξη του run-time μέρους ενός συστήματος διαχείρισης ροών εργασίας είναι η εκτέλεση των ροών εργασίας που του έχουν ανατεθεί κι η δυνατότητα παρακολούθησης και διαχείρισης των ροών εργασίας καθόλη τη διάρκεια της λειτουργίας του. Το run-time μέρος μπορεί κατά τη διάρκεια εκτέλεσης ροών εργασίας να αλληλεπιδρά με τους χρήστες.

Ο τύπος της δράσης ενός συστήματος εκτέλεσης/διαχείρισης ροών εργασίας για κάθε μία από τις ενέργειες εξαρτάται από τις υλοποιήσεις των αλληλο-συσχετιζόμενων εργασιών προς εκτέλεση. Οι εργασίες μπορεί να είναι αυτοματοποιημένες και να μην απαιτείται η παρέμαβαση του χρήστη ή αλλιώς να απαιτείται η παρέμβαση του χρήστη π.χ. για τη συμπλήρωση κάποιων παραμέτρων εκτέλεσης μίας εργασίας.

## 2.6 Υλοποιημένα Συστήματα Διαχείρισης Ροών Εργασίας

Η τεχνολογία των ροών εργασίας είναι ένας ευρύς ερευνητικός τομέας, που πραγματεύεται τη μοντελοποίηση και τον αυτοματισμό διαδικασιών οι οποίες περιλαμβάνουν ποικίλα έργα (tasks). Δυστυχώς, οι πιο πολλές υλοποιήσεις μέχρι τώρα

είναι κεντρικοποιημένες και υποφέρουν από έλλειψη αξιοπιστίας και δυνατότητα κλιμάκωσης.

### 2.6.1 Exotica και WIDE

Τα συστήματα Exotica/FMQM [3] και WIDE [4] παρουσιάζουν κατανεμημένες αρχιτεκτονικές για την ενεργοποίηση (enactment) ροών εργασίας και βασίζονται σε τεχνολογίες middleware. Η σχεδίαση του Exotica/FMQM αποτελείται από διαδοχικά επίπεδα (layers) πάνω από ένα σύστημα ουράς και υποστηρίζει τη σημασιολογία δοσοληψιών μέσω κλήσεων ενός API για την πρόσβαση στην ουρά. Ένας μεταγλωττιστής κατανέμει σε κάθε κόμβο ένα μέρος των προς εκτέλεση εργασιών. Κάθε κόμβος συγχρονίζεται με τους υπόλοιπους μέσω του δικτύου των ουρών.

Το WIDE είναι μια αρχιτεκτονική για τη διαχείριση ροών εργασίας βασιζόμενη στην CORBA [5]. Βασίζεται σε ένα σύστημα διαχείρισης ροών εργασίας με υποστήριξη ενεργών κανόνων (active rules). Κάθε μηχανή ροών εργασίας συντηρεί μία βάση δεδομένων από γεγονότα και περιλαμβάνει έναν δρομολογητή (scheduler) ο οποίος ταιριάζει γεγονότα με κανόνες. Οι επιλεγμένοι κανόνες καταγράφονται σε μια λίστα εκτέλεσης (to-execute), η οποία με τη σειρά της διαβάζεται κατά τακτά χρονικά διαστήματα (polling) από έναν ερμηνευτή κανόνων.

### 2.6.2 METEOR2

Το METEOR2 [6] έχει αναπτύξει δύο πρωτότυπα: το ORB Work, που βασίζεται στη CORBA και το WebWork, που βασίζεται ολοκληρωτικά στις τεχνολογίες διαδικτύου. Και τα δύο παραπάνω πρωτότυπα υποστηρίζουν τη δρομολόγηση κατανεμημένων ροών εργασίας. Ο προσδιορισμός της ροής εργασίας αποθηκεύεται σε μία μορφή η οποία περιλαμβάνει τις εξαρτήσεις προκατόχου-διαδόχου των έργων (tasks), τους ορισμούς των αντικειμένων των δεδομένων που μεταφέρονται μεταξύ έργων καθώς επίσης και τις πληροφορίες έναρξης λειτουργίας των έργων (task invocation information). Ένας γεννήτορας κώδικα παράγει οντότητες διαχείρισης έργων και wrappers για τις εργασίες που πρέπει να εκτελεστούν. Κάθε wrapper περιλαμβάνει έναν ενσωματωμένο (hard-wired) προσδιορισμό των προκατόχων και διαδόχων του έργου που διαχειρίζεται καθώς επίσης και κώδικα ο οποίος αποτιμά τη συνθήκη ενεργοποίησης του έργου.

Συγκεκριμένα, ο wrapper θέτει το έργο σε λειτουργία ενώ χειρίζεται και την περίπτωση ανάνηψης (recovery) από λάθη. Με την εγκατάσταση των οντοτήτων σε κάθε τόπο (site), οι χρήστες μπορούν να δημιουργήσουν περιπτώσεις διαδικασιών. Ένα κεντρικοποιημένο όργανο παρακολούθησης παρέχει περιορισμένη υποστήριξη για εντοπισμό (tracking) και παρακολούθηση (monitoring) της εκτέλεσης περιπτώσεων διαδικασιών. Όταν ο κώδικας του wrapper για έργα ενεργοποιηθεί, παίρνει την είσοδο του έργου και βάζει σε λειτουργία τον κώδικα του έργου στην κατάλληλη οντότητα επεξεργασίας. Μόλις τελειώσει προσδιορίζει την τελική κατάσταση του έργου, πακετάρει τα αντικείμενα δεδομένων της εξόδου κι ειδοποιεί τα διάδοχα έργα.

### 2.6.3 Middleware for Enterprise-Wide Workflow Management (MENTOR)

Το MENTOR [7] project στοχεύει στην ανάπτυξη ενός κλιμακώσιμου περιβάλλοντος υψηλής διαθεσιμότητας για την εκτέλεση, την παρακολούθηση και τον έλεγχο των ροών εργασίας. Η αρχιτεκτονική του είναι ανοικτή και αρθρωτή (modular). Επιπλέον, η αρχιτεκτονική του MENTOR επιτρέπει την παρακολούθηση των εκτελούμενων δοσοληψιών, ενώ παράλληλα χρησιμοποιεί μία υλοποίηση της CORBA που αντιμετωπίζει την ετερογένεια [8] των εφαρμογών που είναι σε λειτουργία. Επιπρόσθετες λειτουργίες που υποστηρίζονται είναι οι εξής:

- Εκτέλεση επερωτήσεων πάνω στην ιστορία των ροών εργασίας
- Διαλειτουργικότητα με ετερογενή συστημάτα διαχείρισης ροών εργασίας
- Έλεγχος της εγκυρότητας των προσδιορισμών ροών εργασίας

Η αρχιτεκτονική του MENTOR βασίζεται στην αρχιτεκτονική πελάτη-εξυπηρετητή (client-server). Σύμφωνα με αυτήν την αρχιτεκτονική μία ροή εργασίας εκτελείται από κατάλληλους εξυπηρετητές, ενώ οι εφαρμογές που τίθενται σε λειτουργία από διάφορες ενέργειες μίας ροής εργασίας εκτελούνται σε υπολογιστές του πελάτη (όπου οι εφαρμογές με τη σειρά τους μπορεί να εκδώσουν αιτήσεις για άλλους εξυπηρετητές, ανεξαρτήτως εάν η εκάστοτε εφαρμογή τίθεται σε λειτουργία μέσα από μία ροή εργασίας ή όχι). Στις εφαρμογές μεγάλων οργανισμών, οι ροές εργασίας μπορεί να χωριστούν σε πολλαπλά οργανωτικά μέρη τα οποία είναι συχνά σε μεγάλο βαθμό αυτόνομα.

Επομένως, για λόγους κλιμακωσιμότητας, ακολουθείται η προσέγγιση ότι μία μεγάλη ροή εργασίας μπορεί να διαμεριστεί σε ένα πεπερασμένο αριθμό από υπο-ροές εργασίας, κάθε μία από τις οποίες χειρίζεται ένας μόνο εξυπηρετητής.

### 2.6.4 WASA

Το WASA [9] είναι ένα ευέλικτο σύστημα διαχείρισης ροών εργασίας και παρουσιάζει τα εξής χαρακτηριστικά **:**

- Γραφικός ορισμός ροών εργασίας που αλληλεπιδρούν μεταξύ τους.
- Κατανεμημένη και αξιόπιστη εκτελέση ροών εργασίας.
- Αντικειμενοστραφής σχεδίαση μέσω της UML [10].
- Τυπικές υπηρεσίες middleware με τη βοήθεια της τεχνολογίας CORBA.

Το WASA παρουσιάζει μία αρχιτεκτονική τριών επιπέδων. Οι χρήστες προσπελαύνουν το σύστημα μέσω μιας γραφική διεπαφής χρήσης η οποία μπορεί να εκκινήσει εξωτερικά προγράμματα, τα οποία θα υλοποιήσουν τις ενέργειες της ροής εργασίας. Παραδείγματα εξωτερικών προγραμμάτων είναι εφαρμογές και συστήματα λογισμικού που υλοποιούν συγκεκριμένα έργα σε μία ροή εργασίας.

Η αρχιτεκτονική του WASA βασίζεται σε components που επιτρέπουν την επαναχρησιμοποίηση των ροών εργασίας σε διαφορετικές CORBA εφαρμογές. Το σημαντικότερο επίπεδο στην αρχιτεκτονική του WASA είναι το επίπεδο των facilities όπου βρίσκονται κι αντικείμενα ροών εργασίας. Στο χαμηλότερο επίπεδο, πρότυπες υπηρεσίες CorbaServices χρησιμοποιούνται για την κάλυψη των αναγκών των αξιόπιστων και αποδοτικά κατανεμημένων ροών εργασίας.

## 2.7 Γλώσσες και Μετα-γλώσσες Ορισμού Ροών Εργασίας

### 2.7.1 Ορισμός Γλώσσας Προσδιορισμού Ροών Εργασίας

Οι γλώσσες ορισμού ροών εργασίας είναι ειδικές γλώσσες που χρησιμοποιούνται για τον πλήρη ορισμό μίας ροής εργασίας. Ο ορισμός αυτός περιλαμβάνει:

- Ορισμό των ενεργειών που πρέπει να γίνουν σε κάθε πιθανή κατάσταση

- '*Πριν*' και '*μετά από*' συνθήκες των καταστάσεων.
- Μεταβάσεις μεταξύ καταστάσεων.
- Ορισμό της ακολουθίας των καταστάσεων/εργασιών.
- Ορισμό των αυτοματοποιημένων εργασιών και των εργασιών που απαιτούν είσοδο από το χρήστη.

Οι ενέργειες μπορεί να είναι τετριμμένες λειτουργίες όπως η μεταφορά ενός εγγράφου από ένα μέρος σε ένα άλλο ή ακόμη και πολύπλοκες εμπλέκοντας πολλές λειτουργίες. Οι 'πριν και μετά' συνθήκες μπορούν να χρησιμοποιηθούν για τον έλεγχο της ακεραιότητας του συστήματος ροών εργασίας. Μπορεί να υπάρχουν πολλές πιθανές μεταβάσεις από μία κατάσταση κι η επιλογή της μετάβασης να εξαρτάται είτε από τους αυτοματοποιημένους υπολογισμούς είτε από την είσοδο του χρήστη. Μία κατάσταση μπορεί να μοντελοποιηθεί έτσι ώστε να αποτελείται από πολλές υποκαταστάσεις οι οποίες να συμβαίνουν είτε ταυτόχρονα είτε ακολουθιακά. Αυτή η τεχνική μοντελοποίησης είναι χρήσιμη ειδικά για εκείνες τις περιπτώσεις όπου το σύστημα ροών εργασίας είναι αρκετά πολύπλοκο οπότε κι είναι δυνατό να παρουσιαστεί σε πολλά επίπεδα αφαίρεσης.

Η γλώσσα ορισμού ροών εργασίας ερμηνεύεται και εκτελείται από το λογισμικό ενεργοποίησης ροών εργασίας (workflow enactment software) το οποίο με τη σειρά του εκτελεί τις καθορισμένες εργασίες, χρησιμοποιεί άλλες εφαρμογές (όπως είναι ορισμένες από το σύστημα) και απαιτεί είσοδο από το χρήστη όποτε αυτή κρίνεται απαραίτητη.

### 2.7.2 Είδη Γλωσσών Προσδιορισμού Ροών Εργασίας

Για τον προσδιορισμό των διαδικασιών ροών εργασίας, διάφορες γλώσσες έχουν αναπτυχθεί κατά καιρούς. Συνήθως κάθε έργο χρησιμοποιεί έναν αποκλειστικό τρόπο προσδιορισμού μοντέλων ροών εργασίας και λίγες φορές χρησιμοποιούνται υπάρχουσες γλώσσες μοντελοποίησης διαδικασιών όπως είναι τα αρκετά γνωστά Petri-Nets [11], [12], [13], [14] για το κομμάτι της μοντελοποίησης του προϊόντος τους. Σύμφωνα με τον Carlsen [15] οι υπάρχουσες γλώσσες μοντελοποίησης των διαδικασιών ροών εργασίας μπορούν να ταξινομηθούν στις εξής πέντε κατηγορίες :

α) Input-Process-Output (IPO) βασιζόμενες γλώσσες, όπως είναι τα δίκτυα ενεργειών που χρησιμοποιούνται στο IBM MQSeries Workflow [16]. Αυτές οι γλώσσες περιγράφουν μια ροή εργασίας ως άκυκλο γράφο από ενέργειες υποδηλώνοντας την ακολουθία της εκτέλεσής τους.

β) Speech-Act προσεγγίσεις (ή αλλιώς Language Action) όπως χρησιμοποιείται στο προϊόν 'Action Technologies Action Workflow' [17]. Αυτού του είδους οι προσεγγίσεις μοντελοποιούν μία ροή εργασίας ως μια αλληλεπίδραση μεταξύ τουλάχιστον δύο συμμετεχόντων που ακολουθούν ένα δομημένο κύκλο συνομιλίας. Σε αυτήν ξεχωρίζουν οι φάσεις της διαπραγμάτευσης, της αποδοχής, της εκτέλεσης και της ανασκόπησης.

γ) Μέθοδοι μοντελοποίησης βασιζόμενοι σε περιορισμούς, όπως είναι η γραμματική 'Generalized Process Structure Grammar', που προτάθηκε από τον Glance [18]. Αυτού του είδους οι προσεγγίσεις περιγράφουν μία διαδικασία ως ένα σύνολο από περιορισμούς, αφήνοντας χώρο για ευλυγισία, ο οποίος διαφορετικά χάνεται από τις αντίστοιχες προσεγγίσεις του IPO και του Speech-Act. Αυτές οι επικείμενες γλώσσες μοντελοποίησης είναι βασισμένες σε κείμενο και μοιάζουν με τις παραδοσιακές γλώσσες προγραμματισμού. Οπότε, η γραφική αναπαράσταση αυτών των μοντέλων φαίνεται δύσκολη.

δ) Προσδιορισμοί διαδικασιών βάση του μοντέλου των ρόλων όπως είναι τα Role Activity Diagrams (RADs).

ε) Systems thinking and systems dynamic (learning organizations) [19].

### 2.7.3 Γλώσσες Προσδιορισμού Ροών Εργασίας

#### 2.7.3.1 IPL και WFSL

Η InterBase Parallel Language (IPL) χρησιμοποιείται για τον προσδιορισμό δοσοληψιών που υποστηρίζονται από το πρότυπο InterBase [20] το οποίο υποστηρίζει ένα ευέλικτο σύστημα δοσοληψιών που επιτρέπει την εκτέλεση δοσοληψιών πάνω από ετερογενή συστήματα βάσεων δεδομένων. Η γλώσσα ορίζει βάσει της επιθυμητής δοσοληψίας έναν

αριθμό υπο-δοσοληψιών με εξαρτήσεις ροών ελέγχου και δεδομένων. Ο προσδιορισμός περιλαμβάνει τις αποδεκτές εκβάσεις για κάθε δοσοληψία σε σχέση με τα αποτελέσματα των εκβάσεων των υπο-δοσοληψιών της εν λόγω δοσοληψίας.

Η WFSL είναι μια άλλη γλώσσα προσδιορισμού ροών εργασίας για τη μοντελοποίηση της ροής των διαδικασιών και τη γλώσσα προσδιορισμού έργων TSL [21], η οποία και παρέχει μία διεπαφή χρήσης για ετερογενή, αυτόνομα και κατανεμημένα συστήματα. Ο σχεδιασμός της ροής εργασίας διαχωρίζεται από τις λεπτομέρειες της αλληλεπίδρασης της με τις εφαρμογές. Η WFSL επιτρέπει μία δηλωτική περιγραφή της ροής ελέγχου και δεδομένων, βασιζόμενη σε κανόνες. Περιέχει επίσης ένα λεπτομερές μοντέλο εξάρτησης έργων που επιτρέπει τη διαχείριση της ατομικότητας και της συνέπειας των ροών εργασίας.

### 2.7.3.2 FDL

Υπάρχουν πολλά προϊόντα διαχείρισης ροών εργασίας που ειδικεύονται στην διαχείριση δομημένων διαδικασιών. Ένα από αυτά είναι το FlowMark [22] της IBM [23], το οποίο προσφέρει τη γλώσσα προσδιορισμού FDL για τον ορισμό μοντέλων διαδικασίας με βάση έναν κατευθυνόμενο ακυκλικό γράφο από ενέργειες. Αυτή η γλώσσα είναι από τις πλέον σύγχρονες στα εμπορικά προϊόντα ροών εργασίας. Οι κόμβοι στο γράφο της διαδικασίας αντιστοιχούν σε ενέργειες οι οποίες κατά το χρόνο εκτέλεσης ‘ζευγαρώνουν’ με πόρους. Οι πόροι μπορεί να είναι προγράμματα εφαρμογών ή χρήστες. Για κάθε ενέργεια η είσοδος αντιπροσωπεύει το πλαίσιο της ενέργειας. Το πραγματικό περιεχόμενο μίας περίπτωσης δεδομένης ενέργειας είναι ένα σύνολο από τιμές τυπικών παραμέτρων που καθορίστηκαν για τη συγκεκριμένη περίπτωση ενέργειας.

### 2.7.3.3 WIDE language

Η γλώσσα προσδιορισμού ροών εργασίας, που παρέχει το WIDE πρωτότυπο [24], υποστηρίζει:

- Περιγραφή γράφων διαδικασιών και μοντέλων διαδικασιών με βάση τις αναθέσεις δουλειάς για συγκεκριμένους ρόλους.

- Ανάλυση ρόλων για τον προσδιορισμό μέσα από ένα pool από διαθέσιμους και κατάλληλους εργάτες και πόρους την κατανομή κατά το χρόνο εκτέλεσης μίας διαδικασίας.

Τα παραπάνω γίνονται με τον ορισμό των προ-συνθηκών, των μετα-συνθηκών, των ενεργειών και των διαχειριστών εξαιρέσεων (exception handlers) για κάθε ενέργεια καθώς και με τον έλεγχο της ανάθεση μίας εργασίας σε έναν πράκτορα προκειμένου αυτή να εκτελεστεί. Ο προσδιορισμός μίας διαδικασίας περιλαμβάνει όχι μόνο τη ροή ελέγχου και τη διαχείριση λαθών μέσω κανόνων, αλλά και τη ροή των δεδομένων.

### 2.7.3.4 XRL

Η XRL [25] είναι ταυτόχρονα γλώσσα κι αρχιτεκτονική που υποστηρίζει τη δρομολόγηση ροών εργασίας για την παροχή διαδικτυακών υπηρεσιών. Το κύριο χαρακτηριστικό της είναι ότι παρέχει ένα μηχανισμό περιγραφής διαδικασιών στο επίπεδο των περιπτώσεων. Είναι βασισμένη στη τεχνολογία της XML κι η σημασιολογία της εκφράζεται με βάση τα Petri-nets. Η αρχιτεκτονική της βασίζεται στην έννοια του routing slip που ορίζει το σχήμα της δρομολόγησης. Η routing slip είναι μια ακολουθία από ρόλους ή χρήστες που πρέπει να αναθεωρήσουν έγγραφα τα οποία βρίσκονται σε μια συγκεκριμένη σειρά. Ένα ή παραπάνω έγγραφα μπορούν να προστεθούν, να αλλαχθούν ή να αφαιρεθούν από τη routing slip με την παρέμβαση υπαλλήλων. Ένας ιδιοκτήτης δημιουργεί τη routing slip κι ορίζει τους συνεργάτες δρομολόγησης. Κάθε slip έχει ένα μοναδικό ID που μπορεί να χρησιμοποιηθεί για την ανίχνευσή του. Το σχήμα δρομολόγησης της XRL γίνεται αποδεκτό σε έναν κόμβο μέσω ηλεκτρονικού μηνύματος. Αυτό επεξεργάζεται με τη βοήθεια ενός XML parser και αποθηκεύεται ως μία δομή δεδομένων. Ο πυρήνας της μηχανής ροών εργασίας διαβάζει αυτή τη δομή και δημιουργεί μία Petri-net αναπαράσταση. Βασιζόμενη σε αυτή την αναπαράσταση, η μηχανή προσδιορίζει το επόμενο/α βήμα/τα που πρέπει να εκτελεστεί/ουν και το/α παρουσιάζει στο χρήστη μέσω μίας διεπαφής χρήσης. Όταν όλα τα βήματα σε έναν κόμβο ολοκληρωθούν, η μηχανή ροών εργασίας αποθηκεύει την αναθεωρημένη δομή δεδομένων και τη στέλνει στον επόμενο κόμβο της ροής εργασίας.

### 2.7.3.5 Workflow Definition Language 2 (WDL2)

Η γλώσσα Workflow Definition Language 2 [26] είναι μία γλώσσα προσδιορισμού ροών εργασίας και χρησιμοποιείται στο σύστημα ροών εργασίας WorkMan [27], το οποίο είναι ένα project του πανεπιστημίου του Helsinki. Η WDL2 γλώσσα ορίζεται με τη βοήθεια της XML. Κάθε στοιχείο της ροής εργασίας ορίζεται με τη χρησιμοποίηση ενός XML αρχείου. Στη συνέχεια τα αρχεία συνενώνονται μέσω της μεθόδου XLink [28] που ορίζεται από το World Wide Web Consortium (W3C) [29]. Αυτή η δόμηση προωθεί τη διαρθρωτικότητα (modularity) και την αναχρησιμοποίηση των στοιχείων μίας ροής εργασίας. Η WDL2 περιέχει δομικά μέρη για υποθετικές δηλώσεις, ανακυκλώσεις, λογικούς τελεστές, υπολογισμούς και πέρασμα παραμέτρων.

### 2.7.4 Παραδείγματα Μετα-γλωσσών Προσδιορισμού Ροών Εργασίας

Στις επόμενες παραγράφους θα παρουσιαστούν πέντε διαφορετικές προσεγγίσεις μετα-γλωσσών για μοντελοποίηση διαδικασιών οι οποίες κι είναι οι εξής:

- Workflow Process Definition Language
- Process Interchange Framework
- Process Specification Language
- Generalized Process Structured Grammar
- Unified Modelling Language (UML)

#### 2.7.4.1 Workflow Process Definition Language

Η γλώσσα 'Workflow Process Definition Language' (WPDL) συστήθηκε από τον οργανισμό WFMC το 1994. Έκτοτε έχει εξελιχθεί αρκετά και μάλιστα έχει αλλάξει και φόρμα αναπαράστασης καθώς τώρα βασίζεται στη φόρμα XML και έχει μετονομαστεί σε 'XML Process Definition Language' [30].

Ο προσδιορισμός του WFMC κινείται γύρω από ένα μοντέλο αναφοράς (reference model) που περιγράφει τα βασικά στοιχεία ενός συστήματος διαχείρισης ροών εργασίας και εστιάζει στις διεπαφές χρήσης με εξωτερικά συστήματα. Το μοντέλο αυτό περιγράφει αποκλειστικά και μόνο εκείνα τα μέρη ενός συστήματος διαχείρισης ροών εργασίας που είναι υποψήφια για τη διαλειτουργικότητα με άλλα συστήματα λογισμικού. Οι διεπαφές χρήσης του μοντέλου αναφοράς συνδέονται με τις υπηρεσίες ενεργοποίησης ροών

εργασίας μέσω του Workflow Application Programming Interface (WAPI) [31]. Η WPDL δημιουργήθηκε ως μία μεταγλώσσα για την ανταλλαγή μοντέλων διαδικασίας ροών εργασίας του build-time. Η σχεδίαση της γλώσσας βασίζεται σε ένα ελάχιστο μετα-μοντέλο το οποίο ορίζει τα στοιχειώδη μέρη που πρέπει να υποστηριχθούν από ένα εργαλείο που διαβάζει ή/και γράφει σε WPDL. Αυτό το ελάχιστο μετα-μοντέλο μπορεί να επεκταθεί ανάλογα με τις ανάγκες μίας εφαρμογής.

Η κύρια έννοια της WPDL είναι η 'Workflow Process Definition' κι αποτελείται από μία ή παραπάνω οντότητες 'Workflow Process Activities'. Οι τύποι οντοτήτων της WPDL δεν είναι επεκτάσιμοι, αλλά ιδιότητες οριζόμενες από χρήστες μπορούν να προστεθούν σε απλούς τύπους οντοτήτων της WPDL. Επιπλέον, αναφορές σε εξωτερικές πηγές δεδομένων ως σημεία σύνδεσης μπορούν να υποδηλωθούν άμεσα, όπως είναι η αναφορά σε έναν εξωτερικό αποθηκευτικό χώρο ενός οργανισμού, σε συστήματα ή σε περιβαλλοντικά δεδομένα.

Για να επιβληθεί συμμόρφωση των συστημάτων διαχείρισης ροών εργασίας που ακολουθούν διαφορετικά παραδείγματα μοντελοποίησης, διάφορες κλάσεις συμμόρφωσης πρέπει να οριστούν. Αυτές οι κλάσεις περιορίζουν τον αριθμό των στοιχείων που πρέπει να αναδειχθούν από ένα σύστημα διαχείρισης ροών εργασίας για την υποστήριξη συμμόρφωσης στην WPDL. Ένας συνήθης περιορισμός είναι του τύπου block για συστήματα ροών εργασίας, ο οποίος απαιτεί κάθε διχοτόμηση (split) μίας διαδικασίας να ακολουθείται από μία παρόμοια ένωση (join) σε ένα μετέπειτα μέρος της διαδικασίας.

### 2.7.4.2 Process Interchange Framework (PIF)

Το πλαίσιο 'Process Interchange Framework' αναπτύχθηκε ως μια πρότυπη γλώσσα για διαδικασίες από το project 'MIT Process Handbook' [32]. Το 'Process Handbook' αποσκοπεί στη συλλογή αντιπροσωπευτικών επιχειρησιακών διαδικασιών και στην παρουσίαση αυτών ώστε να διευκολυνθεί η σύγκριση και η συλλογή εναλλακτικών διαδικασιών κάτω από πραγματικές επιχειρησιακές συνθήκες. Ο κύριος σκοπός του είναι η υποστήριξη αφενός των οργανισμών που αναζητούν ανασχεδίαση υπάρχουσων διαδικασιών και αφετέρου των νέων διαδικασιών που εμφανίζονται λόγω της τεχνολογικής ανάπτυξης.

Μέσα από την PIF προσέγγιση οι διαδικασίες αναπαριστώνται σε διάφορα επίπεδα αφαίρεσης. Οι δημιουργοί της PIF αναφέρουν ως το κυριότερο πλεονέκτημα της PIF το γεγονός ότι επιτρέπει στους χρήστες να απεικονίζουν άμεσα τις ομοιότητες και τις διαφορές μεταξύ συσχετιζόμενων διαδικασιών και επιπλέον να βρίσκουν εύκολα ή να παράγουν λογικές εναλλακτικές λύσεις του τρόπου εκτέλεσης μίας διαδικασίας.

Όλα τα δομικά μέρη της PIF προσδιορίζονται μέσω του Knowledge Interchange Format (KIF), μια γλώσσα που έχει σχεδιαστεί για την ανταλλαγή γνώσης μεταξύ διαφορετικών υπολογιστικών συστημάτων [33]. Η KIF επιτρέπει την επέκταση των υπάρχουσων εννοιών, στοιχείο σημαντικό για τη προσθήκη επεκτάσεων οριζόμενων στη γλώσσα PIF από το χρήστη. Επιπροσθέτως, η KIF είναι ένα προτεινόμενο πρότυπο και διαθέτει καλά ορισμένη τυπική σημασιολογία η οποία απλοποιεί τη διαδικασία ορισμού των δομικών μερών της PIF.

Μία περιγραφή διαδικασίας της PIF βασίζεται σε ένα σύνολο από ορισμούς πλαισίου (frame definitions). Κάθε ένας από τους ορισμούς πλαισίου υποδηλώνει έναν τύπο οντότητας που μπορεί να έχει περιπτώσεις (για παράδειγμα TIMEPOINT ή ACTIVITY). Αυτοί οι τύποι είναι τακτοποιημένοι σε μία ιεραρχία. Για κάθε τύπο της PIF αντιστοιχεί ένα σύνολο από προκαθορισμένες ιδιότητες οι οποίες ορίζουν διάφορα στοιχεία της περίπτωσης αυτού του τύπου. Οι ιδιότητες κληροδοτούνται από τους υπερ-τύπους στους τύπους και από τους τύπους στις περιπτώσεις τους.

Η PIF είναι μία ισχυρή πλατφόρμα ανταλλαγής ορισμών διαδικασιών. Λόγω του διαρθρωτικού (moduler) σχεδιασμού της μπορεί εύκολα να επεκταθεί ώστε να ανταποκρίνεται στις εκάστοτε ανάγκες της μοντελοποίησης διαδικασιών ροών εργασίας. Η ομάδα εργασίας της PIF ανταλλάσσει ιδέες με το WFMC για να γίνουν οι γλώσσες PIF και WPDL διαλειτουργικές κι άρα να προετοιμάσουν το έδαφος για μια ενιαία πλατφόρμα.

### 2.7.4.3 Process Specification Language (PSL)

Η γλώσσα PSL χρηματοδοτείται από το Ινστιτούτο 'National Institute of Standards and Technology' (NIST) [34]. Σκοπός αυτού του project είναι η ανάπτυξη μίας κοινής φόρμας ανταλλαγής για τις παραγωγικές επιχειρήσεις η οποία θα είναι ανεξάρτητη από υπάρχουσες εφαρμογές και αρκετά ισχυρή ώστε να αναπαριστά τις απαραίτητες πληροφορίες μίας διαδικασίας για κάθε είδος εφαρμογής. Ο τελικός στόχος είναι η υποστήριξη της επικοινωνίας μεταξύ διαφορετικών εφαρμογών που θα βασίζεται σε μία κοινή αντίληψη του περιβάλλοντός τους.

Η κύρια έννοια της PSL για την αντιστοίχιση (mapping) μεταξύ δύο προγραμμάτων εφαρμογών είναι αρχικά η αντιστοίχιση της οντολογίας μοντελοποίησης της κάθε εφαρμογής στην οντολογία της PSL. Τα βασικά μέρη της PSL είναι τα εξής **:**

- **Ενέργειες:** Αυτές μπορεί να είναι χαρακτηριστικές ενέργειες, όπως deterministic ή η non-deterministic διαδικασίες, καθώς επίσης μπορεί να είναι και ordering functions π.χ. creates, precedes.

- **Αντικείμενα:** Μπορεί να είναι πόροι π.χ. άνθρωποι, μηχανές ή καταστάσεις π.χ. pre-activity, post-activity καταστάσεις.

- **Σημεία χρονισμού:** Αυτά μπορεί να χρησιμοποιηθούν για την περιγραφή είτε χρονικών σχέσεων μεταξύ ενεργειών είτε της διάρκειας των διαδικασιών.

Όλες οι έννοιες της PSL προσδιορίζονται από την KIF, όπως γίνεται και με την PIF. Τελικά, τα projects PSL και PIF προβλέπεται να ενωθούν. Επίσης, έχει δημιουργηθεί και η αντιστοίχιση μεταξύ της PSL και της XML με το όνομα XPSL, αλλά η διαδικασία αυτή βρίσκεται στα πρώτα στάδια ανάπτυξής της.

### 2.7.4.4 Generalized Process Structure Grammars (GSPG)

Αν κι οι γλώσσες WPDL, PIF και PSL αναπαριστούν γλώσσες μοντελοποίησης διαδικασιών βασιζόμενες στην κατηγορία IPO, η γραμματική 'Generalized Process Structure Grammars' (GSPG), όπως προτείνεται από το project 'GLANCE' [35]**,**

αναπαριστά μία προσέγγιση βασισμένη σε περιορισμούς για τη μοντελοποίηση των διαδικασιών. Αναλυτικότερα, προκειμένου να μοντελοποιηθεί μία διαδικασία κατασκευάζεται μία γραμματική η οποία και θα περιέχει όχι μόνο τα νόμιμα στοιχεία της διαδικασίας, αλλά και τις συσχετίσεις μεταξύ αυτών. Η γραμματική 'στενεύει' το χώρο της διαδικασίας, ώστε να περιέχει μόνο τους ζωτικούς περιορισμούς και τους κανόνες κατασκευής, ενώ κάθε τι που δεν περιορίζεται εξαρχής θεωρείται μεταβλητό κατά το χρόνο της ενεργοποίησης της διαδικασίας (process enactment). Μία περίπτωση διαδικασίας στην GSPG δύναται να αποτελέσει μία νόμιμη φράση που κατασκευάζεται με τη χρησιμοποίηση της γραμματικής του μοντέλου της εν λόγω διαδικασίας. Κάθε γραμματική GSPG περιέχει δύο είδη κανόνων**:**

- Activity-centric rules, οι οποίοι διαχωρίζουν το στόχο μίας διαδικασίας σε υπο-στόχους και επισυνάπτουν περιορισμούς εκτέλεσης.

- Document-centric rules, οι οποίοι περιγράφουν τα αντικείμενα δεδομένων που χρησιμοποιούνται σε μία διαδικασία.

Η εφαρμογή της GSPG στη μοντελοποίηση διαδικασιών μπορεί να διευκρινιστεί με ένα απλό παράδειγμα.

Η ακολουθία δύο ενεργειών σε μία IPO-based γλώσσα ορισμού διαδικασιών, συνήθως, υποδηλώνεται ως εξής:

**B.start := A.end**

Μία GPSG-based γλώσσα μοντελοποίησης θα περιείχε τους εξής περιορισμούς:

B.start = A.end
B.end < deadline
B.start = B.end – B.average_duration

Οι GSPG-based ορισμοί διαδικασιών επιτρέπουν περισσότερη ευελιξία κατά τη διάρκεια της εκτέλεσης μίας διαδικασίας. Το γεγονός αυτό οφείλεται στο ότι όταν οι διαδικασίες

εκτελούνται δεν ακολουθούν ένα αυστηρά καθορισμένο σύνολο από μονοπάτια ροής ελέγχου και καταστάσεων, αλλά αναδύονται μέσα από το χώρο των διαδικασιών που ορίζονται από την αντίστοιχη γραμματική.

### 2.7.4.5 Unified Modelling Language (UML*)*

Αντίθετα με τις τέσσερις προηγούμενες προσεγγίσεις οι οποίες εστιάζουν σε μια ορισμένη περιγραφή μέσω ενός κειμένου των διαδικασιών, η γλώσσα 'Unified Modelling Language' (UML) ορίζει τέσσερα διαφορετικά είδη διαγραμμάτων για τη σχεδίαση αντικειμενοστραφών συστημάτων λογισμικού [36], [37], [38]. Η UML σχεδιάστηκε με τη χρήση ενός περιορισμένου αριθμού βασικών εννοιών που εν μέρει προϋπήρχαν προτού αυτή τυποποιηθεί το 1997. Οι έννοιες της UML μπορούν να επεκταθούν ή να ειδικευτούν από τους χρήστες. Επομένως, η UML επιτρέπει τον ορισμό πολύπλοκων συστημάτων κι είναι ανεξάρτητη του πεδίου εφαρμογής. Οι διαφορετικοί τύποι διαγραμμάτων της UML οδηγούν στη σχεδίαση διαφορετικών όψεων ενός συστήματος.

Κατά τη διάρκεια των τελευταίων δεκαετιών το ενδιαφέρον για χρήση αντικειμενοστραφούς σχεδιασμού ως μέσου παροχής επεκτάσιμων, κλιμακώσιμων και κατανεμημένων συστημάτων διαχείρισης ροών εργασίας έχει αυξηθεί σημαντικά [39]. Παράλληλα, ένας μεγάλος αριθμός από πρωτότυπα συστήματα, όπως το WASA2, έχουν αναπτυχθεί. Αρχίζοντας από την εσωτερική αρχιτεκτονική αυτών των συστημάτων η ανάγκη για μία αντικειμενοστραφή μέθοδο μοντελοποίησης των μοντέλων ροών εργασίας οδήγησε στην αξιολόγηση της UML ως μίας ακόμη γλώσσας μοντελοποίησης [40]. Αντίθετα με τις άλλες γλώσσες μοντελοποίησης που έχουν προαναφερθεί, η UML προσφέρει γραφικές δηλώσεις (notations) των μοντέλων ροών εργασίας. Όμως, δεν υπάρχει μία απλή δήλωση η οποία να μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση όλων των πλευρών ενός μοντέλου ροής εργασίας. Επιπλέον, αρκετοί τύποι διαγραμμάτων πρέπει να χρησιμοποιηθούν για να μοντελοποιήσουν όλες τις πλευρές μίας διαδικασίας ροής εργασίας. Αυτοί οι τύποι διαγραμμάτων είναι οι εξής**:**

- **Use case diagrams:** Τα διαγράμματα εδώ χρησιμοποιούνται για να απεικονίσουν την αλληλεπίδραση ενός συστήματος με το περιβάλλον του. Στην περίπτωση μίας διαδικασίας ροής εργασίας οι υποθέσεις χρήσης μπορεί να χρησιμοποιηθούν για

τη μοντελοποίηση της αλληλεπίδρασης μίας διαδικασίας ή μίας ενέργειας με εξωτερικούς παράγοντες-ενεργούντες (συμμετέχοντες σε ροή εργασίας, πελάτες, εξωτερικά συστήματα κ.λ.π.). Οι υποθέσεις χρήσης υποδηλώνουν μόνο τη στατική σχέση μεταξύ των ενεργούντων και της λειτουργικότητας του συστήματος. Επιπλέον, δεν περιγράφουν τη χρονική ή λογική ακολουθία των βημάτων μίας διαδικασίας. Η αποσύνθεση μίας διαδικασίας σε αρκετές υπο-διαδικασίες ή στοιχειώδεις ενέργειες μπορεί να μοντελοποιηθεί με τη χρησιμοποίηση της σχέσης 'χρήσεων' μεταξύ των υποθέσεων χρήσης.

- **Sequence diagrams:** Τα διαγράμματα απεικονίζουν τη χρονική και λογική σειρά των ενεργειών και των εμπλεκόμενων συμμετεχόντων σε μία δήλωση της μορφής 'swim-lane'. Αν οι διαφορετικοί ενεργούντες μέσα σε μία διαδικασία τακτοποιηθούν σε παράλληλες λωρίδες (lanes), η αλληλεπίδραση μεταξύ των συμμετεχόντων επιτρέπει μία αντιστοίχιση σε μοντέλα ροών εργασίας speech-act ή/και σε IPO-style.

- **Collaboration diagrams (διαγράμματα συνεργασίας):** Μέσα από τα διαγράμματα συνεργασίας περιγράφεται η αλληλεπίδραση μεταξύ των ενεργούντων και των υποθέσεων χρήσης με βάση τα μηνύματα που στέλνονται μεταξύ των διαφορετικών στοιχείων του διαγράμματος. Αυτού του είδους τα διαγράμματα μπορούν να ειδωθούν ως μία επέκταση των use case diagrams, αφού επιτρέπουν τη διάταξη των μηνυμάτων και των κατευθυνόμενων σχέσεων.

- **State-chart diagrams (διαγράμματα καταστατικού χάρτη):** Ένα διάγραμμα καταστατικού χάρτη δείχνει όλες τις δυνατές καταστάσεις μίας υπόθεσης χρήσης και επιπλέον τις μεταβάσεις αυτών των καταστάσεων. Αν ένα τέτοιου είδους διάγραμμα ενταχθεί σε ένα πλαίσιο διαχείρισης ροής εργασίας τότε μπορεί να χρησιμοποιηθεί για την απεικόνιση των πιθανών σημείων έναρξης και τερματισμού ενός μοντέλου ροής εργασίας, καθώς επίσης και των νόμιμων μεταβάσεων μεταξύ καταστάσεων (π.χ. έτοιμη, αρχικοποιημένη, ανεσταλμένη κ.τ.λ.).

- **Activity charts (διαγράμματα ενεργειών):** Τα διαγράμματα ενεργειών είναι παραλλαγές των διαγραμμάτων καταστατικών χαρτών και παρουσιάζουν γραφικά όλα τα πιθανά μονοπάτια των δράσεων (acts) μεταξύ ενεργειών. Ενώ οι καταστατικοί χάρτες μπορεί να περιέχουν παθητικές καταστάσεις, τα διαγράμματα ενεργειών απεικονίζουν τις σχέσεις μεταξύ ενεργειών. Η μετάβαση μεταξύ δύο ενεργειών είναι ενεργή μόνο όταν η προηγούμενη ενέργεια έχει τελειώσει και ένας προαιρετικός περιορισμός φρούρησης (optional guard constraint) στη μετάβαση πάρει αληθή τιμή. Τα στοιχεία μοντελοποίησης επιτρέπουν παράλληλες διακλαδώσεις, καθώς επίσης κι εναλλακτικές πορείες ανάμεσα σε ενέργειες. Αν και η αρχική έκδοση της UML οδηγεί σε πολύπλοκα διαγράμματα για εναλλακτικά μονοπάτια ή για υποθετικές διακλαδώσεις, η νέα έκδοση της UML υποστηρίζει μία πιο σύντομη δήλωση αυτών των υποθέσεων-περιπτώσεων.

Συνολικά, η UML προσφέρει μεγάλη ποικιλία από τύπους διαγραμμάτων και μπορεί να ανταποκριθεί στους διάφορους στόχους της μοντελοποίησης ροών εργασίας. Ωστόσο, η αναγκαιότητα για χρήση διαφορετικών τύπων διαγραμμάτων, που δεν είναι πάντοτε ορθογώνια (orthogonal) μεταξύ τους, σε συνδυασμό με την έλλειψη της μοντελοποίησης πόρων μετατρέπουν την τρέχουσα έκδοση της UML δύσκολη στη χρήση για τα υπάρχοντα εργαλεία ροών εργασίας. Πιθανώς, οι νέες εκδόσεις να προσφέρουν βελτιώσεις προς αυτήν την κατεύθυνση.

## 2.8 Διαλειτουργικότητα συστημάτων ροών εργασίας

Τα συστήματα ροών εργασίας μοιράζονται και κατανέμουν εργασία και συνεπώς πληροφορία και ευθύνη μεταξύ διαφορετικών εργατών (workers). Όμως, αυτά τα συστήματα συνήθως λειτουργούν στα πλαίσια ενός μοναδικού οργανισμού με ένα μεμονωμένο σύνολο από ενδιαφέροντα (τα οποία όλοι οι εργάτες δεσμεύονται να μοιραστούν), καθώς και σε ένα μοναδικό υπολογιστικό σύστημα. Τελευταία παρουσιάζεται η τάση δύο ή περισσότεροι οργανισμοί ή πρόσωπα, καθένα με τα δικά του ενδιαφέροντα, νομικά δικαιώματα, λεξιλόγιο, πρωτόκολλα κι υπολογιστικά συστήματα, να επιθυμούν να συνεργαστούν και να είναι σε θέση να μοιραστούν μία μοναδική

διαδικασία. Πολλές φορές οι οργανισμοί υπό συνεργασία μπορεί να χρησιμοποιούν διαφορετικά προϊόντα ροών εργασίας, τα οποία είναι ποικίλλα στη φύση τους και με πολλές διαβαθμίσεις, Εδώ πλέον τίθεται το πρόβλημα της διαλειτουργικότητας των συστημάτων ροών εργασίας. Πώς κάποιος, δηλαδή, θα συνενώσει διαφορετικά συστήματα ροών εργασίας, παραγόμενα από διαφορετικούς κατασκευαστές, έτσι ώστε να λειτουργούν συνεργατικά.

### 2.8.1 Η βάση του προβλήματος κι η λύση των πρακτόρων

Βάση του προβλήματος της ελλειπούς διαλειτουργικότητας των συστημάτων ροών εργασίας αποτελεί η υπόθεση ότι τα συστήματα αυτά κι οι πράκτορές τους δεν μοιράζονται μία συνεπή σημασιολογική αντίληψη της πληροφορίας η οποία συναλλάσσεται. Επιπλέον, δεν υπάρχει αμοιβαία κατανόηση του τρόπου συντονισμού των διάφορων ενεργειών μίας ροής εργασίας. Κατά τη συγγραφή λογισμικού οι προγραμματιστές δε λαμβάνουν πάντα υπόψιν τους την ετερογένεια των διαφόρων συστημάτων ροών εργασίας διατηρώντας με αυτόν τον τρόπο το πρόβλημα της μη διαλειτουργικότητας μεταξύ των συστημάτων αυτών.

Ο οργανισμός Workflow Management Coalition [1] προκειμένου να ενισχύσει την προσπάθεια για επίτευξη διαλειτουργικότητας επιβάλλει τρεις απαιτήσεις στο περιβάλλον των πρακτόρων οι οποίες είναι οι εξής**:**

α) Πρέπει να υπάρχει μία κοινή αντίληψη των βασικών εννοιών του πεδίου εφαρμογής. Όλη η επικοινωνία ανάμεσα στους πράκτορες πρέπει να βασίζεται σε αυτή την κοινή κατανόηση.

β) Το σύστημα πρέπει να υποστηρίζει ένα βασικό ρεπερτόριο δια-πρακτορικής επικοινωνίας.

γ) Η αλληλεπίδραση ανάμεσα στους πράκτορες πρέπει να συντονιστεί κάθε φορά που θα πρέπει να εκπληρώσουν από κοινού μία ομαδική ενέργεια.

Η ανάγκη για διασφάλιση κοινής αντίληψης της διαμοιραζόμενης πληροφορίας αντιμετωπίζεται αφενός με την ανάπτυξη των οντολογιών, που είναι ανεξάρτητες του έργου (task-independent) κι αναπαριστούν τα αντικείμενα του πεδίου εφαρμογής κι αφετέρου με την ανάπτυξη των εργασιακών οντολογιών (task ontologies) που είναι ανεξάρτητες του πεδίου εφαρμογής και περιγράφουν μεθόδους λύσης προβλημάτων με ένα γενικότερο τρόπο. Η ανάγκη για κοινή κατανόηση της δια-πρακτορικής επικοινωνίας αντιμετωπίζεται με τον ορισμό μιας γλώσσας επικοινωνίας πρακτόρων που περιγράφει ένα ουσιώδες σύνολο από πράξεις επικοινωνίας.

### 2.8.2 Χρήση Ορισμού Διαδικασίας Κατά Μήκος Πολλαπλών Πεδίων Εφαρμογής

Ένα σημαντικό στοιχείο για τη διαλειτουργικότητα δύο υπηρεσιών εκτέλεσης ροών εργασίας αποτελεί το να είναι αυτές σε θέση να ερμηνεύσουν ένα κοινό ορισμό διαδικασίας, ο οποίος για παράδειγμα έχει παραχθεί από ένα κοινό εργαλείο κατασκευής (build tool). Σε αυτήν την περίπτωση και τα δύο περιβάλλοντα καθίστανται ικανά να μοιραστούν μία μοναδική όψη των αντικειμένων του ορισμού της διαδικασίας, καθώς και τις ιδιότητες αυτών. Επίσης, οι ανεξάρτητες μηχανές ροών εργασίας καθίστανται ικανές να μεταφέρουν την εκτέλεση των ενεργειών ή των υπό-διαδικασιών σε ετερογενής μηχανές ροών εργασίας μέσα από το πλαίσιο ενός κοινού μοντέλου ονομασίας κι αντικειμένων (naming and object model).

Όταν αυτή η κοινή επιδίωξη του ορισμού διαδικασίας δεν είναι εφικτή, τότε η εναλλακτική προσέγγιση της ‘εξαγωγής’ λεπτομερειών ενός υποσυνόλου του ορισμού διαδικασίας, ως μέρος της runtime ανταλλαγής, μπορεί να καθίσταται πιθανή.

Επιπλέον, όταν η ανταλλαγή ορισμού διαδικασίας μέσω και των δύο παραπάνω προσεγγίσεων δεν είναι εφικτή τότε η διαλειτουργικότητα περιορίζεται σε μία προσέγγιση ανοίγματος πύλης (gateway approach), κατά την οποία τα ονόματα αντικειμένων κι ιδιοτήτων αντιστοιχίζονται μεταξύ των δύο περιβαλλόντων μέσω μίας πύλης που διασυνδέει τα διαφορετικά συστήματα. Σε αυτήν την περίπτωση τα δύο διαφορετικά συστήματα χρησιμοποιούν τις δικές τους φόρμες ορισμού διαδικασίας κι η πύλη χειρίζεται οποιαδήποτε αντιστοίχιση μεταξύ των δύο.

#### 2.8.2.1 Αλληλεπιδράσεις Runtime Ελέγχου

Το WFMC ορίζει το WAPI (Workflow Client Application Programming Interface), προκειμένου να επιτρέψει την υλοποίηση των front-end εφαρμογών οι οποίες χρειάζονται για την πρόσβαση στις λειτουργίες της μηχανής διαχείρισης ροών εργασίας. Κατά το χρόνο εκτέλεσης (runtime), τα καλέσματα του WAPI χρησιμοποιούνται για τη μεταφορά του ελέγχου μεταξύ των υπηρεσιών ροών εργασίας έτσι ώστε να ενεργοποιηθούν υπό-διαδικασίες ή ανεξάρτητες ενέργειες σε μία συγκεκριμένη υπηρεσία. Όταν κι οι δύο υπηρεσίες υποστηρίζουν ταυτόχρονα ένα κοινό επίπεδο από καλέσματα του WAPI και μία κοινή όψη των αντικειμένων του ορισμού διαδικασίας (συμπεριλαμβανομένων των ονοματικών συμβάσεων και των δεδομένων που σχετίζονται με την εφαρμογή ή με τη ροή εργασίας), τότε το παραπάνω θα γίνεται άμεσα από τις μηχανές ροών εργασίας – αν κι αυτό απαιτεί συμφωνία για ένα κοινό πρωτόκολλο υποστήριξης των στοιχειωδών WAPI μερών.

Όταν η προηγούμενη προϋπόθεση δεν είναι δυνατή, τα καλέσματα του WAPI μπορούν να χρησιμοποιηθούν για τη κατασκευή μίας υπηρεσίας πύλης, παρέχοντας με αυτόν το τρόπο διαλειτουργικότητα μεταξύ των δύο υπηρεσιών ροών εργασίας, η οποία και επιτυγχάνεται αρχικά με την αντιστοίχιση των διαφορετικών αντικειμένων κι όψεων δεδομένων μεταξύ των δύο περιβαλλόντων κι ακολούθως με την υποστήριξη διαφορετικών πρωτοκόλλων για τις διάφορες υπηρεσίες ροών εργασίας.

# Κεφάλαιο 3ο: ARION

## 3.1 Εισαγωγή

Το έργο ARION φιλοδοξεί να παρέχει μία νέα γενιά υπηρεσιών Ψηφιακής Βιβλιοθήκης για την αναζήτηση κι ανάκτηση επιστημονικών ψηφιακών συλλογών που βρίσκονται σε ερευνητικούς και συμβουλευτικούς οργανισμούς. Οι συλλογές αυτές περιέχουν δεδομένα, προγράμματα κι εργαλεία για ποικίλες επιστημονικές περιοχές και συνδυάζουν διαφορετικές περιοχές γνώσης. Ο ARION αναβαθμίζει τα αποτελέσματα μελετών σε περιοχές όπως η διαχείριση επιστημονικών αποθηκών και του metacomputing. Ενισχύει τη δουλειά για την ανάπτυξη ενός διεθνούς interoperability standard με σκοπό την ανάπτυξη ενός συστήματος που δρα συμπληρωματικά προς στις καθιερωμένες επιστημονικές πρακτικές σε αυτούς τους οργανισμούς. Ο ARION είναι ένα ανοιχτό σύστημα κι αναπτύσσεται σε συνεργασία με εθνικούς data providers, επιστημονικούς ερευνητές και SME's για να διασφαλιστεί ότι το έργο καλύπτει τις ανάγκες τους.

Μεγάλα κομμάτια της πολύτιμης επιστημονικής μας κληρονομιάς βρίσκεται σε συλλογές (repositories) οι οποίες αποθηκεύουν πόρους όπως: data sets, προγράμματα κι εργαλεία σχετικά με επιστημονικές εργασίες. Τα data sets μπορεί να είναι δεδομένα μετρήσεων ή δεδομένα τα οποία έχουν προκύψει από κάποιους επιστημονικούς υπολογισμούς. Τα προγράμματα είναι συνήθως μοντέλα προσομοιώσεων/προβλέψεων και τα εργαλεία είναι επίσης προγράμματα κατάλληλα για την παρουσίαση δεδομένων (στατιστικών, γραφικών κ.α) και visualization.

Εδώ και πολύ καιρό τα επιστημονικά δεδομένα και προγράμματα έχουν θεωρηθεί ως 'ιδιωτικοί' πόροι που χρησιμοποιούνται μόνο από τα άτομα/οργανισμούς που τα δημιούργησαν ή τα ανέπτυξαν. Αυτή η 'ιδιωτική ιδιοκτησία' συχνά προκύπτει από τη μη χρήση τους παρά από εφαρμοζόμενη πολιτική περιορισμού της χρήσης τους. Δεδομένα και μοντέλα συλλέγονται κι αναπτύσσονται ως μέρος μίας επιστημονικής μελέτης, αλλά η μετέπειτα χρήση τους δεν είναι προφανές ποια θα είναι. Με την απουσία κατάλληλης υποδομής πολύτιμα και μοναδικά επιστημονικά δεδομένα και μοντέλα 'χάνονται', καθώς

διατηρούνται τοπικά χωρίς κάποια ιδιαίτερη καταχώρησή τους σε κάποιον κατάλογο που θα βοηθούσε στην επαναχρησιμοποίησή τους.

Τεχνικές βασισμένες στη χρήση της υποδομής του Internet έχουν αναπτυχθεί προκειμένου να επιτρέψουν την πρόσβαση σε τέτοιους επιστημονικούς πόρους στην ευρύτερη επιστημονική κοινότητα βελτιώνοντας την υπάρχουσα κατάσταση. Όμως, ακόμα και state of the art συστήματα συνήθως παρουσιάζουν τέσσερα βασικά προβλήματα τα οποία και τα καθιστούν μη ελκύστικα τόσο στους data providers όσο και στους χρήστες.

α) Ο μηχανισμός δημοσίευσης των πόρων παραμένει πολύπλοκος καθώς απαιτεί προγραμματιστική προσπάθεια κι ικανότητες που συνήθως απουσιάζουν από τους data providers.

β) Στους χρήστες συχνά παρέχεται μία απλή διεπαφή χρήσης για την εκτέλεση της αναζήτησης με λίγη καθοδήγηση σχετικά με το πώς θα εντοπίσουν ή θα δημιουργήσουν επιστημονική πληροφορία.

γ) Όταν εντοπιστεί ένα πόρος υπάρχει μικρή υποστήριξη ως προς την ευέλικτη επαναχρησιμοποίησή του, π.χ. κάποιος μπορεί να πάρει/χρησιμοποιήσει τον πόρο είτε ως έχει είτε καθόλου. Συνεπώς, ο δυναμικός συνδυασμός διαφόρων πόρων που ανήκουν σε διαφορετικούς providers είναι σχεδόν αδύνατη.

δ) Οι ήδη υπάρχουσες λύσεις δεν είναι συμβατές με τις υπάρχουσες πρακτικές και τρόπους χρηματοδότησης που χρησιμοποιούνται από τους οργανισμούς που παράγουν δεδομένα κάτι που τις καθιστά μάλλον εμπόδιο παρά βοήθεια.

Ο στόχος του ARION είναι να ξεπεράσει την υπάρχουσα έλλειψη προσβασιμότητας στις ποικίλες πηγές επιστημονικών συλλογών δεδομένων και να προσφέρει μία νέα προσέγγιση στην προώθηση της επαναχρησιμοποίησής τους. Αναπτύσσει μία lightweight αρχιτεκτονική βασισμένη στην τεχνολογία του Internet η οποία θα προσφέρει στους επιστήμονες και τους πολίτες μεγαλύτερη πρόσβαση στη δημιουργία χρήσιμης πληροφορίας από προηγούμενες συλλογές δεδομένων και μοντέλων από διάφορες επιστημονικές περιοχές.

## 3.2 Οι στόχοι του ARION

Το βασικό χαρακτηριστικό του ARION (http://arion-dl.org) είναι ότι καθιστά δυνατή τη δυναμική παραγωγή επιστημονικής πληροφορίας, κάτι που είναι πρωτόγνωρο στο πλαίσιο των ψηφιακών βιβλιοθηκών (Digital Libraries ή DL). Σε αντίθεση με τα παραδοσιακά DL αντικείμενα, τα επιστημονικά αντικείμενα είναι αρκετά πολύπλοκα ως προς την επαναχρησιμοποίησή τους εκτός κι αν ακριβής πληροφορία αποθηκεύεται μέσα στην DL σχετικά με τη χρήση τους. Ο σκοπός της DL επιστημονικών συλλογών που φιλοδοξεί να αναπτύξει ο ARION μπορεί να αναλυθεί σε τρεις άξονες:

1. **Ανάπτυξη μίας ψηφιακής βιβλιοθήκης επιστημονικών συλλογών**

α) Ανάπτυξη οντολογιών για συγκεκριμένα domains επιστημονικής γνώσης. Οι οντολογίες θα είναι ανοιχτές και θα εξελίσονται με την υποστήριξη ενός web-based editor οντολογιών που θα επιτρέπει στους ειδικούς να εισάγουν νέα στοιχεία στην οντολογία

β) Ανάπτυξη μίας ιεραρχίας για τα μεταδεδομένα (metadata hierarchy) βασισμένη σε μία επεκτάσιμη αρχιτεκτονική που θα ενσωματώνει δυναμικά αναπτυσσόμενα λεξικά/οντολογίες η οποία θα μπορεί να χρησιμοποιηθεί ως ένα navigation tool προσθετικά στη λειτουργία της μηχανής αναζήτησης μεταδεδομένων (metadata search engine).

γ) Αναζήτηση κι όπου είναι αναγκαίο ανάπτυξη τεχνικών συσχέτισης μεταδεδομένων με δεδομένα. Θα αναπτυχθούν τεχνικές για τη συσχέτιση των μεταδεδομένων με τα δεδομένα που βρίσκονται σε ετερογενής πηγές, έτσι ώστε με την ανάκτηση μίας εγγραφής μεταδεδομένων να μπορεί να διασφαλιστεί η ανάκτηση των συσχετισμένων δεδομένων.

δ) Υλοποίηση δυνατότητας δυναμικής ανεύρεσης πόρων για την περίπτωση όπου τα δεδομένα δεν είναι άμεσα διαθέσιμα, αλλά μπορούν να παραχθούν δυναμικά μέσω της εκτέλεσης μίας ροής εργασίας που περιλαμβάνει υπάρχοντα προγράμματα και δεδομένα.

2. **Ανάπτυξη νέων τεχνικών και μεθόδων απαραίτητων για τη διαλειτουργικότητα των δεδομένων και των λειτουργιών της ψηφιακής βιβλιοθήκης:**

α) Ανάπτυξη διαδικασίας δημοσίευσης πόρων (Resource Export Procedure). Οι επιστήμονες/παροχείς πόρων θα είναι σε θέση να δημοσιεύουν τα δεδομένα και τα προγράμματά τους με ευκολία μέσω έξυπνων 'export and publish' wizards. Όλες οι τεχνικές λεπτομέρειες που αφορούν την εγκατάσταση των παρεχόμενων πόρων και της δημιουργίας των μεταδεδομένων θα αποκρύπτονται από τους χρήστες μέσω των wizards.

β) Δημιουργία ενός lightweight συστήματος που θα απαιτεί ελάχιστη διαχείριση. Το σύστημα που θα εγκαθίσταται στους providers των επιστημονικών πόρων θα απαιτεί τη μικρότερη δυνατή διαχείριση. Αντιθέτως, heavyweight middleware λογισμικό θα βρίσκεται στους servers που θα διαχειρίζονται εξειδικευμένοι administrators.

γ) Υλοποίηση μεθόδων για δυναμικό φιλτράρισμα δεδομένων (dynamic data filtering). Τα δεδομένα συχνά χρειάζεται να αλλάζουν μορφή έτσι ώστε να μπορούν να αποτελέσουν είσοδο σε κάποιο πρόγραμμα ή visualization tool. Το σύστημα θα πρέπει να επιτρέπει στους χρήστες να δημιουργούν φίλτρα τα οποία θα μπορούν στη συνέχεια να ενσωματώνουν στο σύστημα.

3. **Διασφάλιση της υιοθέτησης και μελλοντικής εκμετάλλευσης των ερευνητικών αποτελεσμάτων του ARION.**

α ) Θα γίνει έλεγχος κι επιβεβαίωση της σωστής λειτουργίας της αρχιτεκτονικής του ARION με τη χρήση πολλών αντιπροσωπευτικών επιστημονικών συλλογών από τον τομέα του coastal zone management (CZM).

β) Ανάπτυξη στρατηγικής για την εκμετάλλευση του ARION μετά την ολοκλήρωση του έργου.

## 3.3 Λίστα συμμετεχόντων

Στο έργο ARION συμμετέχουν οι εξής:

- Ίδρυμα Τεχνολογίας και Έρευνας (FORTH - συντονιστής)

- Πανεπιστήμιο Κρήτης (UoC)
- Consiglio Nazionale delle Ricerche (CNR - IMA)
- Εθνικό Μετσόβειο Πολυτεχνείο (NTUA)
- HR Wallingford Ltd (HR Wallingford)
- Enterprise LSE Limited (ELSE)
- Oceanographic Company of Norway ASA (OCEANOR)
- Commission of the European Communities, Joint Research Centre (CEC-JRC-SAI)

## 3.4 Καινοτομία

### 3.4.1 Ψηφιακές Βιβλιοθήκες: State of the art

Μια ψηφιακή βιβλιοθήκη επιστημονικών συλλογών βασίζεται στη σημαντική πρόοδο που έχει συντελεστεί σε πολλούς τεχνολογικούς τομείς που εμφανίζονται σε ένα τέτοιο έργο. Θα δούμε σύντομα τη σχετική δουλειά που έχει γίνει στο σχετικό χώρο και θα επικεντρωθούμε κυρίως στα συγκεκριμένα προβλήματα που εμφανίζονται σε σχέση με τις επιστημονικές συλλογές. Δεν υπάρχει προηγούμενη εργασία για μία τέτοια ψηφιακή βιβλιοθήκη, όμως πολλά πρόσφατα έργα που αποσκοπούν στη διαχείριση συλλογών repositories που είναι διάσπαρτα στο δίκτυο και τα οποία περιέχουν δεδομένα, προγράμματα και εργαλεία για συγκεκριμένες επιστημονικές περιοχές αποτελούν βήματα προς αυτήν την κατεύθυνση [41], [42], [43], [44], [45], [46], [47], [48]. Ο ARION αποτελεί ένα ανοικτό σύστημα πολλαπλών επιστημονικών συλλογών που γενικεύει τις ιδέες και τις λύσεις που έχουν χρησιμοποιηθεί ως σήμερα στο χώρο της διαχείρισης επιστημονικών αντικειμένων. Επιπλέον, εισάγει νέες έννοιες κι εργαλεία που απαιτούνται για τις ανάγκες ενός τέτοιου συστήματος.

Η ταχεία ανάπτυξη των κατανεμημένων συστημάτων σε συνδυασμό με την ανάπτυξη του Internet και του WWW έχουν επιφέρει μεγάλες αλλαγές στη διαχείριση, επεξεργασία και προώθηση της επιστημονικής πληροφορίας. Αποθήκες δεδομένων που ως τώρα είχαν αναπτυχθεί ξεχωριστά αρχίζουν να συνδέονται σε παγκόσμια δίκτυα. Επιπλέον, με την υιοθέτηση κοινών formats ανταλλαγής δεδομένων [49], [50], [51], standard interfaces για

πρόσβαση σε βάσεις δεδομένων [52], [53] και τεχνολογίες brokering [54] στο πλαίσιο των ψηφιακών βιβλιοθηκών, οι αναπτυσσόμενες αποθήκες δεδομένων μπορούν να προσπελαστούν χωρίς τη γνώση της εσωτερικής μορφής της σύνταξης και του τρόπου αποθήκευσης των δεδομένων. Επιπροσθέτως, οι μηχανές αναζήτησης [55], [56], [57] επιτρέπουν τον εντοπισμό κατανεμημένων πόρων με τη χρήση περιγραφών μεταδεδομένων [58], [59], [60], [61]. Οι ανοικτές αρχιτεκτονικές [62], [63], [64] παρέχουν υποστήριξη της απομακρυσμένης κλήσης legacy κώδικα ανοίγοντας το δρόμο προς μία παγκόσμια κατανεμημένη βιβλιοθήκη επιστημονικών προγραμμάτων. Τέλος, συστήματα διαχείρισης ροών εργασίας (workflow management systems) υπάρχουν για τη δρομολόγηση και την παρακολούθηση της εκτέλεσης επιστημονικών υπολογισμών. Οι δημιουργία standards και η διαλειτουργικότητα είναι οι στόχοι που έχει θέσει το WMC [1].

Αυτή η τεχνολογία έχει επιτυχώς χρησιμοποιηθεί για τις ανάγκες της διαλειτουργικότητας των συστημάτων στο επίπεδο της σύνταξης και της δομής κατανεμημένων και ετερογενών επιστημονικών αποθηκών. Όμως, η διαλειτουργικότητα σε επίπεδο σημασιολογίας απαιτείται για την υπερκέραση του προβλήματος του εντοπισμού των επιστημονικών πόρων που μπορούν κι έχει νόημα να συνδυαστούν για την παραγωγή νέων δεδομένων. Αυτό το στοιχείο είναι βασικό για την παροχή σε μεγάλο εύρος του πληθυσμού εξελιγμένων, value-added υπηρεσιών. Το πρόβλημα της δημιουργίας λογικών συνδυασμών από τους διαθέσιμους πόρους (data sets και προγράμματα) για τη δημιουργία δεδομένων που απαιτούνται για μία συγκεκριμένη εργασία (π.χ. περιβαλλοντικός σχεδιασμός, πρόβλεψη) όταν οι πόροι είναι κατανεμημένοι και σε ετερογενείς αποθήκες έχει αντιμετωπιστεί στο [43].

Ένα άλλο σώμα που έχει ασχοληθεί με τη συνένωση ετερογενούς πληροφορίας που βρίσκεται σε κατανεμημένες πηγές παρουσιάζεται στα [65], [66] και [67]. Σε αυτό το πλαίσιο ο στόχος είναι το χτίσιμο παγκόσμιων περιβαλλοντικών συστημάτων [68]. Η επιδιωκούμενη συνένωση έχει βοηθηθεί από την τεχνολογία ροών εργασίας η οποία είχε χρησιμοποιηθεί αρχικά σε επιχειρηματικές διαδικασίες. Η διαχείριση ροών εργασίας και geo-spatial διαδικασιών ερευνάται στο [69] και [70]. Στο [71] το περιγραφόμενο σύστημα ασχολείται με μία ροή εργασίας που χρησιμοποιείται για τη διαχείριση

επιστημονικών πειραμάτων [72] με τη χρήση ενός αντικεμενοστραφούς μοντέλου διαχείρισης δεδομένων. Στο [73] παρουσιάζεται ένα περιβάλλον διαχείρισης επιστημονικών εφαρμογών το οποίο χρησιμοποιεί ένα εργαλείο διαχείρισης ροών εργασίας [74].

### 3.4.2 Λύσεις για Επιστημονικές Συλλογές

#### 3.4.2.1 Μια ψηφιακή βιβλιοθήκη επιστημονικών συλλογών

Μία ψηφιακή βιβλιοθήκη επιστημονικών συλλογών είναι μία ιδέα που παρουσιάζεται για πρώτη φορά. Περιλαμβάνει τα χαρακτηριστικά μίας παραδοσιακής βιβλιοθήκης και επιπροσθέτως δημιουργεί νέα γνώση. Στις παραδοσιακές βιβλιοθήκες οι άνθρωποι παράγουν νέες γνώσεις αφού έχουν χρησιμοποιήσει το περιεχόμενο της βιβλιοθήκης. Στην περίπτωση των επιστημονικών δεδομένων (και στην ψηφιακή βιβλιοθήκη του ARION) νέο περιεχόμενο δημιουργείται συνεχώς έπειτα από αίτηση του χρήστη. Οποιοσδήποτε επιστημονικός τομέας μπορεί να παρασταθεί όχι μόνο με πολυμεσική πληροφορία, αλλά επίσης και μέσω data sets, προγραμμάτων και εργαλείων που μπορούν να παράγουν νέα πληροφορία είτε αναλύοντας δεδομένα είτε προβλέποντας φυσικά φαινόμενα μέσω προσομοιώσεων των φυσικών φαινομένων. Η ανάλυση δεδομένων μπορεί να αφορά κάποια στατιστική ανάλυση ή την εξαγωγή πληροφορίας π.χ. από δορυφορικές εικόνες ή την ανάκτηση στοιχείων από βάσεις δεδομένων που ανήκουν στη βιβλιοθήκη με τη χρήση data mining εργαλείων κλπ.

Μία άλλη διαφορά από τις παροδοσιακές βιβλιοθήκες είναι ότι το περιεχόμενο μίας τέτοιας βιβλιοθήκης δεν βρίκεται συγκεντρωμένο σε κάποιον χώρο και ούτε θα μπορούσε να είναι. Τα επιστημονικά αντικείμενα όπως τα προγράμματα δεν είναι γενικά μεταφέρσιμα (portable) κι επιπλέον μπορεί να απαιτούν ειδικό λογισμικό/υλικό για να εκτελεστούν. Στο πλαίσια του ARION αυτά μπορεί να βρίσκονται στους servers των providers και να εκτελούνται απομακρυσμένα από το σύστημα. Επομένως, τα περιεχόμενα αυτής της ψηφιακής βιβλιοθήκης είναι κατανεμημένα στους servers των providers. Στην πραγματικότητα, στην αρχιτεκτονική του συστήματος υπάρχει διαχωρισμός μεταξύ των servers του ARION και των servers των providers. Τους τελευταίους τους αποκαλούμε lightweight servers καθώς σε αυτούς δεν βρίσκεται

εγκατεστημένο λογισμικό του συστήματος πέρα από κάποιους wrappers που επιτρέπουν τη χρησιμοποίησή τους από το σύστημα του ARION. Επιπλέον, αυτή η ψηφιακή βιβλιοθήκη καταγράφει όχι μόνο τα επιστημονικά αντικείμενα (metadata), αλλά και την επιστημονική γνώση (ontologies) προκειμένου να γίνει δυνατή η χρήση της από τους χρήστες. Γραφικά εργαλεία χρησιμοποιούνται για τη μεταφορά της πληροφορίας στους χρήστες, ενώ παρέχονται επίσης στατιστικά εργαλεία κι όποιο άλλο εργαλείο χρησιμοποιούν οι επιστήμονες με τα data sets και τα προγράμματά τους.

Ο ARION έχει τη δυνατότητα να γίνει ένα διεθνές forum επιστημονικών δεδομένων και να ηγηθεί της προσπάθειας για τη δημιουργία ψηφιακών βιβλιοθηκών με επιστημονικά αντικείμενα σε παγκόσμιο επίπεδο. Στο βαθμό που είναι γνωστό η γενίκευση των ιδεών που παρουσιάζεται στον ARION δεν έχει επιχειρηθεί ποτέ πριν στον παρελθόν. Η προηγούμενη σχετική δουλειά έχει αντιμετωπίσει θέματα διαχείρισης επιστημονικής πληροφορίας για συγκεκριμένες επιστημονικές περιοχές κι έτσι η δουλειά που είχε να γίνει ήταν πολύ πιο απλή. Στην περίπτωση του ARION η κλιμακωσιμότητα, η γενικότητα των δεδομένων κι ο αυτοματοποιημένος, οπότε κι επιθυμητός, τρόπος προσθήκης νέου περιεχομένου αντιμετωπίζοται στην αρχιτεκτονική του συστήματος. Αυτή η ψηφιακή βιβλιοθήκη είναι από τη φύση της κατανεμημένη. Το WWW interface της την καθιστά προσβάσιμη από οπουδήποτε με τη χρήση ενός web browser και μίας σύνδεσης στο Internet. Επομένως, παρέχει τη δυνατότητα για ένα περιβάλλον διεθνούς συνεργασίας. Αυτό προσθέτει εξαιρετικά μεγάλη αξία σε μία παγκόσμια κοινότητα χρηστών.

#### 3.4.2.2 Τεχνική καινοτομία

Η καινοτομία έγκειται στην προχωρημένη αρχιτεκτονική που προτείνεται η οποία συνδυάζει διάφορες τεχνολογίες για την επιτυχή συνένωση επιστημονικών συλλογών στα πλαίσιο του συστήματος ψηφιακής βιβλιοθήκης ARION. Παρέχει lightweight εργαλεία για την αυτοματοποίηση της δημοσίευσης των συλλογών των providers. Παρέχει στο χρήστη ένα αυτοματοποιημένο, γρήγορο και ακριβές σύστημα για τον εντοπισμό, ανάκτηση και παρουσίαση των δεδομένων που ζητά. Στις επιστημονικές συλλογές η ύπαρξη επιστημονικών προγραμμάτων παρέχει τη δυνατότητα υπολογισμού

δεδομένων έπειτα από αίτηση του χρήστη με τη δημιουργία πολύπλοκων συνδυασμών δεδομένων και προγραμμάτων που βρίσκονται εγκατεστημένα σε ετερογενείς, αυτόνομες και γεωγραφικά κατανεμημένες συλλογές. Η υποστήριξη των προχωρημένων αυτών λειτουργιών βασίζεται στο ταίριασμα οντολογιών με μεταδεδομένα και ροές εργασίας προκειμένου να αντιμετωπιστούν οι ανάγκες υποστήριξης πολλαπλών επιστημονικών συλλογών.

Η προτεινόμενη λειτουργικότητα επιφέρει ποικίλες τεχνικές καινοτομίες που παρουσιάζονται παρακάτω:

- **Καθοδηγούμενη διαχείριση οντολογιών και μεταδεδομένων:** Ένα σύστημα διαχείρισης οντολογιών και δεδομένων θα αναπτυχθεί για on-line χρήση. Οι επιστήμονες θα είναι σε θέση να επεκτείνουν τις οντολογίες προκειμένου να καταγράψουν τους πόρους τους με τρόπο που θα επιτρέπει σε άλλους (και στο σύστημα) να τις χρησιμοποιούν. Οι σχετικές δομές μεταδεδομένων θα ανανεώνονται δυναμικά σε αντιστοιχία με τους ορισμούς των οντολογιών που δίνονται διατηρώντας τις συσχετίσεις μεταξύ των διαφόρων εννοιών.
- **Συσχέτιση δεδομένων με μεταδεδομένα:** Θα διερευνηθούν τρόποι κι όπου χρειαστεί θα αναπτυχθούν νέοι για τη διασφάλιση της συσχέτισης των μεταδεδομένων με τα δεδομένα που περιγράφουν. Σε συστήματα που ασχολούνται με ετερογενείς πόρους, π.χ. επιστημονικές συλλογές, είναι συχνό φαινόμενο οι πόροι στους οποίους καταδεικνύουν κάποια μεταδεδομένα να μην υπάρχουν ή να έχουν μετακινηθεί. Προφανώς, το στοιχείο αυτό αποτρέπει τη χρησιμοποίηση των πόρων. Έτσι, απαιτείται η ανάπτυξη τεχνικών για την αποδοτικότερη συσχέτιση δεδομένων-μεταδεδομένων.
- **Εκτέλεση ροών εργασιών:** Το σύστημα θα είναι σε θέση να εκτελεί ροές εργασιών πραγματοποιώντας τις αναγκαίες μετατροπές στα δεδομένα εκτελώντας τα σχετικά προγράμματα στο background. Πέρα από τη δυνατότητα να λαμβάνει ο χρήστης τα αποτελέσματα θα μπορεί επίσης να παρακολουθεί την κατάσταση στην οποία βρίσκεται ο υπολογισμός. Για την εκτέλεση των ροών εργασίας θα υιοθετηθεί μια λύση βασισμένη σε πράκτορες.

- **Data Export Wizards:** Θα είναι δυνατόν οι επιστήμονες να κάνουν export τα δεδομένα τους χρησιμοποιώντας έξυπνα εργαλεία (wizards) τα οποία κρύβουν τις τεχνικές λεπτομέρειες και επιτρέπουν τη διμοσίευση επιστημονικών συλλογών μέσω φιλικών διεπαφών χρήσης.
- **Βιβλιοθήκες φίλτρων δεδομένων και μετατροπείς:** Μία on-line βιβλιοθήκη για την πρόσβαση σε δεδομένα και φίλτρα μετατροπής θα υλοποιηθεί για την υποστήριξη της επαναχρησιμοποίησης δεδομένων. Η δυναμική εφαρμογή τέτοιων φίλτρων πάνω σε data sets και προγράμματα θα υποστηριχθεί χρησιμοποιώντας τεχνικές βασισμένες σε πράκτορες για την υλοποίηση κινούμενου κώδικα.

### 3.4.3 ARION: An Advanced Lightweight Architecture for a Digital Library of Scientific Collections

Έχουμε προτείνει και υλοποιούμε μία προχωρημένη αρχιτεκτονική για μία ψηφιακή βιβλιοθήκη επιστημονικών συλλογών. Αυτή προωθεί τα ανεπτυγμένα χαρακτηριστικά της τεχνολογίας ψηφιακών βιβλιοθηκών και επιπλέον προωθεί χαρακτηριστικά που λαμβάνουν υπόψη τους το περιεχόμενο και τις ιδιαιτερότητες των επιστημονικών συλλογών. Το λογισμικό του συστήματος ARION θα αποτελείται από middleware που θα διασυνδέει της ετερογενείς πηγές επιστημονικών συλλογών. Επιπλέον, μία διεπαφή χρήστη προσβάσιμη μέσω web με κάποιον web browser κι έναν υπολογιστή συνδεδεμένο στο Internet, θα παρέχει πρόσβαση στο σύστημα. Μία σειρά εργαλείων αναπτύσσεται για τους providers προκειμένου να μπορούν να δημοσιεύουν και να εγκαθιστούν επιστημονικές συλλογές σε μία επιστημονική ψηφιακή βιβλιοθήκη. Τα εργαλεία αυτά θα είναι μέρος της αρχιτεκτονικής του ARION.

Συγκεκριμένα, η αρχιτεκτονική συνιστά το middleware που θα συνδυάζει την έννοια της ψηφιακής βιβλιοθήκης, την έξυπνη αναζήτηση/συνένωση πληροφοριών και τεχνολογίες ροών εργασίας. Η αρχιτεκτονική αποτελείται από τρία κύρια υποσυστήματα:

- Metadata Search Engine
- Workflow Database System
- Workflow Runtime System

τα οποία θα συνεργάζονται για να παρέχουν στον χρήστη την επιθυμητή λειτουργικότητα. Η αρχιτεκτονική παρουσιάζεται στο Σχήμα 3.1. Ακολουθεί μία σύντομη περιγραφή των κύριων modules της αρχιτεκτονικής.

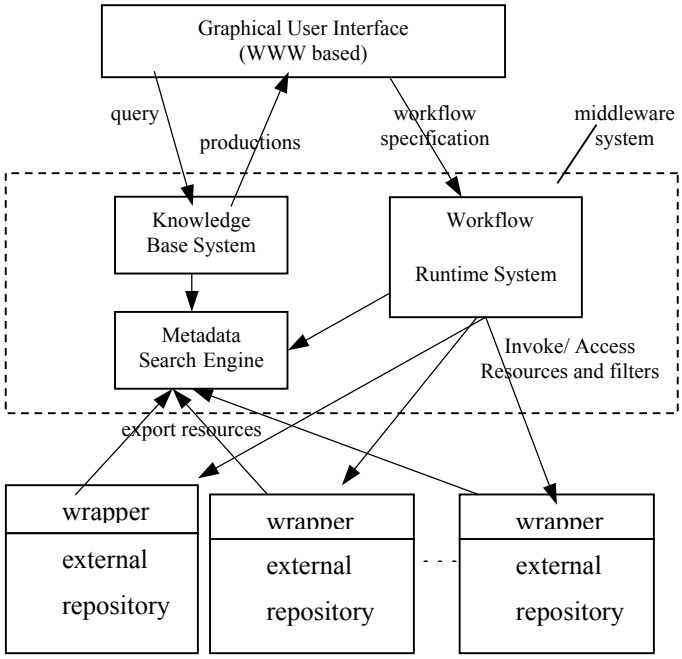
Η **Metadata Search Engine** είναι υπεύθυνη για τον εντοπισμό εξωτερικών πόρων (δεδομένων ή προγραμμάτων). Επίσης αναλαμβάνει την ανάκτηση συμπληρωματικής πληροφορίας που βρίσκεται αποθηκευμένη σε repositories. Η Search Engine δέχεται επερωτήσεις μεταδεδομένων στις ιδιότητες των πόρων κι επιστρέφει μία λίστα από περιγραφές μεταδεδομένων κι αναφορών. Οι αναφορές δείχνουν σε repository wrappers που παρέχουν πρόσβαση σε δεδομένα και δυνατότητα εκτέλεσης των προγραμμάτων στο μηχάνημα όπου είναι εγκατεστημένα.

Το **Workflow Database System** δέχεται επερωτήσεις σχετικά με τη διαθεσιμότητα οντολογιών εννοιών. Δημιουργεί κι επιστρέφει τα σχετικά δεδομένα που βασίζονται στους διαθέσιμους πόρους και τους περιορισμούς που τίθενται από τους κανόνες της οντολογίας. Σε αυτά βρίσκονται όλες οι πληροφορίες που απαιτούνται για την κατασκευή περιγραφών ροών εργασίας. Το Workflow Database System σε τακτά διαστήματα επικοινωνεί με τη Metadata Search Engine προκειμένου να ενημερώσει τη βάση δεδομένων του.

Το **Workflow Runtime System** κατευθύνει και παρακολουθεί την εκτέλεση των ροών εργασίας. Εκτελεί τα ενδιάμεσα βήματα μίας περιγραφής ροής εργασίας έχοντας πρόσβαση στα δεδομένα και εκτελώντας προγράμματα μέσω μίας υλοποίησης βασισμένης σε κινούμενους πράκτορες. Έχει ενσωματωμένες λειτουργίες checkpoint και ανάνηψης έπειτα από σφάλμα για την υποστήριξη της ανεκτικότητας του συστήματος σε σφάλματα.

Αυτή η αρχιτεκτονική διασφαλίζει την κλιμακωσιμότητα και επεκτασιμότητα που απαιτείται σε συστήματα μεγάλων επιστημονικών συλλογών. Επιτρέπει σε λειτουργικά αυτόνομους και γεωγραφικά απομακρυσμένος οργανισμούς επιλεκτικά να δημοσιεύουν και να δίνουν πρόσβαση σε πόρους τους. Η δημοσίευση/εγκατάσταση ενός νέου πόρου

στο σύστημα απαιτεί απλώς την παροχή της κατάλληλης περιγραφής μεταδεδομένων/οντολογίας και wrappers.




**Σχήμα 3.1.** Μία middleware αρχιτεκτονική για κατανεμημένες επιστημονικές πηγές.

Για την ενίσχυση της απόδοσης και της ανοχής σε λάθη η Metadata Search Engine μπορεί να κατανεμηθεί σε πολλές μηχανές.

Τα lightweight χαρακτηριστικά αυτής της αρχιτεκτονικής βασίζονται στα χαρακτηριστικά υλοποίησης των διαφόρων components του συστήματος και περιλαμβάνουν:

- Χρήση wizards για τη δημοσίευση δεδομένων και προγραμμάτων. Επιπλέον, αυτοματοποιημένη (ηλεκτρονική) δημιουργία μεταδεδομένων/οντολογίας κι αποστολή της μέσω κατάλληλων editors.
- Lightweight εγκατάσταση συστήματος στους providers με ελάχιστες απαιτήσεις διαχείρισης.
- Χρήση κινούμενου κώδικα (mobile agents)
- Υποστήριξη φιλτραρίσματος δεδομένων

# Κεφάλαιο 4ο: Τεχνολογία κινούμενων πρακτόρων

## 4.1 Εισαγωγή

Η τεχνολογία των κινούμενων πρακτόρων είναι μία σχετικά νέα περιοχή στο χώρο των κατανεμημένων εφαρμογών. Τα τελευταία χρόνια η εξάπλωση της χρήσης της γλώσσας προγραμματισμού Java [75] έχει οδηγήσει στην εξάπλωση του ενδιαφέροντος για συστήματα κινούμενων πρακτόρων. Τα προηγούμενα χρόνια συστήματα κινούμενων πρακτόρων βασίζονταν σε πειραματικές γλώσσες προγραμματισμού, όπως Tcl [76], Sheme [77], Oblique και Rosette.

Η γλώσσα προγραματισμού Java συνέβαλε αρκετά στην εμφάνιση αρκετών συστημάτων κινούμενων πρακτόρων είτε ως εμπορικά προϊόντα είτε ως ερευνητικά προγράμματα. Η ιδεατή μηχανή που χρησιμοποιεί ως περιβάλλον εκτέλεσης των εφαρμογών Java κι ο μηχανισμός φόρτωσης κλάσεων που ενσωματώνει παράλληλα με τη δυνατότητα σειριοποίησης αντικειμένων, την απομακρυσμένη επικλήση μεθόδων και τη χρήση ινών (threads) εκτέλεσης καθιστούν την υλοποίηση μίας agent πλατφόρμας μία σχετικά εύκολη εργασία.

## 4.2 Τι είναι ένας κινούμενος πράκτορας;

Ο όρος πράκτορας ορίζεται σαν ένα πρόγραμμα υπολογιστή που δρα αυτόνομα εκ μέρους κάποιου προσώπου ή οργανισμού. Επιπλέον, οι ιδιότητες αυτόνομος, έξυπνος και κινούμενος χρησιμοποιούνται συχνά προκειμένου να χαρακτηρίσουν πράκτορες. Σήμερα ο όρος χρησιμοποιείται ευρέως για αναφορές από συστήματα ρομποτικής έως και φίλτρα ηλεκτρονικού ταχυδρομείου. Ακόμα χρησιμοποιείται για περιγραφή εφαρμογών που βοηθούν στη χρήση παραθυρικών περιβαλλόντων επικοινωνίας με τα χρήστη καθώς και για την περιγραφή κινούμενου κώδικα. Εμείς θα σταθούμε στην τελευταία περιγραφή, δηλαδή αυτήν του κινούμενου κώδικα.

Ένας κινούμενος πράκτορας (mobile agent) είναι ένα αντικείμενο μίας agent πλατφόρμας που έχει τη δυνατότητα να μεταναστεύει σε απομακρυσμένες μηχανές και να εκτελεί κώδικα τοπικά σε αυτές. Μετακινούμενοι σε περιοχές όπου βρίσκονται τα απαιτούμενα

δεδομένα ή προγράμματα είναι σε θέση να εκμεταλλευτούν τα πλεονεκτήματα της τοπικής επικοινωνίας σε σχέση με την εξ'αποστάσεως αλληλεπίδραση μέσω του δικτύου.

Οι κινούμενοι πράκτορες που θα εξετάσουμε είναι αντικείμενα (instances) κλάσεων που έχουν γραφτεί στη γλώσσα προγραμματισμού Java. Ο κώδικας που είναι γραμμένος στο σώμα μιας κλάσης είναι ο προς εκτέλεση κώδικας του πράκτορα. Μέσα στον κώδικα του πράκτορα μπορούν να εμφανίζονται εντολές που αναγκάζουν τον πράκτορα να μετακινηθεί σε κάποιο απομακρυσμένο μηχάνημα, π.χ. *move(dest)*. Οι περιορισμοί τους οποίους θέτει η γλώσσα Java δεν επιτρέπουν την συνέχιση της εκτέλεσης του κώδικα ενός κινούμενου πράκτορα από το σημείο αμέσως μετά από την εντολή *move()*. Έτσι, προκειμένου να προσωμοιωθεί η συνεχιζόμενη εκτέλεση του κώδικα από τους πράκτορες κατά τη μετάβασή τους από μηχανή σε μηχανή συνιστάται ο κώδικας των πρακτόρων να γράφεται με την εξής μορφή:

```
live() {
        switch (state) {
                case 1 :
                        //code to be executed at first site
                         ...
                         ...
                        state = 2;
                        move(destination1);
                        break;

                case 2 :
                        //code to de executed at second site
                        ...
                        ...
                        state = 3;
                        move(destination2);
                        break;
```

```
                case 3:
                .
                .
                .
        }
}
```

## 4.3 Γιατί κινούμενοι πράκτορες;

Το runtime σύστημα του ARION εκμεταλλεύεται την τεχνολογία των κινούμενων πρακτόρων προκειμένου να εξυπηρετήσει την ανάγκη εκτέλεσης προγραμμάτων που βρίσκονται σε απομακρυσμένες μηχανές.

Οι κινούμενοι πράκτορες γενικά χρησιμοποιούνται προκειμένου να μειωθεί ο φόρτος του δικτύου, για να ξεπεραστεί η καθυστέρηση του δικτύου (network latency) και για να ενσωματώσουν πρωτόκολλα. Επιπλέον, η χρήση τους έχει υιοθετηθεί ευρέως για τους κάτωθι λόγους:

- Η τεχνολογία των κινούμενων πρακτόρων είναι κατάλληλη για την εκτέλεση ροών εργασίας. Η έννοια της εκτέλεσης προγραμμάτων που είναι εγκατεστημένα σε διαφορετικές μηχανές έρχεται σε απόλυτη αντιστοιχία με το μοντέλο εκτέλεσης των κινούμενων πρακτόρων. Επίσης, παρέχει δυνατότητες για την υποστήριξη περίπλοκων υπηρεσιών ανάκτησης πληροφοριών.

- Επιτρέπει μεγάλη ευελιξία στο runtime σύστημα σε σχέση με τη δρομολόγηση των εργασιών προς εκτέλεση. Ένας πράκτορας μπορεί να δημιουργηθεί με μεταβλητό επίπεδο ευφυίας και δικαιοδοσίας να αποφασίζει ή/και να συνεργάζεται με άλλους πράκτορες κατά τη διάρκεια της εκτέλεσης μίας ροής εργασίας. Έτσι, οι πράκτορες είναι σε θέση να αναλαμβάνουν μέρος της διαδικασίας της δρομολόγησης των εργασιών δημιουργώντας ένα αποκεντρωμένο σύστημα δρομολόγησης.

- Οι πράκτορες μπορούν να εκτελέσουν πολύπλοκες εργασίες καθώς και να επικοινωνήσουν/συνεργαστούν με άλλους πράκτορες εκ μέρους του χρήστη. Επίσης, λόγω της ιδιότητας της αυτονομίας τους είναι σε θέση να λειτουργούν ανεξάρτητα, ακόμα κι αν ο χρήστης είναι αποσυνδεμένος από το κεντρικό σύστημα, στοιχείο που τους καθιστά ιδανικούς για την εκτέλεση αυτοματοποιημένων εργασιών.

- Ο χρήστης είναι απαλλαγμένος από την υποχρέωση να είναι μονίμως σενδεδεμένος στο σύστημα. Οι πράκτορες λειτουργούν εκ μέρους των χρηστών χωρίς να απαιτούν από τους χρήστες να είναι συνδεδεμένοι καθόλη τη διάρκεια της εκτέλεσης μίας ροής εργασίας. Οι χρήστες μπορούν όποτε το επιθυμούν να συνδέονται με το σύστημα και να ελέγχουν την πρόοδο της εκτέλεσης μίας ροής εργασίας.

- Αντιμετωπίζει την περιορισμένη κλιμακωσιμότητα που εμφανίζεται στο RPC-based μοντέλο. Λόγω των ενσωματωμένων χαρακτηριστικών του RPC μοντέλου απαιτούνται δύο βήματα στην επικοινωνία για την ανάθεση εργασίας και ανάκτησης των αποτελεσμάτων ενώ η μηχανή εκτέλεσης αναλαμβάνει πλήρως το βάρος της δρομολόγησης και ανάθεσης των εργασιών. Επιπλέον, σε αυτήν την περίπτωση όλος ο φόρτος της επικοινωνίας συγκεντρώνεται στη μηχανή εκτέλεσης. Είναι αρκετά συνηθισμένη σήμερα για πολλούς οργανισμούς η ανάγκη να χειριστούν την ταυτόχρονη εκτέλεση πολλών ροών εργασίας με αποτέλεσμα τη συνεχώς αυξανόμενη ανάγκη για καλύτερες επιδόσεις και κλιμακωσιμότητα. Η τεχνολογία των κινούμενων πρακτόρων μπορεί να χρησιμοποιηθεί για να ξεπεραστούν αυτοί οι περιορισμοί. Η βασική διαφορά στο μοντέλο των πρακτόρων σε σχέση με το RPC μοντέλο έγκειται στο γεγονός ότι στην πρώτη περίπτωση η δρομολόγηση και η ανάθεση των εργασιών δεν είναι αποκλειστική ευθύνη των μηχανών εκτέλεσης. Οι κινούμενοι πράκτορες μπορούν να μεταφέρουν μαζί τους μέρος ή και όλη την περιγραφή μιας ροής εργασίας και να αναλαμβάνουν αυτοί τη δρομολόγηση της εκτέλεσης κάποιων εργασιών απελευθερώνοντας την κεντρική μηχανή από το βάρος της αποκλειστικής διαχείρισης της εκτέλεσης της ροής εργασίας. Αυτό συνεπάγεται ότι ο υπολογιστικός φόρτος καθώς κι αυτός της επικοινωνίας κατανέμεται ανάμεσα στις μηχανές εκτέλεσης και τους εκτελεστές εργασιών (πράκτορες).

Η λειτουργία ενός κινούμενου πράκτορα υποστηρίζεται από την ύπαρξη μίας πλατφόρμας πρακτόρων η οποία και πρέπει να είναι ενεργή σε όλες τις μηχανές που συμμετέχουν στην εκτέλεση μίας ροής εργασίας προκειμένου να εξυπηρετήσουν την ανάγκη μετανάστευσης των πρακτόρων μεταξύ διαφορετικών μηχανών.

## 4.4 Πλατφόρμες κινούμενων πρακτόρων

Όπως αναφέραμε σήμερα υπάρχει ένας σημαντικός αριθμός από πλατφόρμες κινούμενων πρακτόρων (mobile agent platforms) που είναι διαθέσιμες. Σκόπιμο είναι λοιπόν να παρουσιάσουμε τις κυριότερες από αυτές στεκόμενοι κυρίως στην πλατφόρμα που επιλέχθηκε για τους σκοπούς της εργασίας η οποία κι εξετάζεται στην παράγραφο που ακολουθεί.

### 4.4.1 Grasshopper

#### 4.4.1.1 Εισαγωγή

Το Grasshopper [78] είναι μια ανοικτή agent πλατφόρμα 100% υλοποιημένη στη γλώσσα προγραμματισμού Java η οποία είναι συμβατή και με τα δύο παγκόσμια standards πρακτόρων: το OMG MASIF [79] και το FIPA [80] τα οποία υποστηρίζουν τη διαλειτουργικότητα μεταξύ πλατφορμών πρακτόρων διαφορετικών κατασκευαστών καθώς και των πρακτόρων μεταξύ τους. Το Grasshopper περιλαμβάνει δύο open source πακέτα που υλοποιούν τα interfaces των δύο standards το οποία και μπορούν προαιρετικά να εγκατασταθούν.

Το Grasshopper είναι χτισμένο για λειτουργία στα πλαίσια ενός κατανεμημένου συστήματος. Έτσι, επιτυγχάνει τη συνένωση του παραδοσιακού client/server μοντέλου με το μοντέλο των κινούμενων πρακτόρων.

#### 4.4.1.2 Λειτουργικότητα

Το Grasshopper επιτρέπει την κατασκευή εφαρμογών Java που βασίζονται στην τεχνολογία των κινούμενων πρακτόρων. Είναι κατάλληλο για την ανάπτυξη εφαρμογών ηλεκτρονικού εμπορίου, δυναμικής ανάκτησης πληροφοριών, τηλεπικοινωνιακών

υπηρεσιών και mobile computing. Η χρήση πρακτόρων που μετακινούνται μεταξύ διαφορετικών συστημάτων επιτρέπει την εκτέλεση υπολογισμών σε απομακρυσμένα μηχανήματα επιτρέποντας στην εφαρμογή να εκμεταλλευτεί τα πλεονεκτήματα της γρήγορης επικοινωνίας και πρόσβασης σε απομακρυσμένους πόρους καθώς πλέον η εκτέλεση γίνεται τοπικά από έναν πράκτορα.

Συνοπτικά, το Grasshopper επιτρέπει στο χρήστη:

- τη δημιουργία πρακτόρων οι οποίοι μπορούν να μεταναστεύσουν σε απομακρυσμένα μηχανήματα

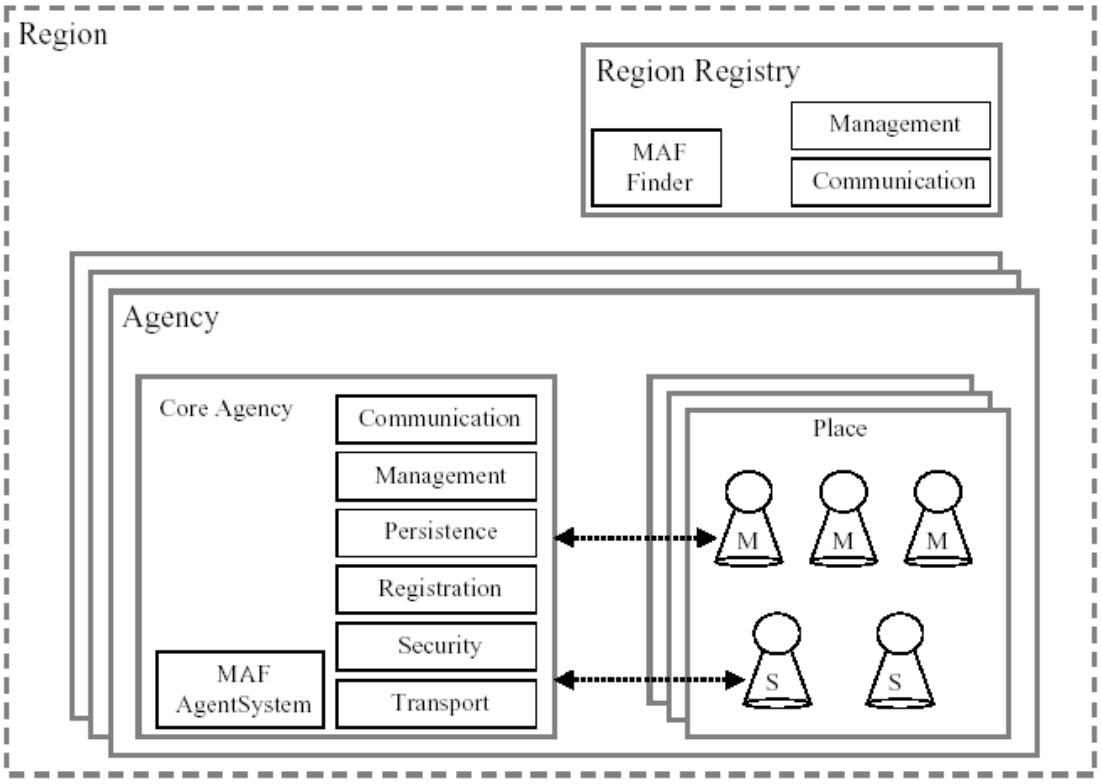
- τον αδιαφανή εντοπισμό πρακτόρων και την αποστολή μηνυμάτων προς αυτούς.

- τη διαχείριση και τον έλεγχο των πρακτόρων κατά τη διάρκεια της ζωής τους μέσω κατάλληλων εργαλείων

- τη γρήγορη ανάπτυξη εφαρμογών με την παροχή υλοποιημένων παραδειγμάτων κι εκτενούς περιγραφής του παρεχόμενου Application Programming Interface (API).

#### 4.4.1.3 Έννοιες

Η παράγραφος αυτή περιγράφει τη δομή του περιβάλλοντος του Grasshopper κι αναλύει τις έννοιες που χρησιμοποιούνται για τον προσδιορισμό των επιμέρους στοιχείων της πλατφόρμας. Το περιβάλλον του Grasshopper αποτελείται από regions, places, agencies και διάφορους τύπους πρακτόρων. Στο παρακάτω σχήμα παρουσιάζονται τα στοιχεία αυτά.




**Εικόνα 4.1.** Το περιβάλλον του Grasshopper

#### 4.4.1.3.1 Πράκτορες

##### 4.4.1.3.1.1 Κινούμενοι πράκτορες

Οι κινούμενοι πράκτορες είναι σε θέση να μετακινούνται μέσω του δικτύου μεταξύ διαφορετικών τοποθεσιών. Το στοιχείο αυτό μπορεί να θεωρηθεί ως επέκταση του μοντέλου client/server. Ενώ η client/server τεχνολογία βασίζεται στα remote procedure calls (RPCs) μέσω του δικτύου, οι κινούμενοι πράκτορες μετακινούνται αρχικά στο απομακρυσμένο μηχάνημα κι εκμεταλλεύονται την τοπικότητα στην αλληλεπίδρασή τους με αυτό. Με αυτόν τον τρόπο επιτυγχάνεται η μείωση της κίνησης στο δίκτυο και μείωση της εξάρτησης από τη διαθεσιμότητα του δικτύου.

##### 4.4.1.3.1.2 Στατικοί πράκτορες

Σε αντίθεση με τους κινούμενους πράκτορες οι στατικοί πράκτορες δεν έχουν τη δυνατότητα να μεταναστεύσουν σε απομακρυσμένα μηχανήματα. Αντιθέτως, βρίσκονται καθόλη τη διάρκεια της ζωής τους σε ένα συγκεκριμένο μηχάνημα.

#### 4.4.1.3.2 Agencies

Ένα agency αποτελεί το περιβάλλον μέσα στο οποίο βρίσκονται οι πράκτορες (τόσο οι κινούμενοι όσο κι οι στατικοί). Τουλάχιστον ένα agency πρέπει να είναι ενεργό σε κάθε μηχάνημα προκειμένου να υποστηριχτεί η λειτουργία των πρακτόρων. Ένα agency αποτελείται από δύο μέρη: το core agency κι από ένα η περισσότερα places.

##### 4.4.1.3.2.1 Core Agency

Τα core agencies παρέχουν την ελάχιστη δυνατή λειτουργικότητα που απαιτείται από ένα agency προκειμένου να υποστηρίξουν τη λειτουργία των πρακτόρων. Ένα core agency παρέχει τις παρακάτω υπηρεσίες:

**- Υπηρεσία επικοινωνίας (Communication Service)**

Αυτή η υπηρεσία είναι υπεύθυνη για την απομακρυσμένη αλληλεπίδραση μεταξύ των κατανεμημένων μερών του Grasshopper, π.χ. η επικοινωνία των πρακτόρων, η μετανάστευση των πρακτόρων, ο εντοπισμός των πρακτόρων μέσω του region registry κ.α. Όλες οι αλληλεπιδράσεις μπορούν να εκτελεστούν μέσω του CORBA IIOP [81], Java RMI [82], ή απλών συνδέσεων μέσω socket. Προαιρετικά η επικοινωνία μέσω RMI και απλών sockets μπορεί να προστατευθεί με τη χρήση του Secure Socket Layer (SSL) [83] που αποτελεί το de-facto standard πρωτόκολλο ασφαλείας του Internet. Η υπηρεσία επικοινωνίας υποστηρίζει τόσο τη σύγχρονη όσο και την ασύγχρονη επικοινωνία καθώς και τη δυνατότητα multicast.

**- Υπηρεσία καταχώρησης (Registration Service)**

Κάθε agency πρέπει να είναι σε θέση να γνωρίζει ποιοι είναι οι φιλοξενούμενοι πράκτορες και places για λόγους εξωτερικής διαχείρισης, αλλά και προκειμένου να μπορεί να παρέχει πληροφορίες στους πράκτορες σχετικά με της οντότητες που βρίσκονται εντός του agency. Η υπηρεσία καταχώρησης φροντίζει για την καταγραφή αυτού του είδους της πληροφορίας.

**- Υπηρεσία διαχείρισης (Management Service)**

Η υπηρεσία διαχείρισης επιτρέπει την παρακολούθηση και τον έλεγχο των πρακτόρων και των places ενός agency από τους χρήστες. Μεταξύ άλλων υποστηρίζονται τα παρακάτω:

- Δημιουργία, διαγραφή, πάγωμα και επαναφορά πρακτόρων και places
- Ανάκτηση πληροφορίας σχετικά με συγκεκριμένους πράκτορες και places
- Απαρίθμηση όλων των πρακτόρων ενός συγκεκριμένου place
- Απαρίθμηση όλων των πρακτόρων ενός agency

Επιπλέον, οι χρήστες μπορούν να ορίσουν ιδιότητες σχετικές με το σύστημα, την καταγραφή των ενεργειών, της ασφάλειας και της επικοινωνίας.

**- Υπηρεσία μεταφοράς (Transport Service)**

Αυτή η υπηρεσία επιτρέπει τη μετανάστευση των πρακτόρων από ένα agency προς κάποιο άλλο. Στο agency προορισμού ο πράκτορας θα είναι σε θέση να συνεχίσει την εκτέλεσή του διατηρώντας την κατάσταση των δεδομένων του (data state). Η υπηρεσία μεταφοράς χειρίζεται το externalisation και internalisation των πρακτόρων και κατευθύνει τη μεταφορά που πραγματοποιείται από την υπηρεσία επικοινωνίας (communication service).

**- Υπηρεσία ασφάλειας (Security Service)**

Το Grasshopper υποστηρίζει δύο είδη μηχανισμών ασφαλείας: την εξωτερική και την εσωτερική ασφάλεια.

> **- Εξωτερική ασφάλεια:** Προστατεύει τις απομακρυσμένες αλληλεπιδράσεις μεταξύ των κατανεμημένων κομματιών του Grasshopper, π.χ. agencies και region registries. Για το σκοπό αυτό χρησιμοποιούνται X.509 certificates [84] και το Secure Socket Layer (SSL).
>
> **- Εσωτερική ασφάλεια:** Προστατεύει τους πόρους του agency από unauthorised πρόσβαση από τους πράκτορες. Επιπλέον, προστατεύει του πράκτορες από άλλους πράκτορες. Αυτό επιτυγχάνεται με το authentication και το authorisation του χρήστη για λογαριασμό του οποίου εκτελείται ο πράκτορας.

Λόγω της ανάγκης για authentication/authorisation χρησιμοποιούνται πολιτικές ελέγχου πρόσβασης (access control policies). Η εσωτερική ασφάλεια εντός του Grasshopper βασίζεται κυρίως στο μηχανισμό ασφάλειας που παρέχεται από το JDK.

#### 4.4.1.3.2.2 Places

Ένα place παρέχει την ομαδοποίηση της λειτουργικότητας εντός ενός agency. Π.χ. μπορεί να υπάρχει ένα place που παρέχει εξελιγμένες δυνατότητες επικοινωνίας ή ένα place όπου οι πράκτορες αγοράζουν και πουλάνε πληροφορίες ή πρόσβαση σε υπηρεσίες. Το όνομα ενός place θα πρέπει να αντικατροπτρίζει το σκοπό του. Π.χ.σε κάθε agency υπάρχει by default ένα place με όνομα "InformationDesk". Ένας πράκτορας που δε ζητά να μεταφερθεί σε κάποιο συγκεκριμένο place μεταφέρεται αυτόματα στο "InformationDesk" όπου μπορεί να αναζητήσει περαιτέρω πληροφορίες.

#### 4.4.1.3.3 Regions

Το region εξυπηρετεί τη διαχείριση των κατανεμημένων κομματιών στο περιβάλλον του Grasshopper, π.χ. agencies, places και πράκτορες. Τα agencies καθώς και τα σχετιζόμενα με αυτά places μπορούν να συνδεθούν με κάποιο συγκεκριμένο region. Οι οντότητες καταγράφονται στο region registry. Κάθε registry αυτόματα καταχωρεί κάθε πράκτορα που φιλοξενείται σε κάποιο agency που έχει συνδεθεί με το συγκεριμένο region. Αν ένας πράκτορας μετακινηθεί σε κάποια άλλη τοποθεσία η σχετική πληροφορία στο registry ανανεώνεται αυτόματα. Ένα region θα μπορούσε να ενσωματώσει όλα τα agencies που ανήκουν σε μία εταιρία ή έναν οργανισμό διευκολύνοντας με αυτόν τον τρόπο τη διαχείρισή τους.

#### 4.4.1.3.3.1 Region Registries

Ένα region registry διατηρεί πληροφορίες σχετικές με όλα τα components που έχουν συνδεθεί με ένα συγκεκριμένο region. Όταν ένα νέο component, π.χ. ένα agency, place ή ένας πράκτορας δημιουργηθεί αυτόματα καταχωρείται στο σχετικό region registry. Η τρέχουσα τοποθεσία των κινούμενων πρακτόρων, δηλ. το agency και το place στο οποίο βρίσκονται ανανεώνεται έπειτα από κάθε μετανάστευση του πράκτορα. Το region registry μπορεί ανά πάσα στιγμή να πληροφορήσει τους χρήστες ή τους πράκτορες για τη

θέση των πρακτόρων καθώς και να δώσει πληροφορίες σχετικές με τα places και τα agencies εντός του region. Επίσης, διευκολύνει την επικοινωνία μεταξύ των agencies και των πρακτόρων οι οποίοι αρκεί να γνωρίζουν κάποιο αναγνωριστικό για τον πράκτορα με τον οποίο θέλουν να επικοινωνήσουν κι όχι απαραίτητα την τρέχουσα θέση του.

#### 4.4.1.4 Υποστηριζόμενα standards

Οι κινούμενοι πράκτορες αποτελούν μία σχετικά νέα τεχνολογία. Σήμερα, υπάρχει ένας σημαντικός αριθμός από πλατφόρμες κινούμενων πρακτόρων οι οποίες διαφέρουν σε μεγάλο βαθμό στην αρχιτεκτονική τους. Οι διαφορές αυτές αποτελούν εμπόδιο στην προσπάθεια για την επίτευξη διαλειτουργικότητας μεταξύ των διάφορων πλατφορμών. Όμως, η επίτευξη της διαλειτουργικότητας μεταξύ συστημάτων διαφορετικών κατασκευαστών είναι μία σημαντική απαίτηση προκειμένου να εξυπηρετηθούν οι ανάγκες μίας ανοικτής αγοράς υπηρεσιών βασισμένη σε πράκτορες. Καθώς τη βάση της διαλειτουργικότητας αποτελεί το standardisation της χρησιμοποιούμενης τεχνολογίας μία απαίτηση που αφορά τις πλατφόρμες κινούμενων πρακτόρων αποτελεί η συμμόρφωση με τα διαθέσιμα standards.

##### 4.4.1.4.1 MASIF

Ως σήμερα, η σημαντικότερη προσπάθεια κάποιου οργανισμού δημιουργίας standards στην περιοχή των κινούμενων πρακτόρων έχει γίνει από το Object Management Group (OMG) το οποίο έχει εκδώσει το πρότυπο *Mobile Agent System Interoperability Facility (MASIF)*. Το MASIF standard έχει υιοθετηθεί από το Φεβρουάριο του 1998. Περιλαμβάνει πολλά σημαντικά θέματα, όπως τη διαχείριση, την κινητικότητα, την ονομασία και τον εντοπισμό των πρακτόρων. Πιθανότατα περαιτέρω agent standards θα προκύψουν από το OMG τα οποία συνολικά θα αποτελέσουν ένα ευρέως αποδεκτό framework για την τεχνολογία κινούμενων πρακτόρων. Προκειμένου να αντιμετωπιστούν οι μελλοντικές απαιτήσεις και να παραμείνει ανοικτή προς πλατφόρμες άλλων κατασκευαστών η πλατφόρμα του Grasshopper είναι συμβατή με το OMG MASIF standard. Για το σκοπό αυτό έχει υλοποιηθεί το Grasshopper MASIF Addon που καθιστά την πλατφόρμα συμβατή με το πρότυπο.

#### 4.4.1.4.2 FIPA

Το FIPA είναι ένας μη κερδοσκοπικός οργανισμός που αποσκοπεί στη δημιουργία standards για τη διαλειτουργικότητα ετερογενών πρακτόρων λογισμικού. Το Foundation for Intelligent Physical Agents (FIPA) δημιουργήθηκε το 1996 με σκοπό την παραγωγή standards λογισμικού για ετερογενείς πράκτορες και συστήματα βασιζόμενα σε πράκτορες που αλληλεπιδρούν μεταξύ τους.

Ο σκοπός του FIPA εκφράζεται καθαρά στο επίσημο mission statement της:

> *The promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent agent systems in modern commercial and industrial settings.*

Η έμφαση δίνεται στην πρακτική εμπορική και βιομηχανική χρήση των συστημάτων πρακτόρων. Το FIPA επιχειρεί μέσω του συνδυασμού πράξεων επικοινωνίας και δημόσιων οντολογιών να προσφέρει standard τρόπους διερμήνευσης της επικοινωνίας μεταξύ πρακτόρων. Για την υποστήριξη των στόχων της το FIPA έχει υιοθετήσει και εργάζεται πάνω σε προδιαγραφές που καλύπτουν αρχιτεκτονικές για την υποστήριξη της επικοινωνίας μεταξύ πρακτόρων, γλώσσες επικοινωνίας και πρωτόκολλα αλληλεπίδρασης που επεκτείνουν τη δυνατότητα ανταλλαγής απλών μηνυμάτων σε εκτέλεση ολοκληρωμένων transactions. Υπάρχουν σχέδια για μελλοντική επέκταση των προδιαγραφών ώστε να υποστηρίζονται κι οι μακρού χρόνου σχέσεις μεταξύ πρακτόρων (long term relationships).

Το Grasshopper υλοποιεί και το FIPA standard για όσους χρήστες επιθυμούν να το χρησιμοποιήσουν.

### 4.4.1.5 Grasshopper URL

Γενικά, μια τοποθεσία Grasshopper ορίζεται μέσω ενός URL-like notation της μορφής:

```
<protocol>://<host>:<port>/<agency>/<place>
```

`protocol`: όνομα του πρωτοκόλλου που θα χρησιμοποιηθεί (π.χ. socket)

`host`: όνομα ή IP διεύθυνση του μηχανήματος

`port`: αριθμός port του μηχανήματος (προαιρετικό)

`agencyName`: το όνομα του agency

`placeName`: το όνομα του place (προαιρετικό)

#### 4.4.1.6 Επιλογή Grasshopper

Η πλατφόρμα κινούμενων πρακτόρων Grasshopper επιλέχτηκε να χρησιμοποιηθεί για την ανάπτυξη του σύστηματος της παρούσας εργασίας. Η επιλογή αυτή στηρίχθηκε σε μία σειρά λόγων οι κυριότεροι από τους οποίους παρατίθενται παρακάτω:

- **Ύπαρξη εκτεταμένου documentation σχετικά με την πλατφόρμα:** Είχαμε τη δυνατότητα γρήγορης εκμάθησης της λειτουργικότητας της πλατφόρμας και κατανόησης του API που παρέχει μέσω αρκετά καλών εγγράφων που διατίθενται ελεύθερα.
- **Ενεργό community χρηστών:** Η ύπαρξη ενός σημαντικού πλήθους χρηστών της πλατφόρμας αποδείχτηκε ιδιαίτερα σημαντική καθώς σε αρκετές περιπτώσεις ζητήθηκε η συνδρομή τους και προβλήματα που παρουσιάζονταν ξεπερνιούνταν αρκετά γρήγορα.
- **Ενσωματωμένη λειτουργικότητα:** Το Grasshopper έχει ενσωματωμένη λειτουργικότητα που χρησιμοποιήθηκε για την ταχύτερη ανάπτυξη της απαιτούμενης λειτουργικότητας (π.χ. μηχανισμός επικοινωνίας πρακτόρων με το runtime σύστημα). Αφού αναπτύχθηκε ο πυρήνας του συστήματός μας περάσαμε στην υλοποίηση των ενσωματωμένων μηχανισμών που παρέχει το Grasshopper (πέρα των βασικών της δημιουργίας και μεταφοράς πρακτόρων) τους οποίους και υλοποιήσαμε εσωτερικά στο σύστημά μας έτσι ώστε να μειώσουμε την εξάρτησή μας από τη συγκεκριμένη πλατφόρμα.

### 4.4.2 Άλλες agent πλατφόρμες

#### 4.4.2.1 IBM Aglets

Το Aglets Software Development Kit (ASDK) που αναπτύχθηκε από την IBM είναι ένα framework βασισμένο στη γλώσσα προγραμματισμού Java για τη δημιουργία κινούμενων πρακτόρων [85, 86, 87]. Το ASDK παρέχει έναν φορτωτή κλάσεων για πράκτορες μέσω δικτύου το οποίο καθιστά εφικτή τη μετακίνηση του κώδικα και της

κατάστασης ενός πράκτορα. Προς το παρόν τα Aglets δεν εκτελούνται στο περιβάλλον της Java 2 παρά μόνο σε παλιότερες εκδόσεις της Java. Οι πράκτορες ονομάζονται aglets από τη σύνθεση των λέξεων agents και applets.

Το runtime επίπεδο των aglets παρέχουν τη βασική λειτουργικότητα, όπως τη δημιουργία, μετακίνηση και διαγραφή των aglets. Το Application Programmer Interface (API) παρέχεται από έναν aglet server που ονομάζεται Tahiti. Το Tahiti παρέχει μία διεπαφή χρήσης για την παρακολούθηση, δημιουργία, μετακίνηση και διαγραφή των πρακτόρων καθώς και για τον ορισμό των δικαιωμάτων πρόσβασης σε έναν πράκτορα από τον server. Τα aglets χρησιμοποιούν ένα event-driven μοντέλο και περιέχουν event listeners για να μαθαίνουν για τη μετακίνηση των πρακτόρων. Όταν συμβαίνει ένα γεγονός (event) τότε μία μέθοδος του αντίστοιχου event listener καλείται για τη λειτουργία της διαχείρισης του server (π.χ. όταν ένας πράκτορας μεταναστεύει).

Η βασική λειτουργικότητα και τα runtime interfaces των aglets ορίζονται από τις Java κλάσεις: Aglet, AgletProxy και AgletContext. Κάθε aglet περιλαμβάνει κι ένα proxy που έλεγχει την επικοινωνία προς αυτό το aglet.

Η abstract κλάση Aglet ορίζει τις βασικές μεθόδους που έλεγχουν την κινητικότητα και τον κύκλο ζωής ενός aglet. Επίσης επιτρέπει την πρόσβαση σε κληροδοτημένες ιδιότητες προς αυτό το aglet (π.χ. ώρα δημιουργίας, ιδιοκτήτης, codebase και επίπεδο ασφάλειας), καθώς και σε δυναμικές ιδιότητες (π.χ. ώρα άφιξης, τρέχουσα διεύθυνση).

Το AgletContext interface παρέχει το runtime περιβάλλον εκτέλεσης στα aglets εντός του Tahiti server. Όταν ένα aglet αντικείμενο μετακινείται προς ένα απομακρυσμένο μηχάνημα αποσυνδέεται από το τρέχον AgletContext αντικείμενο, σειριοποείται σε μια ακολουθία από bytes, αποστέλλεται μέσω δικτύου και επαναδημιουργείται στο απομακρυσμένο μηχάνημα όπου συνδέεται με ένα νέο AgletContext αντικείμενο που παρέχει πρόσβαση στο νέο περιβάλλον εκτέλεσης.

Το AgletProxy interface παρέχει ένα handle που χρησιμοποιείται για τις ανάγκες πρόσβασης στο aglet. Το AgletProxy αντικείμενο παρέχει τη δυνατότητα πρόσβασης προς έναν πράκτορα χωρίς να υπάρχει η ανάγκη γνώσης εκ μέρους του υπόλοιπου συτήματος ή του χρήστη της τρέχουσας θέσης του πράκτορα. Το αντικείμενο

αναλαμβάνει να προωθήσει τα απαιτούμενα μηνύματα προς τον πράκτορα καθώς και να επιστρέψει τα αποτελέσματά του στο τοπικό μηχάνημα. Επίσης, λειτουργεί ως ασπίδα που προστατεύει τις μεθόδους του aglet αντικειμένου από την άμεση πρόσβαση άλλων αντικειμώνων.

Επιπλέον των τριών βασικών κλάσεων το aglet μοντέλο παρέχει επιπρόσθετες κλάσεις για την υποστήριξη σύγχρονης και ασύγχρονης επικοινωνίας μεταξύ των aglets.

Η ασφάλεια είναι πρωταρχικής σημασίας για την τεχνολογία κινούμενων πρακτόρων και τα aglets παρέχουν ένα επεκτάσιμο μοντέλο ασφάλειας βασιζόμενο στο μοντέλο ασφάλειας της Java [88].

Η πλατφόρμα αποτελεί ένα open source project και ο πηγαίος κώδικάς του διατίθεται μέσω του Internet.

#### 4.4.2.2 April agent platform

Η April Agent Platform [89] (AAP) δημιουργήθηκε από τους Jonathan Dale και Johnny Knottenbelt. Η April είναι μια FIPA-compliant agent πλατφόρμα που σχεδιάστηκε προκειμένου να αποτελέσει μία ισχυρή και ταυτόχρονα απλή λύση για τη δημιουργία συστημάτων βασισμένων σε πράκτορες. Η πλατφόρμα γράφτηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού April και το InterAgent Communication System (IMC) και παρέχει πολλά στοιχεία που βοηθούν στην επιτάχυνση της ανάπτυξης πρακτόρων. Η AAP απαιτεί τη γλώσσα προγραμματισμού April programming language και το ICM προκειμένου να εγκατασταθεί και τρέχει στα λειτουργικά συστήματα UNIX και Windows. Η April πλατφόρμα είναι ένα open source project (κάτω από GPL άδεια).

# Κεφάλαιο 5ο: Σχετική Εργασία

## 5.1 Service-Oriented Middleware

### 5.1.1 JINI

Το JINI [90] είναι τυπικό παράδειγμα service-oriented middleware. Μπορεί να περιγραφεί ως σαν ένα σύστημα με κατανεμημένες ομάδες πόρων. Οι πόροι μπορούν να είναι διάφορες συσκευές, λογισμικό ή και συνδυασμός των δύο παραπάνω. Για την αντιμετώπιση του δυναμικού περιβάλλοντος ενός κατανεμημένου συστήματος το JINI επιτρέπει τη δυναμική προσθήκη και διαγραφή υπηρεσιών κατά τη διάρκεια της λειτουργίας του με έναν ευέλικτο τρόπο. Παρόλα αυτά καθώς το JINI απαιτεί τουλάχιστον μια Java πλατφόρμα έτσι ώστε να είναι εφικτή η επικοινωνία μέσω RMI, το JINI μπορεί να θεωρηθεί αρκετά βαρύ.

### 5.1.2 Cooltown

Το Cooltown [91] αποτελεί μια αντίστοιχη προσέγγιση που επίσης ανήκει στην κατηγορία του service-oriented middleware. Στα πλαίσια του Cooltown κάθε φυσική οντότητα π.χ. δωμάτια, αντικείμενα ή χρήστες έχουν μια ιδεατή αναπαράσταση. Μια συσκευή π.χ. ένας εκτυπωτής, μπορεί να χρησιμοποιεί έναν embedded web server για την υποστήριξη της πρόσβασης σε αυτόν μέσω ενός web interface. Άλλοι πόροι μπορούν να αναπαρασταθούν μέσω μιας web σελίδας. Η προσέγγιση του Cooltown έχει αρκετά μειονεκτήματα. Οι συσκευές μπορούν να αναπαρασταθούν μονάχα μέσω web σελίδων οι οποίες πρέπει να έχουν κατασκευαστεί εκ των προτέρων και δεν υπάρχει υποστήριξη για ad-hoc δίκτυα.

### 5.1.3 Grid Systems

Μια άλλη προσέγγιση service-oriented αρχιτεκτονικής είναι το Open Grid Service Architecture (OGSA) [92] που έχει οριστεί από το Global Grid Forum. Η βασική ιδέα πίσω από το OGSA είναι η παροχή ενός υψηλότερου επιπέδου έννοιας υπηρεσιών για συστήματα grid computing. Τα σημαντικότερα χαρακτηριστικά είναι η χρήση Web Services standards όπως το SOAP, το WSDL κ.α., καθώς και standard interface definition μηχανισμούς, αδιαφάνεια ως προς τον την τοπική/απομακρυσμένη εκτέλεση και ομοιόμορφη σημασιολογία υπηρεσιών.

Ο όρος ‘grid’ χρησιμοποιείται συχνά για την αναφορά σε υπολογιστικά συστήματα υψηλής απόδοσης που συνδέονται μέσω δικτύου. Τα συστήματα Grids μπορούν να θεωρηθούν ως μια service-oriented αρχιτεκτονική στην οποία οντότητες παρέχουν υπηρεσίες η μία στην άλλη στη βάση διαφόρων μορφών συμβολαίων. Όλο και πιο πολύ τα τελευταία χρόνια έχει επικρατήσει η τάση να θεωρούμε τα μεγάλης κλίμακας συστήματα από την άποψη των υπηρεσιών που παρέχουν. Μια υπηρεσία μπορεί να θεωρηθεί ως ένας αφηρημένος τρόπος χαρακτηρισμού και encapsulation κάποιων περιεχομένων ή δυνατοτήτων επεξεργασίας. Η τεχνολογία των Grids φιλοδοξεί να χρησιμοποιηθεί ως ο μηχανισμός που θα συνενώσει σε ένα μεγάλης κλίμακας πλέγμα υπολογιστών μια μεγάλη γκάμα παρεχόμενων υπηρεσιών.

Οι υπηρεσίες σαφώς δεν υφίστανται από μόνες τους. Αντιθέτως υπάρχουν και λειτουργούν μέσα στο πλαίσιο ενός οργανισμού. Όλες οι υπηρεσίες έχουν έναν ιδιοκτήτη (ή μια ομάδα ιδιοκτητών). Ο ιδιοκτήτης είναι ο άνθρωπος ή ο οργανισμός που είναι υπεύθυνος για την παροχή της συγκεκριμένης λειτουργίας για χρήση από άλλους. Ο ιδιοκτήτης θέτει τους όρους και τις συνθήκες κάτω από τις οποίες η υπηρεσία θα είναι προσβάσιμη. Η σχέση μεταξύ παροχέα και χρήστη μιας υπηρεσίας μπορεί να ποικίλλει (π.χ. ελεύθερη χρήση, χρέωση). Η σχέση αυτή ορίζεται σαφώς σε ένα συμβόλαιο για τη συγκεκριμένη υπηρεσία. Συνεπώς, ένα grid σύστημα που υποστηρίζει μία service-oriented αρχιτεκτονική έχει ιδιοκτήτες υπηρεσιών που παρέχουν τις υπηρεσίες σε χρήστες κάτω από συγκεκριμένα συμβόλαια. Η αλληλεπίδραση ιδιοκτητών-χρηστών γίνεται βάσει των κανόνων που έχουν οριστεί για το συγκεκριμένο περιβάλλον εκτέλεσης.

Οι υπηρεσίες σε ένα grid μπορούν να προστίθενται και να αφαιρούνται δυναμικά. Αυτό σημαίνει ότι το σύστημα βρίσκεται συνεχώς σε μεταβατική κατάσταση και δε φτάνει ποτέ μία σταθερή κατάσταση.

Συνοπτικά, οι δυνατότητες που παρέχονται από ένα grid σύστημα αναφέρονται παρακάτω:

- Αποθήκευση και επεξεργασία μεγάλου όγκου δεδομένων σε λογικά χρονικά πλαίσια.

- Διατήρηση ιδιοκτησίας των περιεχομένων και δυνατοτήτων επεξεργασίας τους με ταυτόχρονη δυνατότητα παροχής πρόσβασης σε αυτά σε χρήστες βάσει συγκεκριμένων όρων και συνθηκών.

- Ανακάλυψη, αδιαφανής πρόσβαση και επεξεργασία περιεχομένων ανεξαρτήτως της θέσης τους πάνω στο Grid.

- Συνδυασμός περιεχομένων από πολλαπλές πηγές βάσει των αναγκών των χρηστών.

- Δυναμικός χαρακτήρας του Grid. Νέες υπηρεσίες μπορούν να εισάγονται ή να αφαιρούνται ενώ το σύστημα είναι σε λειτουργία.

## 5.2 Message Oriented Middleware

Το Java Messaging Service (JMS) [93] ανήκει στην κατηγορία των Message Oriented Middlewares. Παρέχει ένα API που επιτρέπει την ασύγχρονη αποστολή μηνυμάτων μεταξύ πελατών. Η αποστολή μηνυμάτων διαφέρει από τεχνολογίες όπως το Remote Method Invocation (RMI) που απαιτεί από μια εφαρμογή να γνωρίζει τον απομακρυσμένο πελάτη και να πραγματοποιεί μια απευθείας σύνδεση με το απομακρυσμένο endpoint.

## 5.3 Distributed Event Systems

Το Hermes [94] ανήκει στην κατηγορία των event-based middleware συστημάτων. Χρησιμοποιεί ένα type- και attribute- based μοντέλο δημοσίευσης/εγγραφής publish/subscribe για την κατασκευή κατανεμημένων συστημάτων μεγάλης κλίμακας. Αυτή η προσέγγιση αντιμετωπίζει τις αδυναμίες που παρουσιάζονται στη

λειτουργικότητα παραδοσιακών συστημάτων publish/subscribe όπως έλεγχος τύπου, αξιοπιστία, έλεγχος πρόσβασης και συναλλαγές (transactions).

## 5.4 Virtual Shared Memory Middleware

Η πλατφόρρμα EQUIP [95] που αναπτύχθηκε από το Πανεπιστήμιο του Nottingham παρέχει τη middleware υποδομή για τη διερεύνηση της σχέσης μεταξύ φυσικών και ψηφιακών αντικειμένων. Τα κύρια χαρακτηριστικά της είναι το cross language integration, το modularization, η επεκτασιμότητα, η αλληλεπιδραστική εκτέλεση και η ανομοιογένεια συσκευών και δικτύων. Ο διαμοιρασμός της κατάστασης επιτυγχάνεται μέσω μιας υπηρεσία διαμοιρασμού δεδομένων που είναι παρόμοια με την VSM αρχιτεκτονική. Το EQUIP υποστηρίζει το δυναμικό φόρτωμα κώδικα και μπορεί επίσης να χρησιμοποιηθεί ως ένα σύστημα κατανομής γεγονότων.

## 5.5 Peer-to-Peer Middleware

### 5.5.1 XMIDDLE

Το XMIDDLE peer-to-peer middleware [96] έχει ως στόχο την αγορά του mobile computing υποστηρίζοντας ad-hoc workgroups και διαμοιρασμό των πόρων ενός peer. XML έγγραφα μπορούν να αντιστοιχηθούν σημασιολογικά με δέντρα επιτρέποντας στα κλαδιά να συνδεθούν και να διαμοιραστούν μεταξύ των hosts. Το XMIDDLE βασίζεται στην οργάνωση των δεδομένων σε διαμοιραζόμενες δενδρικές δομές.

### 5.5.2 JXTA

Μια ευρέως υιοθετημένη αρχιτεκτονική για peer-to-peer δίκτυα είναι το JXTA [97]. Στόχος του JXTA είναι η ανάπτυξη μιας κοινής γλώσσας που θα επιτρέπει στους developers peer-to-peer συστημάτων να πραγματοποιούν τις θεμελειώδεις λειτουργίες του P2P networking. Το JXTA είναι ένα σύνολο ανοικτών peer-to-peer πρωτοκόλλων που θα καταστήσουν εφικτή την ανάπτυξη νέων δικτυακών εφαρμογών.

Τα πρωτόκολλα αυτά κρατιούνται απλά, στοιχείο που επιτρέπει τη δημιουργία συσκευών με δυνατότητα ενσωμάτωσής τους σε ένα P2P δίκτυο. Το JXTA χρησιμοποιεί XML για

την εκτέλεση εργασιών σχετικών με P2P λειτουργίες όπως η αποστολή μηνυμάτων. Το JXTA ορίζει συγκεκριμένους μηχανισμούς διευθυνσιοδότησης και είναι συνεπώς ανεξάρτητο από κάθε δίκτυο ή πρωτόκολλο μεταφοράς.

# Κεφάλαιο 6ο: Το runtime σύστημα του ARION

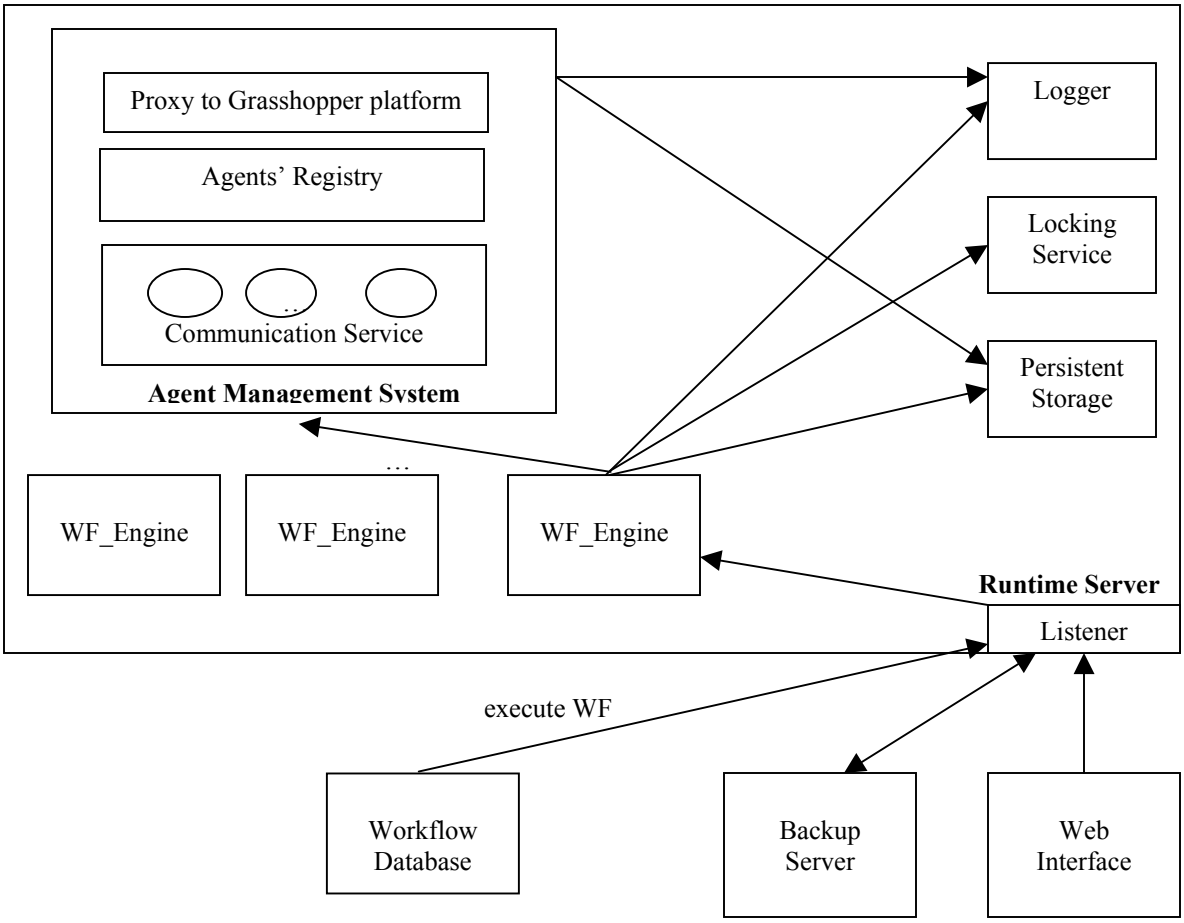
## 6.1 Εισαγωγή

Ο σκοπός του runtime συστήματος του ARION είναι η υποστήριξη της εκτέλεσης ροών εργασίας (workflows). Πιο συγκεκριμένα ο server του runtime συστήματος δέχεται αιτήσεις από τη Workflow Database (§ 4.4.3) που αφορούν την εκτέλεση μίας ή περισσοτέρων ροών εργασίας κι αναλαμβάνει την εξυπηρέτησή τους. Μία εισερχόμενη αίτηση για την εκτέλεση μίας ροής εργασίας συμπεριλαμβάνει τον ορισμό της ροής. Η εκτέλεση μίας ροής εργασίας ανατίθεται σε μία μηχανή εκτέλεσης ροής εργασίας (workflow engine). Το runtime σύστημα υποστηρίζει την παρακολούθηση (monitoring) της πορείας εκτέλεσης των ροών εργασίας, ενώ όπως θα αναλυθεί παρακάτω επιτρέπει την αλληλεπίδραση ενός χρήστη με το σύστημα και τη μηχανή εκτέλεσης έχοντας τη δυνατότητα ακόμα και να επηρρεάσει την πορεία εκτέλεσης.

Το runtime σύστημα εκμεταλλεύεται πολλές διαφορετικές τεχνολογίες τις οποίες εντάσσει σε ένα κοινό πλαίσιο χρήσης. Παραδείγματα τέτοιων τεχνολογιών είναι η τεχνολογία των κινούμενων πρακτόρων (mobile agents), η δημιουργία δυναμικών HTML σελίδων με τη χρήση Java Servlets, τα Java Applets κ.α.

## 6.2 Αρχιτεκτονική

Το runtime σύστημα σχεδιάστηκε ακολουθώντας τις βασικές αρχές σχεδίασης ενός οντοκεντρικού προγράμματος. Το σύστημα το απαρτίζουν πολλές διαφορετικές οντότητες οι οποίες αλληλεπιδρούν μέσω καθορισμένων διεπαφών (interfaces). Στο παρακάτω σχεδιάγραμμα παρουσιάζονται οι βασικές οντότητες του συστήματος όπου και γίνονται φανερές οι δυνατότητες αλληλεπίδρασης μεταξύ τους (Σχήμα 6.1).




**Σχήμα 6.1.** Η αρχιτεκτονική του runtime συστήματος του ARION

### 6.2.1 Runtime Server

#### 6.2.1.1 Εισαγωγή

Ο σκοπός του server του runtime συστήματος είναι η υποστήριξη της εκτέλεσης ροών εργασίας. Οι αιτήσεις για την εκτέλεση κάποιας περιγραφής ροής εργασίας αποστέλλονται στον server από τη Workflow Database προκειμένου να παραχθούν ένα ή περισσότερα data sets. Μία αίτηση εκτέλεσης ροής εργασίας έχει ως αποτέλεσμα τη δημιουργία μίας μηχανής εκτέλεσης. Η μηχανή εκτέλεσης αναλαμβάνει εξ'ολοκλήρου την ευθύνη της εκτέλεσης της ροής εργασίας. Το βασικότερο μέρος μιας μηχανής εκτέλεσης αποτελεί ο δρομολογητής εργασιών (task scheduler) ο οποίος αναλαμβάνει τη μετάφραση της περιγραφής της ροής εργασίας σε μία σειρά ενεργειών που πρέπει να εκτελεστούν καθώς και τη δρομολόγησή τους.

Ο δρομολογητής καθορίζει ποια πρέπει να είναι η σειρά εκτέλεσης των προγραμμάτων που απαρτίζουν την περιγραφή της ροής εργασίας στους απομακρυσμένους υπολογιστές, όπου αυτά είναι εγκατεστημένα, καθώς και των υπόλοιπων εργασιών που πρέπει να γίνουν, ενώ επίσης φροντίζει για τη σωστή διαχείριση των δεδομένων που παράγονται από τα προγράμματα ή παρέχονται από το χρήστη (π.χ. παράμετροι αρχικοποίησης προγραμμάτων). Οι συσχετίσεις μεταξύ των εργασιών, όπως αυτές ορίζονται στη ροή εργασίας, διατηρούνται με παράλληλη προσπάθεια να επιτευχθεί η μέγιστη δυνατή ταύτοχρονη εκτέλεση εργασιών.

Η εκτέλεση μίας εργασίας ανατίθεται σε έναν εκτελεστή εργασίας (task manager). Ένας εκτελεστής εργασίας αναλαμβάνει να εκτελέσει τα επιμέρους βήματα που αφορούν μία συγκεκριμένη εργασία ενημερώνοντας το δρομολογητή για την πρόοδο της εκτέλεσής της. Ένας εκτελεστής εργασίας αρχικοποιείται με την περιγραφή μιας εργασίας την οποία και μεταφράζει στις επιμέρους ενέργειες που πρέπει να εκτελεστούν τις οποίες και δρομολογεί. Παραδείγματα ενεργειών που δρομολογεί ένας εκτελεστής εργασίας είναι η μεταφορά κάποιου/ων data set(s) από ένα μηχάνημα σε κάποιο άλλο, η αποστολή ενός πράκτορα για την εκτέλεση ενός προγράμματος σε απομακρυσμένο μηχάνημα κ.α. Συνεπώς ένας εκτελεστής εργασίας πραγματοποιεί κάποιο είδος δρομολόγησης ενεργειών σε επίπεδο εργασίας (task) αντίστοιχα όπως κι ο δρομολογητής σε επίπεδο ροής εργασίας.

#### 6.2.1.2 Εκκίνηση server

Ο server εκκινείται παίρνοντας ως παράμετρο ένα αρχείο αρχικοποίησης το οποίο περιέχει τις αρχικές παραμέτρους που λαμβάνει ο server. Μέσα σε αυτό ορίζονται:

- Η IP διεύθυνση του μηχανήματος όπου θα τρέχει ο server (*serverAddress*)
- Ο αριθμός του port στο οποίο ο server ακούει για εισερχόμενες αιτήσεις (*serverPort*)
- Το όνομα του server (*serverName*)
- Πληροφορίες σχετικά με τον server (*serverInfo*)

- Το directory όπου θα κρατούνται τα logs καθώς και τα στιγμιότυπα που θα σώζουν οι διάφορες οντότητες (*serverLogDir*) (για λόγους ανάνηψης έπειτα απο σφάλμα)
- Flag που δείχνει αν ο server εκκινείται σε κατάσταση ανάνηψης (*recovering*)
- Το directory από το οποίο θα διαβαστεί η σωσμένη κατάσταση για την εκκίνηση του server σε κατάσταση ανάνηψης (*recoverDir*)
- Η διεύθυνση της πλατφόρμας πρακτόρων που θα χρησιμοποιεί ο server (*agencyAddress*)
- Flag που δείχνει αν ο server εκκινείται ως πρωτεύον (primary) ή ως εφεδρικός (backup) (*master*)
- Flag που δείχνει αν επιτρέπεται η σύνδεση εφεδρικού server (*allowBackup*)
- Μέγιστος αριθμός εφεδρικών servers που επιτρέπεται να συνδεθούν στον κύριο server (*maxBackupServers*)
- Η διεύθυνση του κύριου server για την περίπτωση εκκίνησης εφεδρικού server (*primaryServerAddress*)
- Το port στο οποίο ακούει ο κύριος server για την περίπτωση εκκίνησης εφεδρικού server (*primaryServerPort*)
- Μέγιστος αριθμός εκτελεστών εργασίας (task managers) ανά μηχανή εκτέλεσης

Παρακάτω δίνεται ένα παράδειγμα ενός αρχείου αρχικοποίησης ενός κύριου server.

*#Comment line – Primary server sample configuration file*
*#Server configuration - Required*
*serverAddress= 139.91.182.216*
*serverPort= 5892*
*serverName= Arion Server*
*serverInfo= Agent-based Runtime System for Workflow Execution*
*serverLogDir= /home/smardas/Arion/runtime/logs/*

*#Indicates whether server should enter recovering mode – Required flag*
*recovering= false*
*#It is taken into consideration only if recovering flag is set to true*

*recoverDir=*

*#Address of the agent platform to be contacted - Required*
*agencyAddress= socket://139.91.182.216:7000/arion-dl.ics.forth.gr/InformationDesk*

*#Primary/backup flag - Required*
*master= true*
*#Flag indicating whether a backup server is allowed - Required*
*allowBackup= true*
*#No of backup servers allowed - Required*
*maxBackupServers= 1*
*#It is taken into consideration only if server is in backup mode*
*primaryServerAddress=*
*primaryServerPort=*
*maxTaskManagersPerEngine=10*

Ο server μπορεί να εκκινηθεί με ενεργή τη λειτουργία σωσίματος συνεπών ενδιάμεσων καταστάσεων καθιστώντας έτσι εφικτή τη δυνατότητα συνέχισης της εκτέλεσης των λειτουργιών του έπειτα από κάποια αποτυχία από την τελευταία συνεπή κατάσταση που έχει σωθεί.

### 6.2.1.3 Listener

Ο listener είναι το μέρος του runtime συστήματος το οποίο αποτελεί το σύνδεσμο με τα άλλα μέρη (componets) του ARION, καθώς και με τους χρήστες. Συγκεκριμένα ακούει για εισερχόμενες αιτήσεις αφενός από τη Workflow Database του ARION και αφετέρου από τους χρήστες του runtime συστήματος που αλληλεπιδρούν με αυτό μέσω του web interface.

#### 6.2.1.3.1 Αιτήσεις από τη Workflow Database

Ο listener λαμβάνει από τη Workflow Database αιτήσεις για την εκτέλεση ροών εργασίας. Μία τέτοια αίτηση περιέχει την περιγραφή της προς εκτέλεση ροής εργασίας και επιπλέον την πληροφορία για την επιθυμητή κατάσταση εκτέλεσης

(αλληλεπιδραστική ή αυτόματη, βλ. §6.3). Στη συνέχεια ανατίθεται σε μία μηχανή εκτέλεσης η ευθύνη της εκτέλεσης της ροής εργασίας.

#### 6.2.1.3.2 Αιτήσεις από τους χρήστες

Ο listener λαμβάνει επίσης αιτήσεις από τους χρήστες. Οι αιτήσεις αυτές αφορούν ενέργειες αλληλεπίδρασης των χρηστών με το runtime σύστημα μέσω του web interface που χειρίζονται (βλ. §6.3.2.1). Οι αιτήσεις αυτές σχετίζονται είτε με λειτουργίες παρακολούθησης της πορείας εκτέλεσης μίας ροής εργασίας είτε με λειτουργίες επίδρασης πάνω στην πορεία εκτέλεσης. Συγκεκριμένα, οι εισερχόμενες αιτήσεις από τους χρήστες αφορούν:

**Λειτουργίες παρακολούθησης:**

- Τρέχουσα κατάσταση ροής εργασίας
- Ιστορικό εκτέλεσης ροής εργασίας

**Λειτουργίες επίδρασης:**

- Προσωρινή παύση εκτέλεσης ροής εργασίας / Συνέχιση εκτέλεσης ροής εργασίας
- Οριστική παύση εκτέλεσης ροής εργασίας
- Επανεκκίνηση ροής εργασίας
- Παροχή παραμέτρων για ένα πρόγραμμα της ροής εργασίας
- Παροχή επιλογής backtracking

### 6.2.1.4 Agent Management System

#### 6.2.1.4.1 Εισαγωγή

Το Agent Management System (AMS) αναλαμβάνει τη δημιουργία και την διαχείριση των κινούμενων πρακτόρων. Επιπλέον υλοποιεί ένα μηχανισμό επικοινωνίας που επιτρέπει την 2-way ανταλλαγή μηνυμάτων μεταξύ εκτελεστών εργασιών/AMS και κινούμενων πρακτόρων.

#### 6.2.1.4.2 Σύνδεση με agent πλατφόρμα

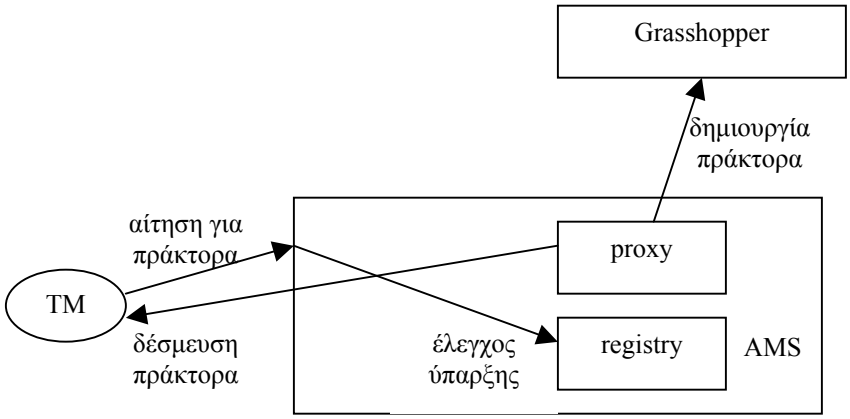
Το AMS είναι η μοναδική οντότητα του runtime συστήματος το οποίο επικοινωνεί με την agent πλατφόρμα (Grasshopper). Κατά την εκκίνηση του συστήματος το AMS λαμβάνει τη διεύθυνση της agent πλατφόρμας και αναλαμβάνει να δημιουργήσει ένα proxy αντικείμενο προς την πλατφόρμα αυτή χρησιμοποιώντας το API το οποίο παρέχει. Μέσω του proxy αντικειμένου το AMS και κατά συνέπεια το runtime σύστημα είναι σε θέση να αξιοποιήσει τη λειτουργικότητα που παρέχει η πλατφόρμα.

Σκοπίμως, επιλέξαμε να εκμεταλλευτούμε μόνο τις βασικές λειτουργίες της agent πλατφόρμας οι οποίες είναι αυτές της δημιουργίας και της μετανάστευσης των πρακτόρων και να αγνοήσουμε τις πιο εξειδικευμένες λειτουργίες που παρέχονται, όπως μηχανισμοί επικοινωνίας υποστηριζόμενοι από την πλατφόρμα, υλοποιώντας τους εντός του AMS. Η επιλογή αυτή έγινε προκειμένου το runtime σύστημα **να μην είναι εξαρτημένο από μία συγκεκριμένη agent πλατφόρμα** χρησιμοποιώντας μόνο core functions που είναι διαθέσιμες σε κάθε agent πλατφόρμα καθιστώντας με αυτόν τον τρόπο εύκολη τη μετάβαση σε κάποια άλλη πλατφόρμα. Μία τέτοια μετάβαση θα απαιτούσε απλώς την προσαρμογή του AMS στο API της νέας πλατφόρμας χωρίς την ανάγκη υλοποίησης επιπλέον λειτουργικότητας που προηγουμένως καλυπτόταν από την agent πλατφόρμα που είχε επιλεγεί. Κατά τη φάση ανάπτυξης του συστήματός μας πολλές από τις δυνατότητες που παρέχει η πλατφόρμα Grasshopper χρησιμοποιήθηκαν προκειμένου να επιταχυνθεί η διαδικασία ανάπτυξης και ελέγχου της ζητούμενης λειτουργικότητας, αλλά εν συνεχεία υλοποιήσαμε δικές μας λύσεις προκειμένου να επιτευχθεί ο σκοπός που αναφέρθηκε.

#### 6.2.1.4.3 Δημιουργία – Διαχείριση πρακτόρων

Το AMS δέχεται αιτήσεις από τους εκτελεστές εργασιών για χρήση κάποιου πράκτορα προκειμένου αυτός να εκτελέσει κάποια ενέργεια σε ένα απομακρυσμένο μηχάνημα. Η ενέργεια αυτή μπορεί να αφορά την εκτέλεση κάποιου προγράμματος ή τη μεταφορά κάποιου/ων αρχείου/ων. Με την παραλαβή μιας αίτησης το AMS ελέγχει για την ύπαρξη κάποιου ανενεργού πράκτορα στο μηχάνισμα προορισμού. Ο έλεγχος γίνεται με τη χρήση των εσωτερικών δομών δεδομένων που διατηρεί το AMS για τους πράκτορες που έχουν δημιουργηθεί. Αν δε βρεθεί κάποιος ανενεργός πράκτορες στο μηχάνημα

προορισμού τότε δημιουργείται νέος πράκτορας προκειμένου να του ανατεθεί η προς εκτέλεση ενέργεια. Σε κάθε περίπτωση το AMS επιστρέφει στον εκτελεστή εργασίας πληροφορία σχετικά με τον πράκτορα που δεσμεύεται για λογαριασμό του (Σχήμα 6.2). Το AMS πληροφορείται για την τρέχουσα κατάσταση των πρακτόρων μέσω του μηχανισμού επικοινωνίας που έχει υλοποιηθεί.

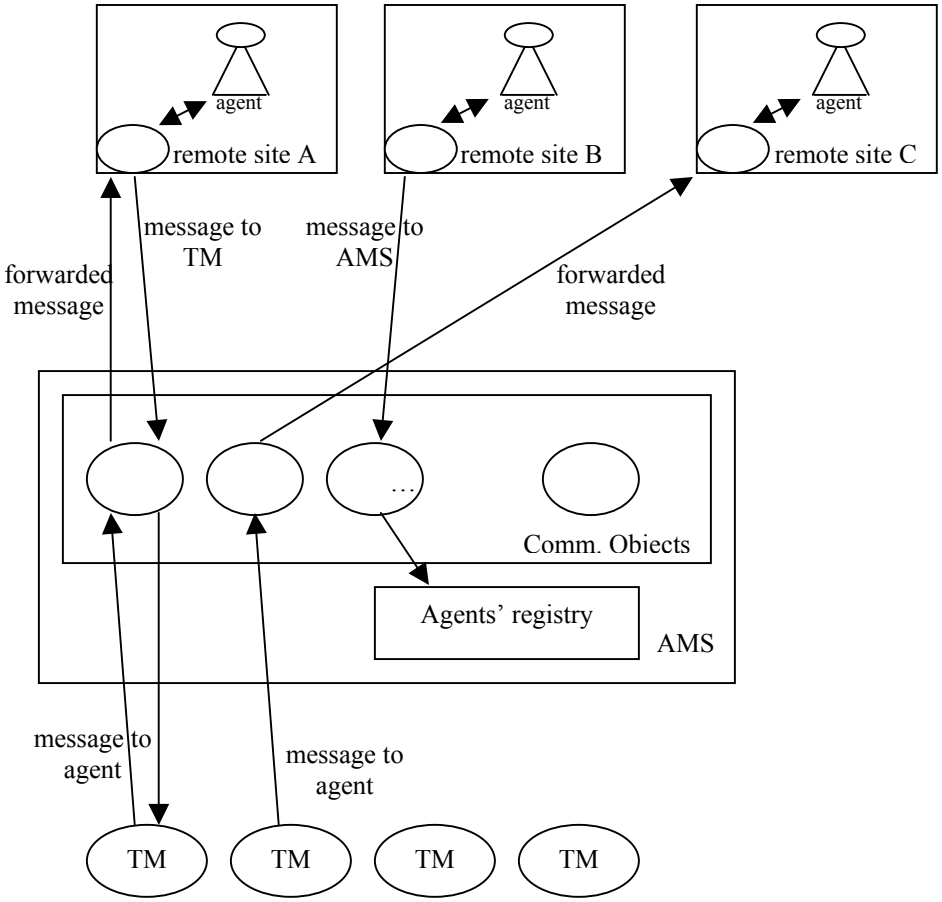



**Σχήμα 6.2.** Αίτηση για χρήση πράκτορα από έναν εκτελεστή εργασίας

### 6.2.1.4.4 Μηχανισμός επικοινωνίας

### 6.2.1.4.4.1 Εισαγωγή

Το AMS σε συνεργασία με τους πράκτορες υλοποιούν ένα μηχανισμό επικοινωνίας που καθιστά δυνατή την 2-way επικοινωνία των εκτελεστών εργασιών, αλλά και του ίδιου του AMS με τους κινούμενους πράκτορες. Ο μηχανισμός στηρίζεται στη χρήση αντικειμένων επικοινωνίας (communication objects) τα οποία υλοποιούν λειτουργικότητα προώθησης εισερχόμενων-εξερχόμενων μηνυμάτων από και προς τους πράκτορες αντίστοιχα. Το AMS διατηρεί μία συλλογή από communication objects τα οποία μπορούν να εξυπηρετήσουν τις ανάγκες επικοινωνίας μίας ή περισσότερων οντοτήτων. Επιπλέον, οι πράκτορες όταν μεταναστεύουν σε κάποιο απομακρυσμένο μηχάνημα εγκαθιστούν αυτομάτως το δικό τους μέρος του μηχανισμού το οποίο και λειτουργεί ανεξάρτητα από την agent πλατφόρμα που υποδέχεται τους πράκτορες. Στη συνέχεια ενημερώνουν το AMS για τη θέση τους και είναι σε θέση να δεχτούν εισερχόμενα μηνύματα.




**Σχήμα 6.3.** Μηχανισμός επικοινωνίας

Όλοι οι εκτελεστές εργασίας προκειμένου να χρησιμοποιήσουν τη λειτουργικότητα που παρέχει ένα communication object ζητούν από το AMS να κάνουν registration σε ένα communication object. Η αίτηση ενός εκτελεστή εργασίας για registration σε ένα communication object στέλνεται στο AMS το οποίο φροντίζει στη συνέχεια να επιλέξει ένα communication object και αναθέτει σε αυτό την υποστήριξη των αναγκών επικοινωνίας του εκτελεστή εργασίας. Η πληροφορία για το communication object που επιλέχθηκε επιστρέφεται στον εκτελεστή εργασίας μέσω της οποίας μπορεί να επικοινωνήσει με τους πράκτορες που έχει δεσμεύσει, ενώ οι πράκτορες είναι σε θέση να επικοινωνήσουν με το runtime σύστημα μέσω αυτού του communication object.

Ένα communication object υποστηρίζει την επικοινωνία μόνο των οντοτήτων που έχουν γίνει registered σε αυτό και αγνοεί τις αιτήσεις για επικοινωνία που προέρχονται από

άλλες οντότητες. Το ίδιο το AMS είναι registered σε όλα τα communication objects προκειμένου να μπορεί να λαμβάνει πληροφορίες από όλα τα communication objects και συνεπώς από όλους τους πράκτορες .

#### 6.2.1.4.4.2 Εισερχόμενα μηνύματα

Ένα communication object μπορεί να δέχεται εισερχόμενα μηνύματα από τους κινούμενους πράκτορες που επιθυμούν να επικοινωνήσουν με το runtime σύστημα. Τα μηνύματα αυτά απευθύνονται σε κάποια οντότητα και συγκεκριμένα είτε στο AMS είτε στον εκτελεστή εργασίας ο οποίος έχει δεσμεύσει τον πράκτορα. Το communication object με το οποίο επικοινωνεί ο πράκτορας έχει δωθεί ως παράμετρος σε αυτόν τον πράκτορα από τον εκτελεστή εργασίας όταν του ανατέθηκε η ενέργεια που εκτελεί και είναι το communication object στο οποίο έχει κάνει register ο εκτελεστής εργασίας. Όπως αναφέρθηκε, στο communication object είναι registered και το AMS οπότε ο πράκτορας μπορεί να στέλνει μηνύματα και στις δύο αυτές οντότητες.

- **Μήνυμα προς τον εκτελεστή εργασίας:**

Σε ένα μήνυμα προς τον εκτελεστή εργασίας ο πράκτορας αναφέρει την πρόοδο της ενέργειας που έχει αναλάβει. Προκειμένου για έναν πράκτορα εκτέλεσης προγράμματος (Task Execution Agent) δηλώνει την αρχή ή το τέλος της εκτέλεσης του απομακρυσμένου προγράμματος. Οι πράκτορες μεταφοράς αρχείων (File Server Agent και File Client Agent) δηλώνουν την αρχή και το τέλος αποστολής του/ων αρχείου/ων αντίστοιχα.

- **Μήνυμα προς το AMS:**

Σε ένα μήνυμα προς το AMS ο πράκτορας ενημερώνει το AMS για την τρέχουσα θέση και κατάστασή του. Αυτό γίνεται στις εξής περιπτώσεις:

- Ο πράκτορας φτάνει σε απομακρυσμένο μηχάνημα (status = IDLE ή status = OCCUPIED, αναλόγως αν του έχει ανατεθεί η εκτέλεση μίας ενέργειας ή όχι).
- Στον πράκτορα ανατίθεται κάποια εργασία (status = OCCUPIED)
- Ο πράκτορας τερματίζει την εργασία του (status = IDLE)
- Ο πράκτορας είναι έτοιμος να μεταναστεύσει σε ένα απομακρυσμένο μηχάνημα. (status = MOVING)

Με τα μηνύματα που λαμβάνει το AMS από τους πράκτορες ενημερώνει τις δομές του σχετικά με την τρέχουσα θέση και την κατάσταση των πρακτόρων.

#### 6.2.1.4.4.3 Εξερχόμενα μηνύματα

Ένας εκτελεστής εργασίας που έχει γίνει registered σε κάποιο communication object μπορεί μέσω αυτού να επικοινωνήσει με τους πράκτορες που χρησιμοποιεί προκειμένου να τους αναθέσει την εκτέλεση κάποιας ενέργειας. Ο εκτελεστής εργασίας ζητά από το communication object την αποστολή ενός μηνύματος προς ένα συγκεκριμένο πράκτορα και το communication object με τη σειρά του αναλαμβάνει την προώθηση του μηνύματος προς τον πράκτορα. Το communication object ζητά την τοποθεσία του πράκτορα από το AMS και στη συνέχεια προωθεί το μήνυμα του task manager στον πράκτορα. Στον αποστολέα του μηνύματος επιστρέφεται επιβεβαίωση για την επιτυχή αποστολή του μηνύματος ή κατάλληλο μήνυμα λάθους.

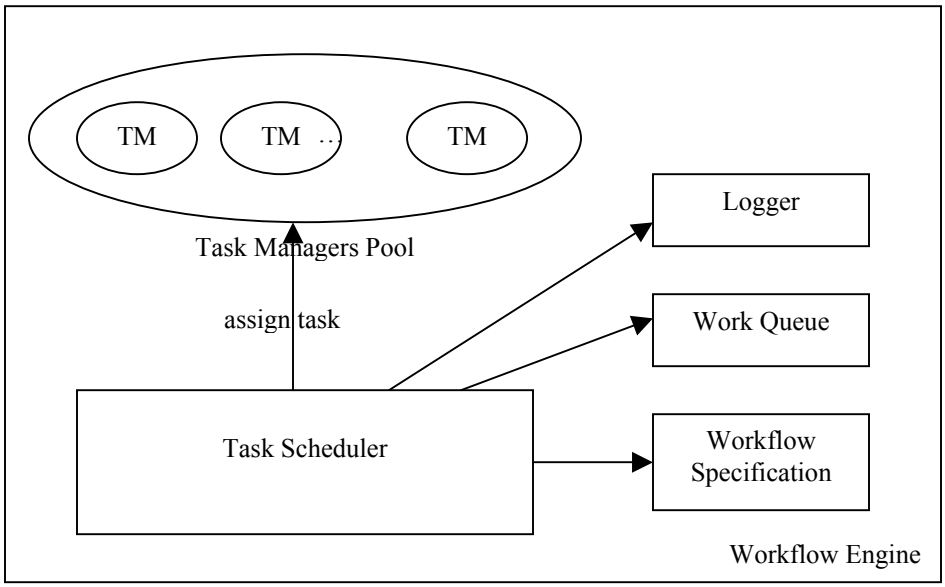
### 6.2.2 Μηχανή Εκτέλεσης Ροών Εργασίας

Η εκτέλεση μίας ροής εργασίας ανατίθεται σε μια μηχανή εκτέλεσης (workflow engine). Η μηχανή αυτή αναλαμβάνει να περατώσει την εκτέλεση της ροής εργασίας, ενώ καθιστά δυνατή και την παρακολούθηση της πορείας εκτέλεσής της.

#### 6.2.2.1 Ορισμός

Μια μηχανής εκτέλεσης είναι το κομμάτι ενός συστήματος ροών εργασίας που γνωρίζει όλες τις απαιτούμενες διαδικασίες που πρέπει να εκτελεστούν, τα επιμέρους βήματα σε μία διαδικασία καθώς και τους κανόνες που έχουν οριστεί για κάθε βήμα. Η μηχανή εκτέλεσης αποφασίζει πότε η εκτέλεση της/ων επόμενης/ων εργασίας/ών είναι σε θέση να ξεκινήσουν.

Κάθε μηχανή εκτέλεσης είναι υπεύθυνη για τη διαχείριση ενός αντικειμένου ροής εργασίας (workflow instance). Η μηχανή εκτέλεσης αποτελεί μια state transition μηχανή, όπου οι αλλαγές στις καταστάσεις έρχονται ως αντίδραση σε εξωτερικά γεγονότα π.χ. ολοκλήρωση ενός βήματος ή σε συγκεκριμένες εντολές ελέγχου που παίρνονται από το χρήστη, π.χ. μετάβαση σε κάποιο προηγούμενο βήμα της εκτέλεσης.




**Σχήμα 6.4.** Μηχανή εκτέλεσης ροών εργασίας

### 6.2.2.2 Δρομολογητής Εργασιών

Ο δρομολογητής εργασιών (task scheduler) είναι το κυριότερο κομμάτι μίας μηχανής εκτέλεσης. Αποτελεί την οντότητα που παίρνει τις αποφάσεις δρομολόγησης των εργασιών προς εκτέλεση. Ο δρομολογητής εργασιών εγγυάται τη διατήρηση όλων των κανόνων που ορίζονται στην περιγραφή της προς εκτέλεσης ροής εργασίας, π.χ. σχέσεις μετά-από κλπ.

#### 6.2.2.2.1 Λειτουργία

Η λειτουργία που εξυπηρετεί ο δρομολογητής εργασιών είναι η δρομολόγηση της εκτέλεσης των επιμέρους εργασιών (tasks) τα οποία απαρτίζουν την εκτελούμενη ροή εργασίας. Η δρομολόγηση που γίνεται βασίζεται στη διαθέσιμη πληροφορία για την κατάσταση εκτέλεσης των επιμέρους εργασιών και στις σχέσεις εξάρτησης που υπάρχει μεταξύ των διαφόρων εργασιών όπως αυτές εκφράζονται στη δομή της ροής εργασίας. Συγκεκριμένα, η πληροφορία της τρέχουσας κατάστασης των επιμέρους εργασιών μίας ροής εργασίας διατηρείται σε εσωτερικές δομές του δρομολογητή εργασιών και περιλαμβάνει τις εξής κατηγορίες εργασιών:

- **Εργασίες στην ουρά (queued).** Είναι οι εργασίες που βάσει της τρέχουσας κατάστασης των εργασιών και της περιγραφής της ροής εργασίας μπορούν να εκτελεστούν, αλλά δεν έχουν ανατεθεί σε κάποιον εκτελεστή εργασίας ακόμα. Μια εργασία εισάγεται στην ουρά όταν όλες οι προϋποθέσεις εκτέλεσής της πληρούνται, δηλ. όταν όλες οι προηγούμενες εργασίες από τις οποίες εξαρτάται αυτή έχουν τερματιστεί. Ο τερματισμός μίας εργασίας γνωστοποιείται στον δρομολογητή εργασιών από τον αντίστοιχο εκτελεστή εργασίας.

- **Αρχικοποιημένες (initiated).** Είναι οι εργασίες για οποίες ο δρομολογητής εργασιών έχει δεσμεύσει κάποιον εκτελεστή εργασίας προκειμένου να τις εκτελέσει, αλλά η εκτέλεση της εργασίας δεν έχει ξεκινήσει. Ο δρομολογητής εργασιών «τραβάει» τις εργασίες αυτές από την ουρά και ορίζει εκτελεστές εργασίας για την εκτέλεσή τους. Ο εκτελεστής εργασίας που θα αναλάβει την εκτέλεση μιας εργασίας επιλέγεται από ένα pool εκτελεστών εργασίας.

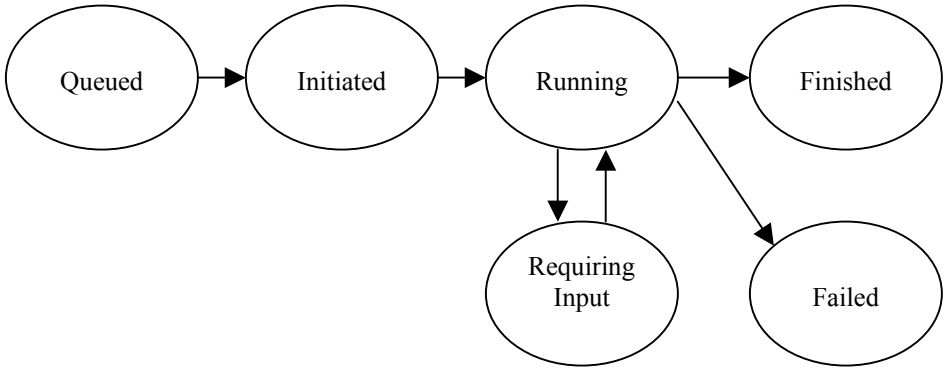
- **Εκτελούμενες (running).** Είναι οι εργασίες που είναι στη φάση εκτέλεσης. Ο εκτελεστής εργασίας που έχει αναλάβει την εκτέλεσή της εκτελεί τα επιμέρους βήματα που απαιτούνται για την περάτωση της εργασίας.

- **Απαιτούν είσοδο από το χρήστη (requiring input).** Είναι οι εργασίες που απαιτούν κάποια είσοδο προκειμένου να μπορέσει να συνεχιστεί η εκτέλεσή τους. Μια εργασία μπορεί να απαιτεί την εισαγωγή κάποιων παραμέτρων από το χρήστη προκειμένου να εκτελεστεί κάποιο πρόγραμμα που αποτελεί μέρος της ροής εργασίας ή την επιλογή από το χρήστη για την επιθυμητή πορεία εκτέλεσης της ροής εργασίας (δυνατότητα backtracking, βλ. 6.3.2.1.2). Ο δρομολογητής εργασιών ενημερώνεται για την απαίτηση εισόδου από μία εργασία από τον αντίστοιχο εκτελεστή εργασίας.

- **Τερματισμένες (finished).** Είναι οι εργασίες των οποίων η εκτέλεση έχει περατωθεί επιτυχώς. Για τον τερματισμό εκτέλεσης μίας εργασίας ο δρομολογητής εργασιών ενημερώνεται από τον αντίστοιχο εκτελεστή εργασίας. Βάσει των τερματισμένων

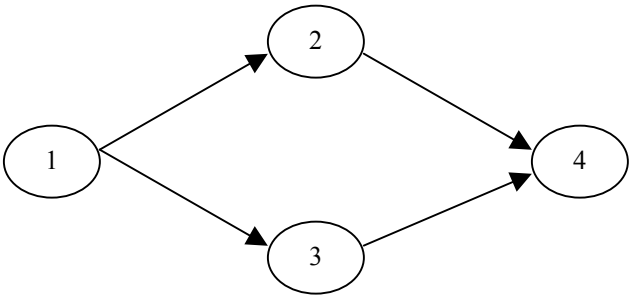
εργασιών ο δρομολογητής εργασιών μπορεί να λάβει τις αποφάσεις δρομολόγησης των επόμενων εργασιών.

- **Αποτυχημένες (failed).** Είναι οι εργασίες των οποίων η εκτέλεση έχει αποτύχει. Η αποτυχία μπορεί να οφείλεται σε μία σειρά λόγων, όπως ανυπαρξία των απαιτούμενων πόρων, π.χ. απουσία του εκτελέσιμου προγράμματος από το απομακρυσμένο μηχάνημα, αδυναμία μεταφοράς πράκτορα λόγω προβλημάτων δικτύου κ.α. Όταν μία εργασία εισέλθει στην αποτυχημένη κατάσταση ο δρομολογητής εργασιών θα επιχειρήσει να επαναδρομολογήσει την εργασία αυτή.




**Σχήμα 6.5.** Μεταβάσεις καταστάσεων σε μία εργασία

Οι σχέσεις εξάρτησης αφορούν το control flow της ροής εργασίας, δηλ. τις συνθήκες που πρέπει να ισχύουν προκειμένου μία εργασία να μπορεί να δρομολογηθεί. Έτσι, μία εργασία μπορεί να δρομολογηθεί μόνο όταν οι εργασίες με τις οποίες εξαρτάται έχουν τερματίσει επιτυχώς. Ο δρομολογητής εργασιών διατηρεί τις σχέσεις εξάρτησης όπως εκφράζονται στην περιγραφή της ροής εργασίας, ενώ ταυτόχρονα επιδιώκει την επίτευξη του μέγιστου βαθμού ταυτόχρονης εκτέλεσης εργασιών που είναι δυνατή.




**Σχήμα 6.6.** Παράδειγμα ροής εργασίας με τις συνδέσεις του control flow

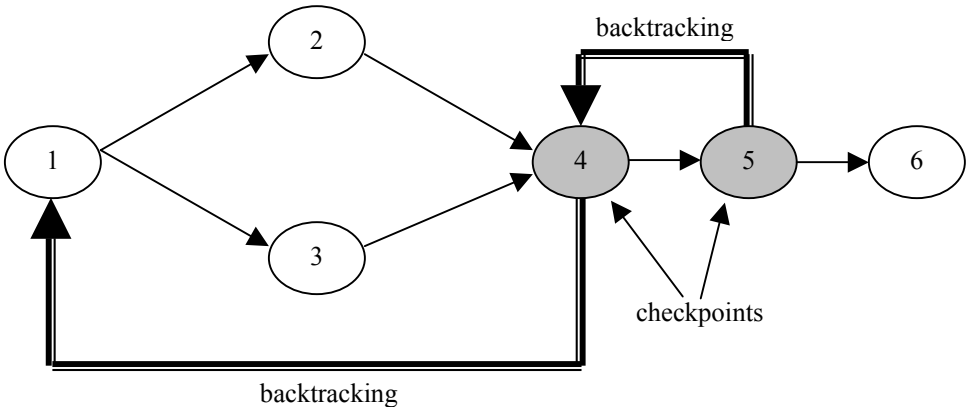
Για τη ροή εργασίας του σχήματος 6.6 μπορούμε να συμπεράνουμε βάσει των συνδέσεων του control flow:

- η εργασία 1 μπορεί να ξεκινήσει αμέσως
- οι εργασίες 2, 3 μπορούν να ξεκινήσουν **μόνο** εφόσον έχει ολοκληρωθεί η εργασία 1
- η εργασία 4 μπορεί να ξεκινήσει **μόνο** εφόσον έχουν ολοκληρωθεί οι εργασίες 2 και 3.

Η δρομολόγηση της εκτέλεσης μιας εργασίας προκαλεί τη δέσμευση ενός εκτελεστή εργασίας μέσα από ένα σύνολο διαθέσιμων εκτελεστών εργασίας που βρίσκοται στο pool εκτελεστών εργασίας το οποίο θα περιγράψουμε στην επόμενη παράγραφο.

### 6.2.2.2.2 Υποστήριξη backtracking

Η έννοια του backtracking αφορά τη δυνατότητα που παρέχει το runtime σύστημα για τη μεταφορά της ροής εκτέλεσης μιας ροής εργασιών σε κάποιο προηγούμενο στάδιο αντί της συνέχισης του κανονικού execution flow. Χάρη σε αυτό το υποστηριζόμενο χαρακτηριστικό, ο χρήστης μπορεί να ορίσει σε μία ροή εργασίας κάποιες εργασίες να αποτελούν κομβικά σημεία της (checkpoints), δηλώνοντας με αυτόν τον τρόπο πως κατά τη διάρκεια της εκτέλεσης της ροής εργασίας όταν τερματίζεται η εκτέλεση μιας τέτοιας εργασίας (checkpoint) θα δίνεται στο χρήστη η δυνατότητα να επιλέγει αν επιθυμεί να συνεχιστεί κανονικά η εκτέλεση με τις επόμενες εργασίες ή να επιστρέψει η εκτέλεση σε κάποιο προηγούμενο στάδιο.




**Σχήμα 6.7.** Παράδειγμα ροής εργασίας με τις συνδέσεις του control flow και backtracking

Στο σχήμα 6.7 παρουσιάζεται μία ροή εργασίας που πέρα από τις συνδέσεις control flow περιέχει και δύο συνδέσεις backtracking (έντονες γραμμές). Μία τέτοια ροή εργασίας συνεπάγεται τα εξής:

- η εκτέλεση ξεκινά από την εργασία 1
- οι εργασίες 2 και 3 μπορούν να ξεκινήσουν μόνο όταν τερματίσει η 1
- η εργασία 4 μπορεί να ξεκινήσει μόνο όταν έχουν τερματίσει και η 2 και η 3.
- όταν τερματιστεί η εργασία 4 **ο χρήστης θα έχει τη δυνατότητα να επιλέξει** αν θα συνεχιστεί η εκτέλεση με την εργασία 5 ή αν θα επιστρέψει η εκτέλεση πίσω στην εργασία 1. Αν επιλέξει να συνεχιστεί κανονικά η εκτέλεση τότε μπορεί να ξεκινήσει η εργασία 5, ενώ αν επιλέξει να επιστρέψει στην 1 τότε θα εκτελεστούν εκ νέου οι εργασίες 1-4 και όταν εκτελεστεί η 4 θα παρουσιαστεί ξανά στο χρήστη η παραπάνω επιλογή.
- όταν τερματιστεί η εργασία 5 **ο χρήστης θα έχει τη δυνατότητα να επιλέξει** αν θα συνεχιστεί η εκτέλεση με την εργασία 6 ή αν θα επιστρέψει η εκτέλεση στην εργασία 4. Αν ο χρήστης επιλέξει να εκτελεστεί η εργασία 6 τότε η 6 μπορεί να ξεκινήσει, ενώ αν επιλέξει να εκτελεστεί η εργασία 4 τότε ισχύει ότι γράφεται παραπάνω για την εργασία 4.

Μέσω του web interface ο χρήστης μπορεί όταν συναντηθεί ένα checkpoint κατά την εκτέλεση μίας ροής εργασίας να επιλέξει προς τα πού επιθυμεί να κινηθεί η πορεία εκτέλεσης.

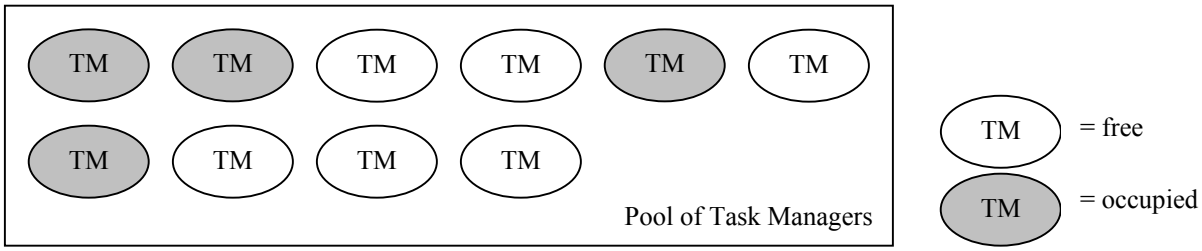
Ο σκοπός για τον οποίο θελήσαμε να υποστηρίξουμε ένα τέτοιο χαρακτηριστικό είναι η επιθυμία των χρηστών για μια τέτοια λειτουργικότητα η οποία επιτρέπει το calibration των παραμέτρων εισόδου έτσι ώστε να επιτευχθεί ένα επιθυμητό αποτέλεσμα. Η εξεταζόμενη λειτουργικότητα που παρέχεται επιτρέπει στο χρήστη να εκτελέσει κάποιους κύκλους εντός μιας ροής εργασίας ως την επίτευξη ενός «επιθυμητού» αποτελέσματος. Έτσι, μπορεί να πραγματοποιηθεί «χειροκίνητα» μια διαδικασία calibration των αποτελεσμάτων μέσα από έναν κύκλο επαναλαμβανόμενων εκτελέσεων κάποιων προγραμμάτων με τη χρήση πιθανότατα διαφορετικών παραμέτρων σε κάθε κύκλο.

Αυτό το χαρακτηριστικό είναι αρκετά σημαντικό καθώς το calibration των παραμέτρων εισόδου επιστημονικών προγραμμάτων είναι σημαντικό κομμάτι του scientific computing και λαμβάνει χώρα σε αλληλεπίδραση με το χρήστη.

Ο δρομολογητής εργασιών υποστηρίζει τη λειτουργία του backtracking, δηλ. της δυνατότητας επιστροφής της πορείας εκτέλεσης μίας ροής εργασίας σε κάποιο προηγούμενο βήμα.

#### 6.2.2.3 Pool Εκτελεστών Εργασίας

Το pool εκτελεστών εργασίας αποτελεί μία συλλογή εκτελεστών εργασίας που είναι στη διάθεση του δρομολογητή εργασιών προκειμένου αυτός να τους αναθέτει την εκτέλεση εργασιών. Ο δρομολογητής εργασιών όταν θελήσει να δρομολογήσει κάποια εργασία και να την αναθέσει σε κάποιον εκτελεστή εργασίας επιλέγει κάποιον ανενεργό εκτελεστή εργασίας από τη συλλογή προκειμένου να του αναθέσει την εργασία αυτή. Αν δεν υπάρχει διαθέσιμος εκτελεστής εργασίας τότε ο δρομολογητής εργασιών είτε μπλοκάρεται είτε δημιουργεί ένα νέο εκτελεστή εργασίας στον οποίο και αναθέτει την εργασία. Η επιλογή αυτή εξαρτάται από τις ρυθμίσεις που έχουν δωθεί στο αρχείο αρχικοποίησης κατά την εκκίνηση του συστήματος. (βλ. §6.2.1.2)




**Σχήμα 6.8.** Pool εκτελεστών εργασίας

Η χρήση ενός pool από εκτελεστές εργασίας επιλέχθηκε για δύο λόγους:

- Δεν απαιτείται η δημιουργία ενός εκτελεστή εργασίας κάθε φορά που χρειάζεται να ανατεθεί μια εργασία από τον δρομολογητή εργασιών. Κατά την εκκίνηση του συστήματος δημιουργούνται οι εκτελεστές εργασίας και είναι διαθέσιμοι κατά τη διάρκεια της εκτέλεσης μίας ροής εργασίας. Επίσης, ένας εκτελεστής εργασίας που περατώνει μία εργασία μπορεί να επαναχρησιμοποιηθεί για την εκτέλεση κάποιας επόμενης εργασίας.

- Επιτρέπεται ο καλύτερος έλεγχος του φόρτου μίας μηχανής εκτέλεσης. Έχοντας όλους τους εκτελεστές εργασίας σε μία συλλογή μπορούμε άμεσα να διαπιστώσουμε το φόρτο μίας μηχανής εκτέλεσης. Μεγάλος αριθμός απασχολούμενων εκτελεστών εργασίας συνεπάγεται μεγάλο φόρτο στη μηχανή εκτέλεσης (και το αντίστροφο). Επιπλέον, μπορεί να τεθεί ένα άνω όριο στο φόρτο που θα προκαλεί μία μηχανή εκτέλεσης ορίζοντας έναν αριθμό διαθέσιμων εκτελεστών εργασίας κι απαγορεύοντας τη δυνατότητα δημιουργίας νέων.

### 6.2.2.4 Εκτελεστής Εργασίας

Ένας εκτελεστής εργασίας αναλαμβάνει την διεκπεραίωση μιας εργασίας (task) που του αναθέτει ο δρομολογητής εργασιών. Η εργασία αυτή αφορά συνήθως την εκτέλεση ενός προγράμματος που βρίσκεται σε μία απομακρυσμένη μηχανή. Η πληροφορία σχετικά με την εκτέλεση μίας εργασίας αποτελεί μέρος της ροής εργασίας και στέλνεται στον εκτελεστή εργασίας όταν του ανατίθεται η διεκπεραίωση μίας εργασίας. Συγκεκριμένα, η

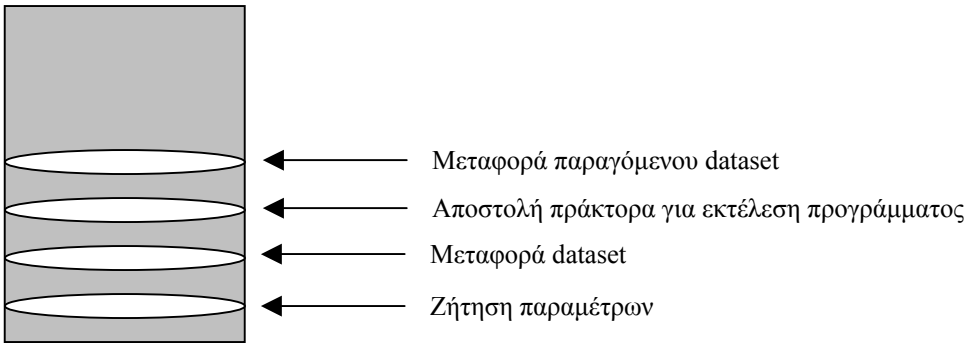
πληροφορία που δίνεται σε έναν εκτελεστή εργασίας για την εκτέλεση μίας εργασίας περιλαμβάνει:

- Τα ονόματα των αρχείων εισόδου που χρειάζονται για την εκτέλεση του προγράμματος καθώς και το μηχάνημα στο οποίο μπορούν αυτά να βρεθούν. Τα απαιτούμενα αρχεία εισόδου μπορεί να είναι η έξοδος προγραμμάτων που είχαν εκτελεστεί σε προηγούμενο στάδιο.

- Στοιχεία σχετικά με το πρόγραμμα που χρειάζεται να εκτελεστεί, όπως διεύθυνση μηχανήματος, όνομα εκτελέσιμου προγράμματος, παράμετροι που απαιτούνται για την εκτέλεσή του, ρυθμίσεις περιβάλλοντος εκτέλεσης, τυχόν απαίτηση για εκτέλεση του προγράμματος σε απομόνωση (μη ταυτόχρονη εκτέλεση του ίδιου προγράμματος από δύο διαφορετικές διεργασίες στο ίδιο μηχάνημα).

- Τα αρχεία εξόδου που θα παραχθούν και προαιρετικά το μηχάνημα στο οποίο θα πρέπει αυτά να μεταφερθούν.

Ένας εκτελεστής εργασίας όταν αναλαμβάνει την εκτέλεση μίας εργασίας δημιουργεί μια ουρά ενεργειών βάσει της περιγραφής της εργασίας που λαμβάνει από τον δρομολογητή εργασιών. Αυτή περιλαμβάνει όλες τις επιμέρους ενέργειες που πρέπει να εκτελεστούν έτσι ώστε να έχει εκτελεστεί όλη η εργασία. Η χρήση της ουράς ενεργειών βοηθά σημαντικά στην υιοθέτηση ενός συνεπούς μοντέλου εκτέλεσης της ροής εργασίας όπως αναλυτικά θα εξηγηθεί στην παράγραφο §6.4.2.




**Σχήμα 6.9.** Ουρά ενεργειών

#### 6.2.2.4.1 Εκτέλεση της εργασίας

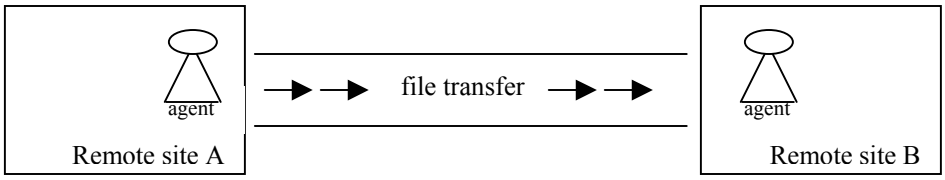
Όπως αναφέρθηκε η εκτέλεση της εργασίας μπορεί να αφορά μια σειρά ενεργειών που περιλαμβάνουν τη ζήτηση παραμέτρων εκτέλεσης, τη μεταφορά αρχείων, την κλήση ενός προγράμματος,

##### 6.2.2.4.1.1 Ζήτηση παραμέτρων

Ένα πρόγραμμα του οποίου η εκτέλεση έχει ανατεθεί σε έναν εκτελεστή εργασίας μπορεί να απαιτεί τη χρήση κάποιων παραμέτρων για την εκτέλεσή του. Οι παράμετροι αυτές αφορούν τις αρχικές παραμέτρους που λαμβάνει ένα πρόγραμμα κατά την κλήση του και είναι αντίστοιχες αυτών που θα δίνονταν αν η κλήση γινόταν μέσω γραμμής εντολών. Μία τέτοια απαίτηση έχει ως αποτέλεσμα ο εκτελεστής εργασίας να αποστείλλει στον δρομολογητή εργασιών αίτηση όπου θα ζητά τις απαιτούμενες παραμέτρους. Εφόσον, η εκτέλεση γίνεται σε κατάσταση αλληλεπιδραστικής λειτουργίας (βλ. §6.3.2) τότε ο εκτελεστής εργασίας τίθεται σε αναμονή ως ότου δωθούν οι ζητούμενες παράμετροι από το χρήστη μέσω του web interface. Αν η εκτέλεση γίνεται σε κατάσταση αυτόματης λειτουργίας ο δρομολογητής εργασιών παρέχει στον εκτελεστή εργασίας τις παραμέτρους οι οποίες έχουν δωθεί εξ'αρχής. Με την παροχή των παραμέτρων μέσω του δρομολογητή εργασίας ο εκτελεστής εργασίας επανέρχεται σε κατάσταση εκτέλεσης και προχωρά στην εκτέλεση των επόμενων ενεργειών.

##### 6.2.2.4.1.2 Μεταφορά αρχείων

Ένας εκτελεστής εργασίας αναλαμβάνει τη μεταφορά αρχειών από ένα μηχάνημα σε ένα άλλο, εφόσον το απαιτεί η περιγραφή της ροής εργασίας. Συγκεκριμένα, μεταφορά αρχείων μπορεί να γίνει τόσο πριν από την εκτέλεση ενός προγράμματος προκειμένου να μεταφερθούν κάποια αρχεία εισόδου από ένα απομακρυσμένο μηχάνημα στο μηχάνημα όπου είναι εγκατεστημένο το πρόγραμμα, όσο και μετά την εκτέλεση ενός προγράμματος για τη μεταφορά κάποιων αρχείων εξόδου σε κάποιο άλλο μηχάνημα. Η λειτουργικότητα αυτή επιτρέπει π.χ. τη μεταφορά κάποιων αρχείων εξόδου ενός προγράμματος στο μηχάνημα όπου βρίσκεται ένα πρόγραμμα που θα εκτελεστεί σε κάποιο επόμενο βήμα της ροής εργασίας (οπότε και τα αρχεία αυτά θα αποτελέσουν είσοδο για αυτό).




**Σχήμα 6.10.** Μεταφορά αρχείου με χρήση πρακτόρων

Η μεταφορά αρχείων από ένα μηχάνημα σε κάποιο άλλο επιτυγχάνεται μέσω της χρήσης κινούμενων πρακτόρων οι οποίοι κι αναλαμβάνουν τη μεταφορά του/ων αρχείου/ων. Συγκεκριμένα, ο εκτελεστής εργασίας αναθέτει σε δύο πράκτορες την εκτέλεση της συγκεκριμένης ενέργειας παρέχοντας τους την απαιτούμενη πληροφορία σχετικά με τα ονόματα, την τοποθεσία των αρχείων κλπ. και στη συνέχεια οι δύο πράκτορες μεταναστεύουν στα μηχανήματα προέλευσης και προορισμού αντίστοιχα και διεκπεραιώνουν τη ζητούμενη ενέργεια. Η μεταφορά των δεδομένων φροντίζεται ώστε να είναι ασφαλής (χρήση ασφαλούς καναλιού), ενώ για την εγκαθίδρυση της επικοινωνίας των πρακτόρων προηγείται πιστοποίηση της ταυτότητας του άλλου μέλους κι από τις δύο πλευρές. Αφού ολοκληρωθεί επιτυχώς η μεταφορά των αρχείων ο πράκτορας που βρίσκεται στο μηχάνημα προορισμού ενημερώνει τον σχετικό εκτελεστή εργασίας για την ολοκλήρωση της μεταφοράς, αλλιώς αναφέρει τα σχετικά λάθη που προκλήθηκαν (αδυναμία αποστολής, timeout κ.α.). Η επικοινωνία μεταξύ πρακτόρων κι εκτελεστή εργασίας γίνεται μέσω των communication objects που παρουσιάστηκαν στην παράγραφο §6.2.1.4.4.

### 6.2.2.4.1.3 Κλήση προγράμματος

Η εκτέλεση ενός προγράμματος σε ένα απομακρυσμένο μηχάνημα πραγματοποιείται με τη χρήση ενός κινούμενου πράκτορα. Ένας εκτελεστής εργασίας που η ουρά ενεργειών του περιλαμβάνει την κλήση ενός προγράμματος χρησιμοποιεί έναν κινούμενο πράκτορα τον οποίο αρχικοποιεί με τα απαραίτητα στοιχεία που αφορούν την εκτέλεση του προγράμματος. Στη συνέχεια ο πράκτορας μεταναστεύει στο μηχάνημα προορισμού και εκτελεί τοπικά το σχετικό πρόγραμμα με τις παραμέτρους που τυχόν έχουν δωθεί. Ο πράκτορας ενημερώνει τον εκτελεστή εργασίας τόσο για την εκκίνηση της εκτέλεσης του προγράμματος όσο και για την επιτυχή περάτωση της εκτέλεσης. Στην περίπτωση που

συμβεί κάποια σφάλμα κατά τη διάρκεια εκτέλεσης ο πράκτορας αποστέλλει κατάλληλο μήνυμα λάθους στον εκτελεστή εργασίας.

## 6.3 Εκτέλεση Ροών Εργασίας

### 6.3.1 Εισαγωγή

Το runtime σύστημα υποστηρίζει την εκτέλεση ροών εργασίας σε δύο διαφορετικές καταστάσεις εκτέλεσης. Συγκεκριμένα, οι δυνατές καταστάσεις εκτέλεσης είναι οι εξής:

- **Αλληλεπιδραστική εκτέλεση**, όπου ο χρήστης μπορεί να αλληλεπιδρά με τη μηχανή εκτέλεσης της ροής εργασίας κατά τη διάρκεια της εκτέλεσης παρέχοντας παραμέτρους για την εκτέλεση κάποιων προγραμμάτων ή και παρεμβαίνοντας στην πορεία εκτέλεσής της.

- **Προγραμματισμένη εκτέλεση**, όπου ο χρήστης δεν παρεμβαίνει στη διάρκεια εκτέλεσης της ροής εργασίας παρέχοντας εξ'αρχής όλες τις απαιτούμενες παραμέτρους για την εκτέλεσή της. Αυτή η κατάσταση λειτουργίας επιτρέπει την αυτοματοποιημένη εκτέλεση μιας ροής εργασίας όπου δεν απαιτείται σε καμία χρονική στιγμή η παροχή κάποιας εξωτερικής εισόδου από το χρήστη. Επιπλέον, σε αυτήν την κατάσταση λειτουργίας είναι εφικτός ο ορισμός ενός αριθμού ροών εργασίας προς εκτέλεση, όπου η περιγραφή της ροής εργασίας παραμένει η ίδια, ενώ οι αρχικές παραμέτροι εκτέλεσης διαφοροποιούνται επιτρέποντας την εκτέλεση της ίδιας ροής εργασίας με τη χρήση ενός εύρους παραμέτρων για τα επιμέρους προγράμματα που περιλαμβάνει.

Το runtime μπορεί να εκτελεί ταυτόχρονα τόσο ροές εργασίας σε κατάσταση αλληλεπιδραστικής εκτέλεσης, όσο και ροές εργασίας σε κατάσταση προγραμματισμένης εκτέλεσης.

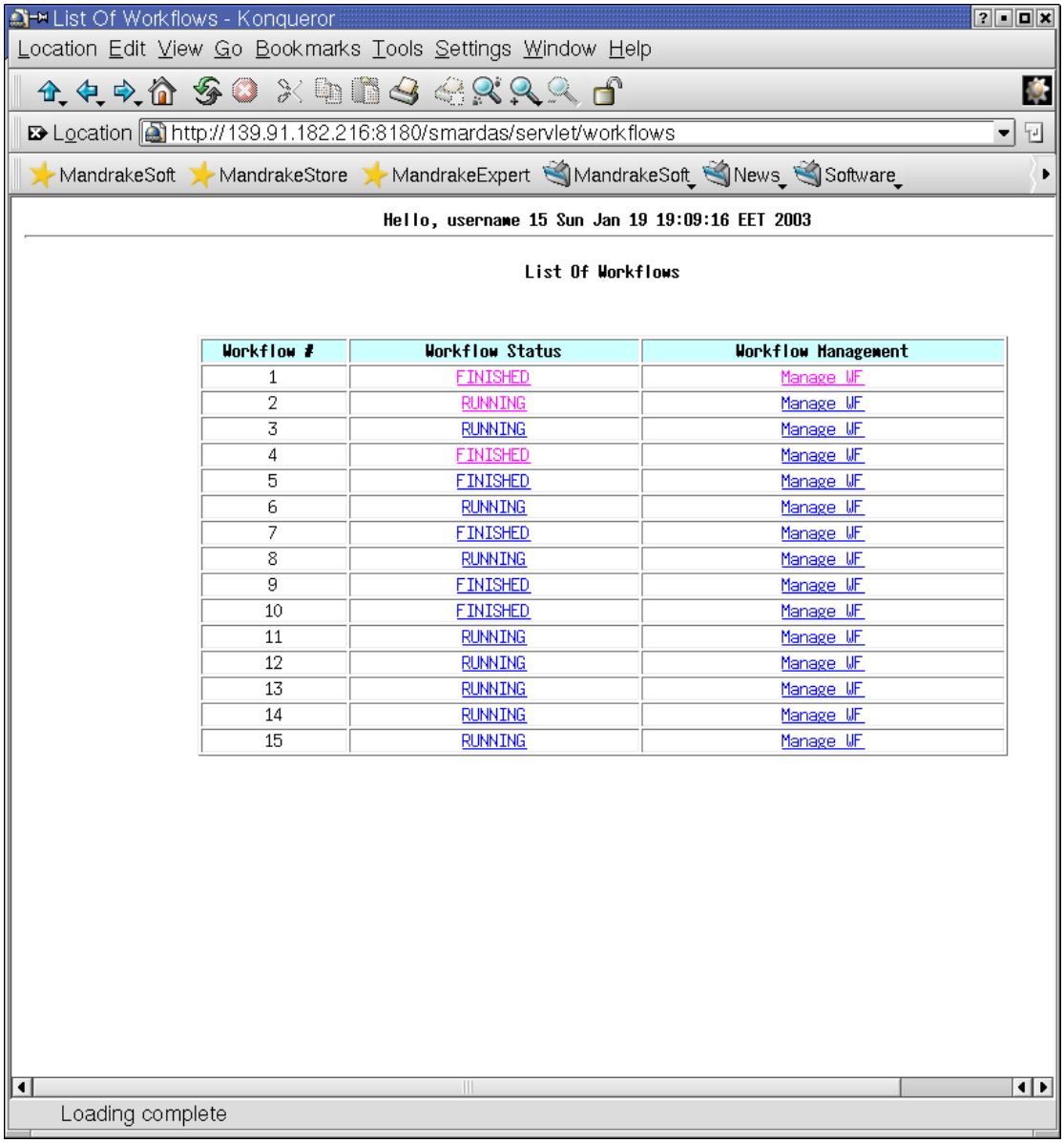
### 6.3.2 Αλληλεπιδραστική εκτέλεση ροών εργασίας

Η αλληλεπιδραστική κατάσταση λειτουργίας επιτρέπει στο χρήστη να επεμβαίνει κατά τη διάρκεια εκτέλεσης μίας ροής εργασίας. Οι δυνατότητες παρέμβασης αφορούν την παροχή παραμέτρων που χρειάζονται για την εκτέλεση ενός προγράμματος και την επιλογή της πορείας της ροής εκτέλεσης όταν αυτό φτάσει σε προκαθορισμένα σημεία. Ο χρήστης αλληλεπιδρά με το runtime σύστημα μέσω ενός Web interface το οποίο έχει αναπτυχθεί και παρουσιάζεται παρακάτω.

#### 6.3.2.1 Web interface

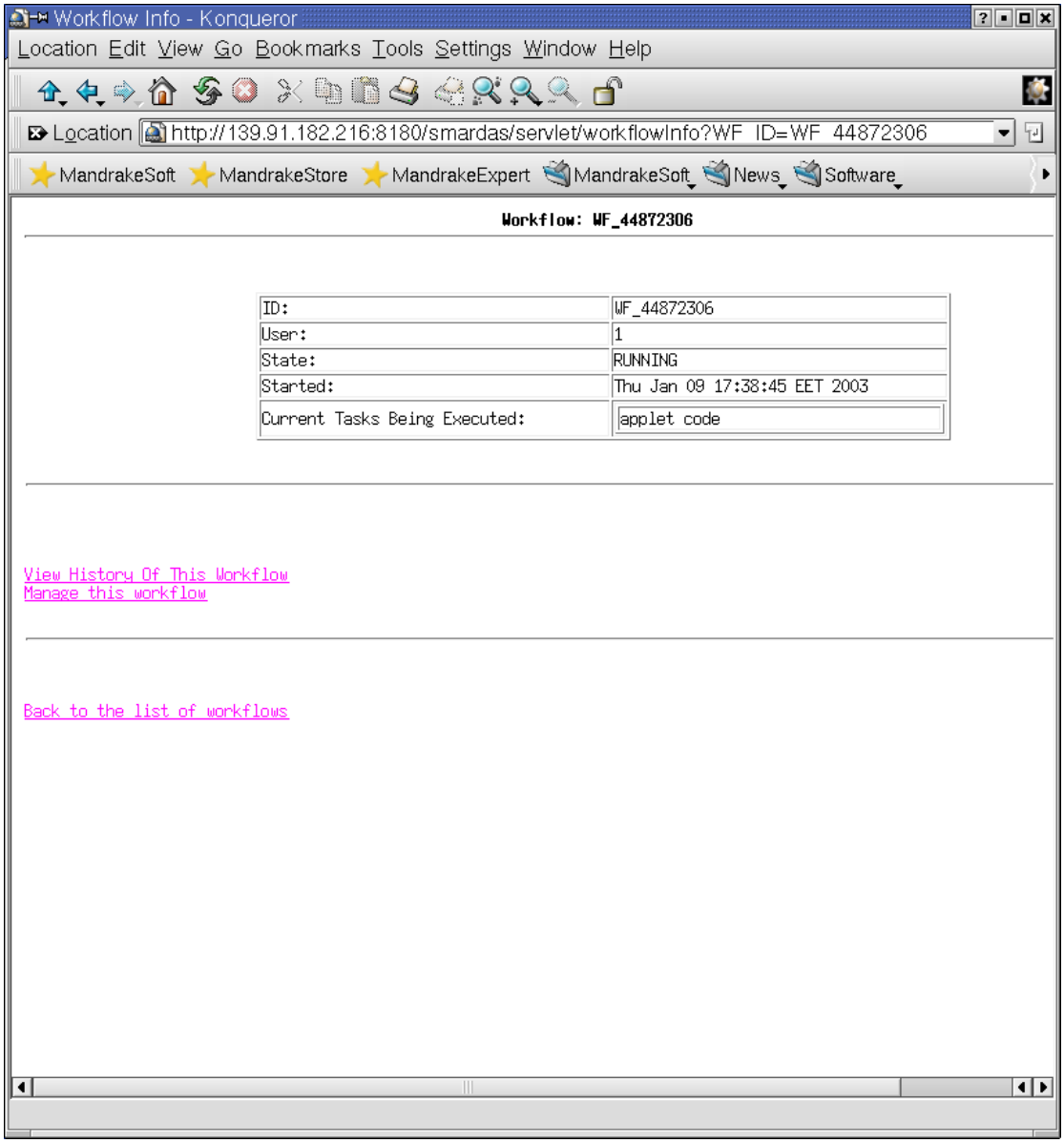
Το web interface επιτρέπει στο χρήστη τη διαχείριση των ροών εργασίας τις οποίες έχει εκκινήσει. Με τη χρήση ενός web browser ο χρήστης μπορεί να διαχειριστεί τις ροές εργασίας μέσω ενός συνόλου επιτρεπόμενων λειτουργιών. Οι λειτουργίες αυτές χωρίζονται σε δύο κύριες κατηγορίες. Αυτές είναι οι:

- λειτουργίες παρακολούθησης της πορείας εκτέλεσης κι οι
- λειτουργίες αλληλεπίδρασης

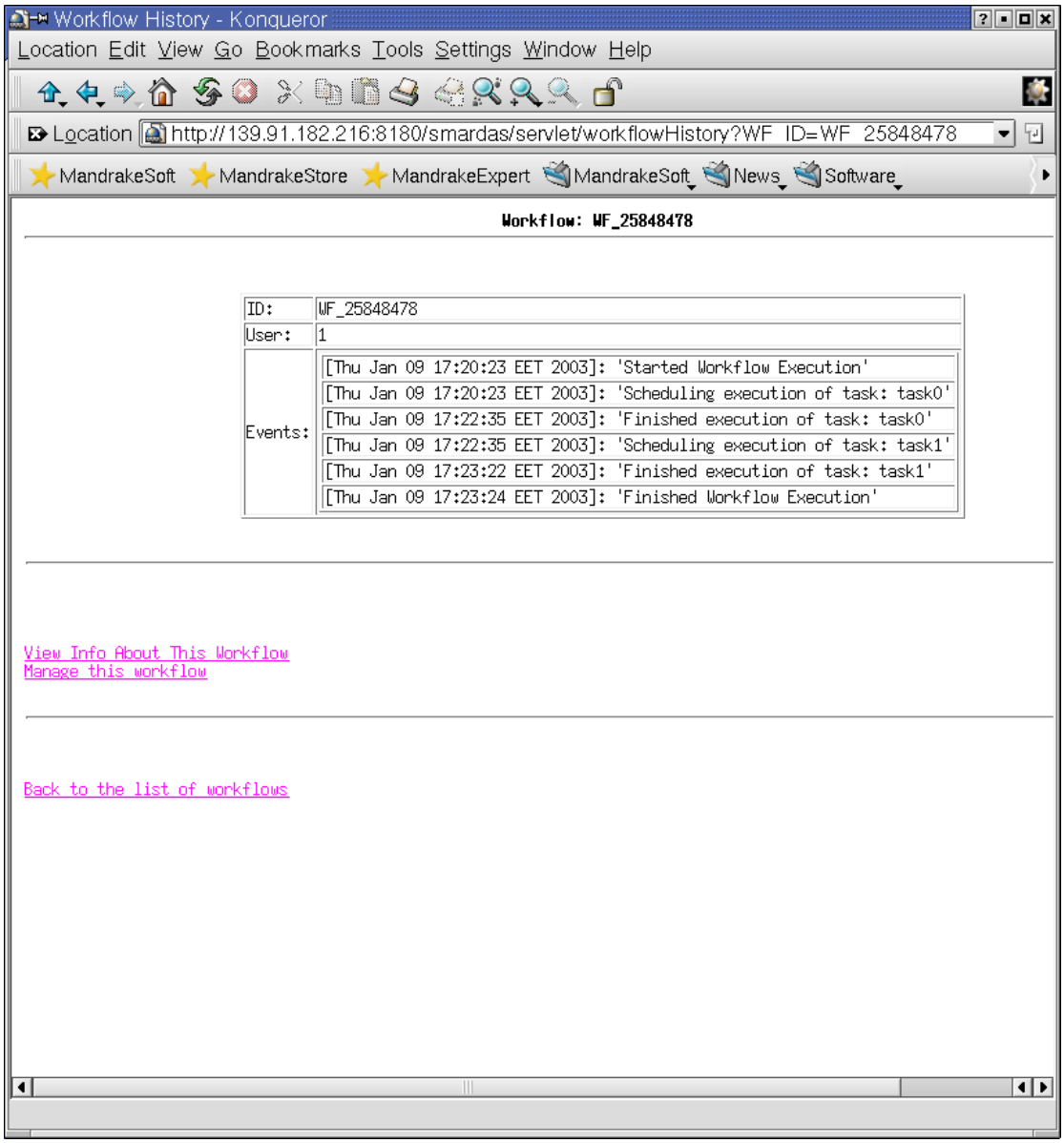



**Εικόνα 6.1.** Λίστα ροών εργασίας

Οι λειτουργίες παρακολούθησης της πορείας εκτέλεσης επιτρέπουν στο χρήστη να μαθαίνει την τρέχουσα κατάσταση μίας ροής εργασίας (πρόγραμμα ή προγράμματα που εκτελούνται την τρέχουσα στιγμή) καθώς και το ιστορικό εκτέλεσης. Το ιστορικό της εκτέλεσης αφορά τα βήματα που εκτελέστηκαν σε προηγούμενα στάδια τα οποία παρουσιάζονται σε χρονολογική σειρά.




**Εικόνα 6.2.** Τρέχουσα κατάσταση ροής εργασίας




**Εικόνα 6.3.** Ιστορικό εκτέλεσης ροής εργασίας

Εκτός από τις λειτουργίες παρακολούθησης παρέχονται δυνατότητες αλληλεπίδρασης στο χρήστη. Μέσω αυτών μπορεί να δώσει κάποιες παραμέτρους που απαιτούνται από ένα πρόγραμμα για την εκτέλεσή του ή να αποφασίσει για την πορεία την οποία θα ακολουθήσει μία ροή εργασίας.

#### 6.3.2.1.1 Παράμετροι προγραμμάτων

Μία εργασία που έχει δρομολογηθεί μπορεί να απαιτεί/επιτρέπει τη χρήση παραμέτρων. Οι παράμετροι αυτές αφορούν τις παραμέτρους που τυχόν δίνονται στη γραμμή εντολών κατά την κλήση του. Ως παράδειγμα μπορούμε να θεωρήσουμε την εντολή cp του UNIX η οποία αντιγράφει ένα αρχείο. Αυτή όταν καλείται δέχεται ως όρισμα δύο παραμέτρους που αφορούν ονόματα αρχείων:

%> cp work.dat backup

Μία εντολή (πρόγραμμα) μπορεί να δέχεται και παραμέτρους/ορίσματα που δεν είναι υποχρεωτικά, αλλά λειτουργούν ως flags ενεργοποιώντας κάποιες ειδικές λειτουργίες. Π.χ. στην εντολή ls η παράμετρος –l μπορεί να θεωρηθεί μη υποχρεωτική.

%>ls –l /home/user1/

Η περιγραφή μιας εργασίας στη ροή εργασίας ορίζει σαφώς ποιες παραμέτρους μπορεί να δεχτεί ένα πρόγραμμα καθώς και τις ιδιότητες των παραμέτρων αυτών. Συγκεκριμένα, για τις παραμέτρους ορίζονται οι εξής ιδιότητες:

- **Υποχρεωτικές ή μη:** Δηλώνουν αν η χρήση τους είναι υποχρεωτική. Το στοιχείο αυτό παρουσιάζεται στο χρήστη ώστε να γνωρίζει ότι πρέπει να δωθεί τιμή για την παράμετρο αυτή ή όχι.
- **Τύπος:** Ο τύπος αφορά τις επιτρεπτές τιμές που μπορεί να πάρει η παράμετρος. Συγκεκριμένα, οι επιτρεπόμενοι τύποι είναι οι εξής:
    - Ακέραιος: η παράμετρος πρέπει να λάβει ακέραια τιμή
    - Πραγματικός: η παράμετρος λαμβάνει κάποια πραγματική τιμή
    - Λογικός: η παράμετρος λαμβάνει μία τιμή αληθείας (αληθή ή ψευδή)
    - Αλφαριθμητικός: η παράμετρος λαμβάνει ως τιμή ένα αλφαριθμητικό, π.χ. μία λέξη ή μία πρόταση.
    - Ειδικός – αναφορά σε dataset. Μία παράμετρος ειδικού τύπου αφορά το όνομα ενός data set που θα χειριστεί ένα πρόγραμμα. Χρησιμοποιείται κυρίως

για τον ορισμό αρχείων εξόδου. Η ροή εργασίας μπορεί να δίνει κάποιο όνομα για ένα τέτοιο data set όμως ταυτόχρονες εκτελέσεις δύο instances μίας ροής εργασίας θα είχαν ως αποτέλεσμα την αδυναμία διαχωρισμού των αρχείων εξόδου. Μέσω των παραμέτρων ειδικού τύπου υπάρχει η δυνατότητα αλλαγής της περιγραφής της ροής εργασίας σε σχέση με τα ονόματα κάποιων παραγόμενων αρχείων εξόδου έτσι ώστε να υπάρχει η δυνατότητα ταυτόχρονων εκτελέσεων ενός προγράμματος εφόσον το πρόγραμμα το επιτρέπει.

- **Default τιμή:** Η τιμή που θα δωθεί αν ο χρήστης δεν επιλέξει κάποια διαφορετική τιμή.
- **Σύνταξη:** ο τρόπος σύνταξης της παραμέτρου έτσι ώστε να γίνει αποδεκτή από το σχετικό πρόγραμμα.
- **Όνομα:** ένα όνομα που χαρακτηρίζει την παράμετρο αυτή
- **Περιγραφή:** σύντομη περιγραφή του ρόλου της παραμέτρου
- **Αναφορά σε κάποιο data set:** αφορά παραμέτρους ειδικού τύπου. Η τιμή της ιδιότητας αναφέρεται σε κάποιο data set ID που περιγράφεται στη ροή εργασίας.

Ο χρήστης είναι σε θέση μέσω του web interface να επιλέξει ποιες από τις δυνατές παραμέτρους θέλει να χρησιμοποιήσει καθώς και τις τιμές που επιθυμεί να δώσει σε αυτές.

#### 6.3.2.1.2 Backtracking

Η έννοια του backtracking παρουσιάστηκε στην παράγραφο §6.2.2.2.2. Η λειτουργικότητα που παρέχει επιτρέπει τη δυνατότητα επιστροφής της πορείας εκτέλεσης σε κάποιο προηγούμενο στάδιο.

Μέσω του web interface ο χρήστης μπορεί όταν συναντηθεί ένα checkpoint κατά την εκτέλεση μίας ροής εργασίας να επιλέξει προς τα πού επιθυμεί να κινηθεί η πορεία εκτέλεσης. Συγκεκριμένα, παρουσιάζονται στο χρήστη οι δυνατές επιλογές σχετικά με τη συνέχιση της πορείας εκτέλεσης της ροής εργασίας και της επιστροφής σε κάποιο προηγούμενο βήμα.

### 6.3.3 Προγραμματισμένη εκτέλεση ροών εργασιών

Όπως αναφέρθηκε προηγουμένως, το runtime σύστημα παρέχει τη δυνατότητα προγραμματισμένης εκτέλεσης ροών εργασίας. Σε αυτήν την κατάσταση εκτέλεσης οι παράμετροι που απαιτούνται από τις ροές εργασίας παρέχονται εξ'αρχής και ως εκ τούτου απενεργοποιείται η δυνατότητα αλληλεπίδρασης από το χρήστη (παροχή παραμέτρων και backtracking). Ουσιαστικά, η χειροκίνητη διαδικασία εκτέλεσης κύκλων σε μια ροή εργασιών μέσω της λειτουργικότητας που παρέχει το backtracking πλέον αυτοματοποιείται και ο χρήστης μπορεί πλέον να επιτύχει αντίστοιχο αποτέλεσμα με αυτοματοποιημένο τρόπο ορίζοντας την παράλληλη εκτέλεση μιας ροής εργασίας σε πολλά διαφορετικά instances όπου για το καθένα από αυτά παρέχονται εξ'αρχής διαφορετικές παράμετροι. Ο αριθμός των instances ροών εργασίων που θα δημιουργηθούν εξαρτάται από το συνδυασμό των παραμέτρων που θα δωθούν από το χρήστη σχετικά με την εκτέλεση των ροών εργασίας.

Ένα παράδειγμα παραμέτρων προγραμματισμένης εκτέλεσης δίνεται στην παράγραφο §7.2.

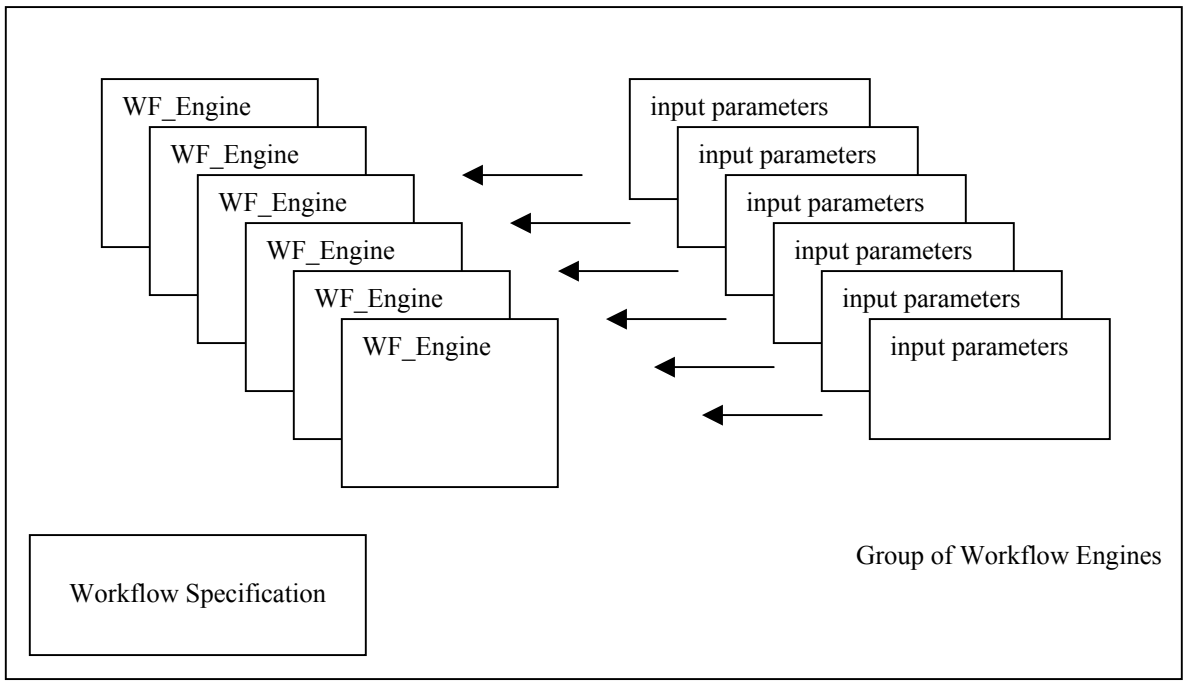
Κάθε μία μηχανής εκτέλεσης, που αντιστοιχεί σε ένα instance, όταν αρχικοποιείται λαμβάνει τις παραμέτρους που θα χρειαστούν κατά την πορεία εκτέλεσης του της ροής εργασίας έτσι ώστε να αυτοματοποιηθεί η διαδικασία εκτέλεσής του. Στη συνέχεια αυτές οι ροές εργασίας θα εκτελεστούν παράλληλα χωρίς καμία ανάγκη εισαγωγής παραμέτρων από το χρήστη και με τον τερματισμό της εκτέλεσής τους μπορούν να ελεγχθούν τα αποτελέσματα που παράχθηκαν από το καθένα.

#### 6.3.3.1 Group μηχανών εκτέλεσης

Η προγραμματισμένη εκτέλεση ροών εργασίας προκαλεί τη δημιουργία ορισμένων μηχανών εκτέλεσης που αναλαμβάνουν την εκτέλεση ενός instance η καθεμία. Όλες αυτές οι μηχανές αποτελούν μια ομάδα μηχανών εκτέλεσης η οποία είναι και η οντότητα που είναι ορατή στο χρήστη μέσω του runtime συστήματος.

Οι χειρισμοί, συνεπώς, γίνονται πάνω στο group των μηχανών και όχι σε κάθε μία μηχανή ξεχωριστά. Ο χρήστης ειδοποιείται ότι τερμάτισε η εκτέλεση των ροών εργασιών

που δημιούργησε μόνο όταν όλες οι ροές εργασιών έχουν τερματίσει την εκτέλεσή τους. Βέβαια, το runtime σύστημα καταγράφει τις λειτουργίες που εκτελεί κάθε μηχανή ξεχωριστά στα logs του συστήματος.




**Σχήμα 6.11.** Group μηχανών εκτέλεσης ροών εργασίας

### 6.3.4 Notification System

Το runtime σύστημα περιλαμβάνει ένα μηχανισμό ειδοποίησης προκειμένου να εξυπηρετηθεί η ανάγκη ενημέρωσης των χρηστών για σημαντικά γεγονότα που αφορούν την εκτέλεση μιας ροής εργασίας. Ένα μήνυμα αποστέλλεται στο χρήστη από μία μηχανή εκτέλεσης ροών εργασίας μέσω του μηχανισμού ειδοποίησης που έχει υλοποιηθεί. Ο μηχανισμός αυτός μπορεί να ενσωματώσει αρκετούς τρόπους ενημέρωσης των χρηστών, όπως e-mail, SMS κ.α. Αφού ο μηχανισμός ειδοποίησης παραλάβει το μήνυμα προς αποστολή τότε επιλέγει έναν από τους δυνατούς τρόπους αποστολής του μηνύματος. Προς το παρόν έχει υλοποιηθεί μόνο ο η λειτουργία αποστολής e-mail, αλλά άμεση μπορεί να είναι και η προσθήκη νέων λειτουργιών αποστολής μηνυμάτων, όπως η υπηρεσία SMS. Τα στοιχεία του παραλήπτη των μηνυμάτων ειδοποίησης περιλαμβάνται μέσα στην περιγραφή της ροής εργασίας και προς το παρόν περιλαμβάνουν μια e-mail διεύθυνση και έναν αριθμό συνδρομητή κινητής τηλεφωνίας.

Ο χρήστης ειδοποιείται από το runtime σύστημα είτε με την πραγματοποίηση σημαντικών γεγονότων που αφορούν την εκτέλεση μίας ροής εργασίας του χρήστη είτε όταν απαιτείται να δωθεί κάποια είσοδος από αυτόν. Συγκεκριμένα, ειδοποίηση αποστέλλεται από το notification system στις ακόλουθες περιπτώσεις:

- Τερματισμός εκτέλεσης ροής εργασίας,
- Ζήτηση παραμέτρων από έναν εκτελεστή εργασίας για την εκτέλεση ενός προγράμματος,
- Τερματισμός εκτέλεσης εργασίας που αποτελεί checkpoint (πιθανό backtracking).

## 6.4 Χαρακτηρισικά runtime συστήματος

### 6.4.1 Εισαγωγή

Τα περισσότερα από τα σημερινά συστήματα ροών εργασίας προσφέρουν σημαντικές δυνατότητες για τη διαχείριση πολύπλοκων διαδικασιών, αλλά γενικά δεν προσφέρουν σημαντική υποστήριξη για την **αυτοματοποιημένη εκτέλεση** ροών εργασίας, την **παρακολούθηση** της κατάστασης των λειτουργιών που εκτελούνται, το **υψηλό ποσοστό διαθεσιμότητας** του συστήματος και κυρίως τη διατήρηση της συνέπειας και του concurrency control και την **ανάνηψη** έπειτα από κάποια αποτυχία. Βέβαια, τα χαρακτηριστικά αυτά είναι απαραίτητα προκειμένου να επιτευχθεί ο έλεγχος και η διαχείριση ταυτόχρονων εκτελέσεων ανεξάρτητων λειτουργιών στο σύστημα.

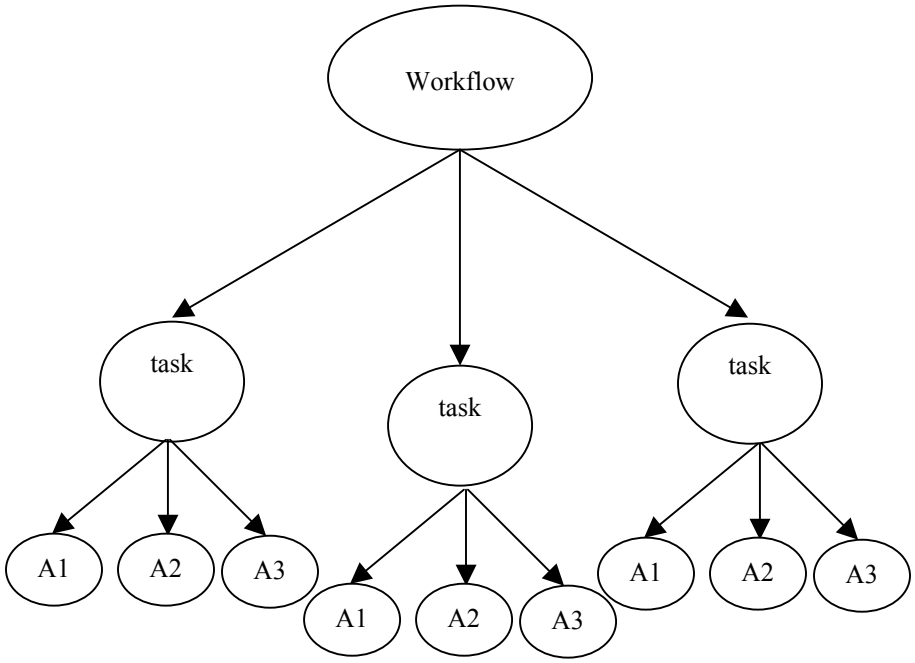
Το runtime σύστημα του ARION καλύπτει τις ανάγκες αυτές υλοποιώντας την απαιτούμενη λειτουργικότητα προσδίδοντας στο συνολικό σύστημα τα επιθυμητά χαρακτηριστικά. Η λειτουργικότητα της αυτοματοποιημένης εκτέλεσης ροών εργασίας περιγράφηκε παραπάνω (βλ. §6.3.3) καθώς επίσης και η δυνατότητα παρακολούθησης της πορείας εκτέλεσής τους (βλ. §6.3.2.1). Στις επόμενες παραγράφους εξετάζονται τα χαρακτηριστικά της συνέπειας και της ανάνηψης του συστήματος έπειτα από κάποια αποτυχία, η δυνατότητα χρήσης ενός εφεδρικού server για υψηλό ποσοστό διαθεσιμότητας του συστήματος και περιγράφεται το παρεχόμενο επίπεδο υποστήριξής τους.

### 6.4.2 Δραστηριότητες Μεγάλης Διάρκειας

#### 6.4.2.1 Εισαγωγή

Η εκτέλεση μίας ροής εργασίας συνήθως αποτελεί μία ιδιαιτέρως χρονοβόρα διαδικασία (long running activity). Ο συνδυασμός πληθώρας προγραμμάτων που εκτελούνται βάσει ορισμένων κανόνων καθώς και η ανάγκη αλληλεπίδρασης με το χρήστη μπορεί να οδηγήσει σε εκτελέσεις που διαρκούν συνολικά πολλές ώρες ή ακόμα και ημέρες. Επιπλέον, η δυνατότητα αυτοματοποιημένης εκτέλεσης ροών εργασίας που παρέχεται από το runtime σύστημα του ARION επιτρέπει τον ορισμό ενός σημαντικού αριθμού instances μίας ροής εργασίας που στη συνέχεια εκτελούνται ομαδικά. Μία τέτοια διαδικασία ομαδικής εκτέλεσης είναι επίσης δυνατόν να διαρκέσει μεγάλο χρονικό διάστημα. Είναι προφανής λοιπόν η ανάγκη παροχής εγγυήσεων από το runtime σύστημα που θα διασφαλίζουν την αποφυγή επανεκτέλεσης χρονοβόρων διαδικασιών σε περιπτώσεις αποτυχίας κατά τη διάρκεια εκτέλεσης μίας ροής εργασίας.

Αν συμβολίσουμε ολόκληρη τη ροή εργασίας ως τον κεντρικό κόμβο μιας δενδρικής δομής τότε μπορούμε να αναλύσουμε τη ροή εργασίας σε επιμέρους τμήματα καθέ ένα από τα οποία θα αποτελεί μία εργασία της ροής αυτής. Μία τέτοια εργασία ανατίθεται σε έναν εκτελεστή εργασίας. Αντίστοιχα, μία εργασία μπορεί να αναλυθεί σε επιμέρους ενέργεις τις οποίες πρέπει να εκτελέσει ο εκτελεστής εργασίας.




**Σχήμα 6.12.** Δενδρικό σχήμα ροής εργασίας

Προκειμένου να αποφύγουμε την ανάγκη επανεκτέλεσης ολόκληρης της ροής εργασίας σε περίπτωση αποτυχίας, μπορούμε να θεωρήσουμε τη ροή εργασίας ως ξεχωριστές ομάδες ενεργειών που αρκεί να έχουν εκτελεστεί αυτές, διατηρώντας βέβαια τους κανόνες που θέτει η ροή εργασίας, έτσι ώστε να έχει εκτελεστεί ολόκληρη η ροή. Η παραδοχή αυτή επιτρέπει να μην υπάρχει η ανάγκη επανεκτέλεσης ολόκληρης της ροής εργασίας σε περίπτωση αποτυχίας, εφόσον π.χ. έχει ήδη εκτελεστεί ένα υπόδενδρο της παραπάνω αναπαράστασης και να ασχοληθούμε μόνο με τα υπόλοιπα.

Η παραπάνω λογική γενκικεύεται και εφαρμόζεται πολύ εύκολα στο σύστημά μας χάρη στη χρήση των ουρών ενεργειών από τους εκτελεστές εργασίας. Όπως αναφέρθηκε στην παράγραφο §6.2.2.4 οι εκτελεστές εργασίας χρησιμοποιούν ουρές ενεργειών οι οποίες περιλαμβάνουν όλες τις επιμέρους ενέργειες που πρέπει να εκτελεστούν. Αυτές μπορεί να συμπεριλαμβάνουν ενέργειες όπως μεταφορά αρχείων, αποστολή πράκτορα σε απομακρυσμένο μηχάνημα για την εκτέλεση κάποιου προγράμματος κ.α. Με την περάτωση μίας ενέργειας ο εκτελεστής εργασίας φροντίζει να αφαιρέσει την ενέργεια που μόλις διεκπεραίωσε από την ουρά και να σώσει ένα στιγμιότυπο της ουράς ενεργειών σε μόνιμο αποθηκευτικό μέσο. Η ουρά που σώζεται είναι σε μία έγκυρη ή

συνεπή κατάσταση, δηλ. αποτελείται από τις ενέργειες που απομένουν να περατωθούν χωρίς να σώζονται ενδιάμεσες μη συνεπείς καταστάσεις μισοτελειωμένων ενεργειών.

Το σώσιμο της κατάστασης της ουράς επιτρέπει τη διατήρηση της πληροφορίας σχετικά με το ποιες ενέργειες έχουν ήδη εκτελεστεί και ποιες απομένουν ακόμα προς εκτέλεση. Αυτή η πληροφορία μπορεί να χρησιμοποιηθεί στην περίπτωση που έχει προκύψει κάποια αποτυχία στο σύστημα. Σε αυτήν την περίπτωση, όταν ο server εκκινηθεί ξανά μπορεί να υπάρξει η επιλογή η εκτέλεση να συνεχιστεί για τις εναπομείναντες εργασίες χωρίς να απαιτείται η επανεκτέλεση των ήδη εκτελεσμένων ενεργειών.

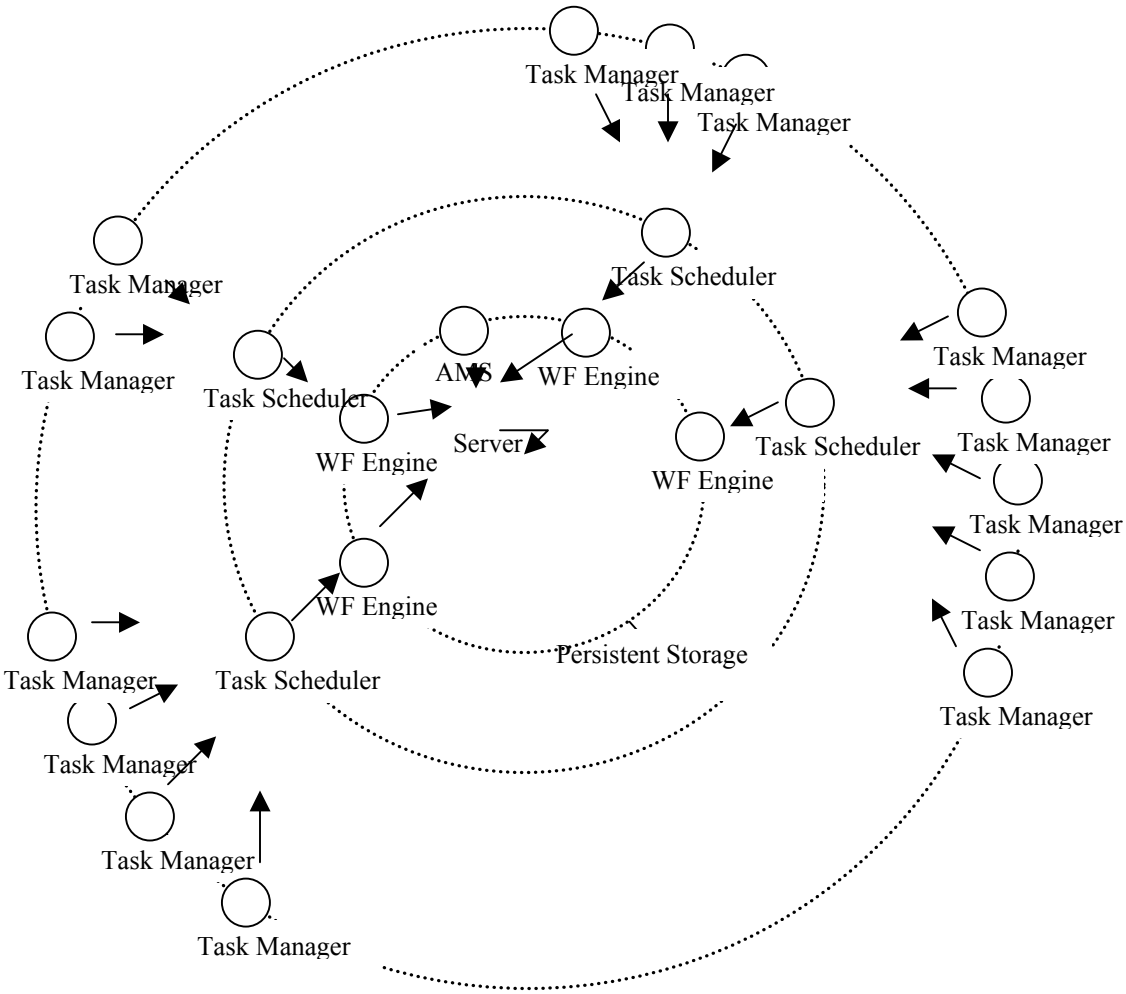
Αντίστοιχα κι άλλα μέρη του runtime συστήματος μπορούν να σώζουν συνεπείς καταστάσεις τους προκειμένου να υπάρχει η δυνατότητα ανάνηψης σε περίπτωση αποτυχίας. Τέτοια μέρη είναι π.χ. ο κεντρικός server, οι μηχανές εκτέλεσης κ.α.

#### 6.4.2.2 Δακτύλιοι

Έχει επιλεγεί ένα σχήμα δακτυλίων για την οργάνωση των επιμέρους κομματιών του runtime συστήματος για την υποστήριξη της λειτουργίας του σωσίματος των συνεπών καταστάσεων. Το σχήμα αυτό εξηγείται παρακάτω.

##### 6.4.2.2.1 Σχεδίαση

Το σώσιμο των καταστάσεων γίνεται σε μόνιμο αποθηκευτικό μέσο που επιτρέπει το διάβασμα της τελευταίας συνεπούς κατάστασης που έχει σωθεί. Αυτό καθιστά εφικτή, σε περίπτωση αποτυχίας, τη συνέχιση της εκτέλεσης μιας ροής εργασίας από την τελευταία συνεπή κατάσταση που είχε σωθεί προτού υπάρξει η αποτυχία. Τα επιμέρους υποσυστήματα που σώζουν συνεπείς καταστάσεις κατά τη διάρκεια της ζωής του runtime συστήματος έχουν οργανωθεί σε 4 δακτυλίους οι οποίοι και αντικατοπτρίζουν τη βασική αρχιτεκτονική του συστήματος.




**Σχήμα 6.13.** Η αρχιτεκτονική των δακτυλίων

Στον εξωτερικό δακτύλιο βρίσκονται οι εκτελεστές εργασίας οι οποίοι είναι ενεργοί και επιθυμούν να σώσουν τις καταστάσεις τους. Στο δεύτερο δακτύλιο βρίσκονται οι δρομολογητές εργασιών, στον τρίτο δακτύλιο οι μηχανές εκτέλεσης και το AMS, ενώ στον εσωτερικό δακτύλιο βρίσκεται ο server. Στο κέντρο όλων των δακτυλίων βρίσκεται το Persistent Storage module το οποίο και υλοποιεί τη λειτουργία της μόνιμης αποθήκευσης των στιγμιοτύπων. Η επικοινωνία για τη λειτουργία του σωσίματος συνεπών καταστάσεων είναι επιτρεπτή μονάχα μεταξύ γειτονικών δακτυλίων και μεταξύ οντοτήτων που σχετίζονται μεταξύ τους, π.χ. ένας εκτελεστής εργασίας μπορεί να απευθυνθεί μονάχα στον δρομολογητή εργασιών ο οποίος του ανέθεσε την εργασία του και σε καμία άλλη οντότητα.

Αυτό συνεπάγεται πως μία αίτηση για σώσιμο κατάστασης από μία οντότητα θα πρέπει να περάσει διαδοχικά από όλους τους ενδιάμεσους δακτυλίους οι οποίοι θα πρέπει να προωθήσουν την αίτηση στον επόμενο δακτύλιο ώσπου να φτάσει στον εσωτερικό ο οποίος είναι και ο μόνος που έχει άμεση πρόσβαση στο υποσύστημα σωσίματος συνεπών καταστάσεων.

Μια τέτοια σχεδίαση καθιστά απλή την εφαρμογή κανόνων ή περιορισμών στη χρήση της δυνατότητας αυτής από τις οντότητες έχοντας επίπεδα διαβαθμίσεων εξουσίας, αφού κάθε επίπεδο μπορεί να απογορέψει ή να επιτρέψει στο εξωτερικό του επίπεδο τη χρήση της δυνατότητας αυτής. Το μπλοκάρισμα μπορεί να γίνει απλώς αγνοώντας μία αίτηση για σώσιμο μίας συνεπούς κατάστασης. Π.χ. ένας δρομολογητής εργασιών θα μπορούσε να αρνηθεί το σώσιμο της κατάσταση ενός εκτελεστή εργασίας στον οποίο έχει αναθέσει κάποια εργασία.

Επίσης, η σχεδίαση αυτή επιτρέπει στις οντότητες να γνωρίζουν πότε οι οντότητες με τις οποίες σχετίζονται σώζουν τις συνεπείς τους καταστάσεις. Αυτή η πληροφορία μπορεί να χρησιμοποιηθεί από μία τέτοια οντότητα για δικούς της σκοπούς.

#### 6.4.2.2.2 Οντότητες που σώζουν την κατάστασή τους

Οι οντότητες του runtime συστήματος που σώζουν συνεπή στιγμιότυπα των καταστάσεών τους είναι οι εξής:

- **Εκτελεστής εργασίας:** Ένας εκτελεστής εργασίας τραβάει διαδοχικά μία μία ενέργεια από την ουρά του και με την επιτυχή ολοκλήρωσή της φροντίζει να σώσει την τρέχουσα κατάσταση του. Τυχόν αποτυχία του server θα είχε ως αποτέλεσμα ένας εκτελεστής εργασίας να συνεχίσει την εργασία του, όταν ο server αναρρώσει, από την πρώτη ανολοκλήρωτη εργασία του και συνεπώς αποφεύγει την εκτέλεση εργασιών που ήδη έχει εκτελέσει (π.χ. λήψη παραμέτρων από το χρήστη).

  Στην περίπτωση εκτελεστών εργασίας που έχουν αναλάβει την εκτέλεση «ευαίσθητων» προγραμμάτων, δηλ. προγραμμάτων που πρέπει να εκτελούνται σε απομόνωση σώσιμο κατάστασης γίνεται μονάχα για τη λήψη παραμέτρων καθώς

όλες οι άλλες επιμέρους εργασίες πρέπει να γίνονται ατομικά χωρίς την οποιαδήποτε τυχόν παρέμβαση από άλλον εκτελεστή εργασίας, στοιχείο το οποίο δεν μπορεί να διασφαλιστεί στην περίπτωση όπου για ορισμένο χρονικό διάστημα είχε διακοπεί η λειτουργία του εκτελεστή εργασίας που είχε τον έλεγχο εκτέλεσης του προγράμματος (περισσότερα στην παράγραφο §6.4.2.4).

- **Δρομολογητής εργασιών:** Σώζει την κατάστασή του όταν αναθέσει την εκτέλεση μίας εργασίας καθώς κι όταν μία εργασία τερματιστεί.

- **Μηχανή εκτέλεσης:** Μία μηχανή εκτέλεσης σώζει την κατάστασή της όταν αρχικοποιηθεί με την περιγραφή της ροής εργασίας και όταν αυτή τερματίζει την εκτέλεσή της

- **Group μηχανών εκτέλεσης:** Ένα group μηχανών εκτέλεσης σώζει την κατάστασή του όταν αρχικοποιηθεί.

- **AMS:** Σώζει την κατάστασή του όταν αρχικοποιείται

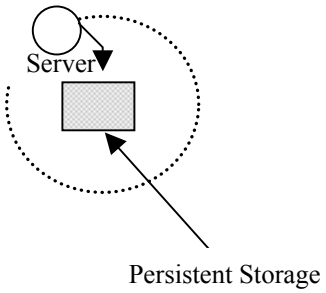
- **Server:** Ο server σώζει την κατάστασή του όταν αρχικοποιηθεί και κάθε φορά που δημιουργείται μία μηχανή εκτέλεσης ή ένα group μηχανών εκτέλεσης.

Οι οντότητες αυτές υλοποιούν το interface gr.ics.forth.dsl.arion.utils.PersistentObject μέσω του οποίου μπορούν να χειριστούν με κοινό τρόπο από το σύστημα.

#### 6.4.2.2.3 Ανάνηψη έπειτα από αποτυχία

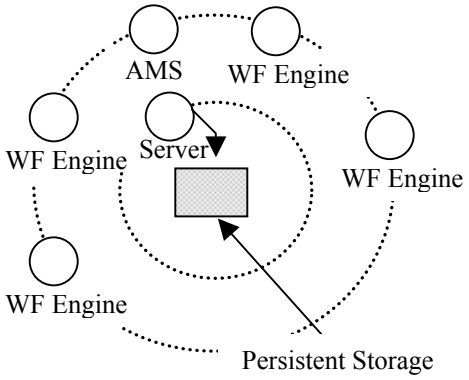
Το σώσιμο των συνεπών καταστάσεων από τις οντότητες του συστήματος επιτρέπει την ανάνηψη του συστήματος έπειτα από αποτυχία. Κατά τη διάρκεια της ανάνηψης το σύστημα επανέρχεται σε λειτουργία και οι οντότητες που υπήρχαν σε αυτό επαναφέρονται με τρέχουσα κατάσταση το τελευταίο στιγμιότυπο που έχει σωθεί για κάθε μία από αυτές. Ένας server μπορεί να εκκινηθεί σε κατάσταση ανάνηψης με τη χρήση κατάλληλου αρχείου αρχικοποίησης.

Κατά τη διάρκεια της ανάνηψης ακολουθείται μια διαδικασία αναδημιουργίας των δακτυλίων ξεκινώντας από τον εσωτερικό δακτύλιο (server) με κάθε δακτύλιο να αναλαμβάνει τη δημιουργία του αμέσως επόμενου δακτυλίου. Έτσι, σε πρώτη φάση όταν ο server εκκινηθεί σε κατάσταση ανάνηψης τότε ο server αρχικοποιείται με το τελευταίο σωσμένο στιγμιότυπο του server που απέτυχε. Η κατάσταση λαμβάνεται από το PersistentStorage module που λειτουργεί ως μόνιμη αποθήκη στιγμιοτύπων.




**Σχήμα 6.14a.** Ανά[illegible]αποτυχία – Α' φάση

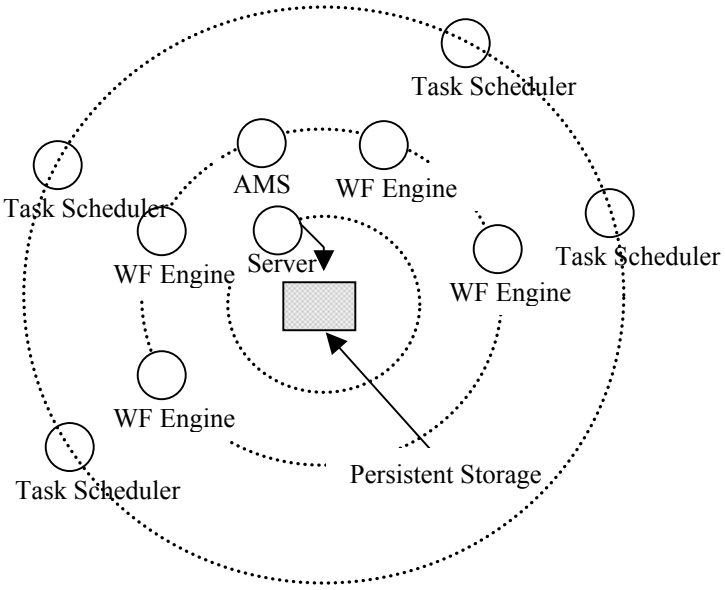
Αφού γίνει η αρχικοποίηση του, ο server αναλαμβάνει να δημιουργήσει το δεύτερο δακτύλιο. Ο δεύτερος δακτύλιος περιέχει το AMS και τις μηχανές εκτέλεσης που υπήρχαν κατά την αποτυχία του συστήματος. Ο server δημιουργεί τις οντότητες αυτές και στη συνέχεια περνάει τον έλεγχο στο δεύτερο δακτύλιο.




**Σχήμα 6.14b.** Ανάνηψη έπειτα από αποτυχία – Β' φάση

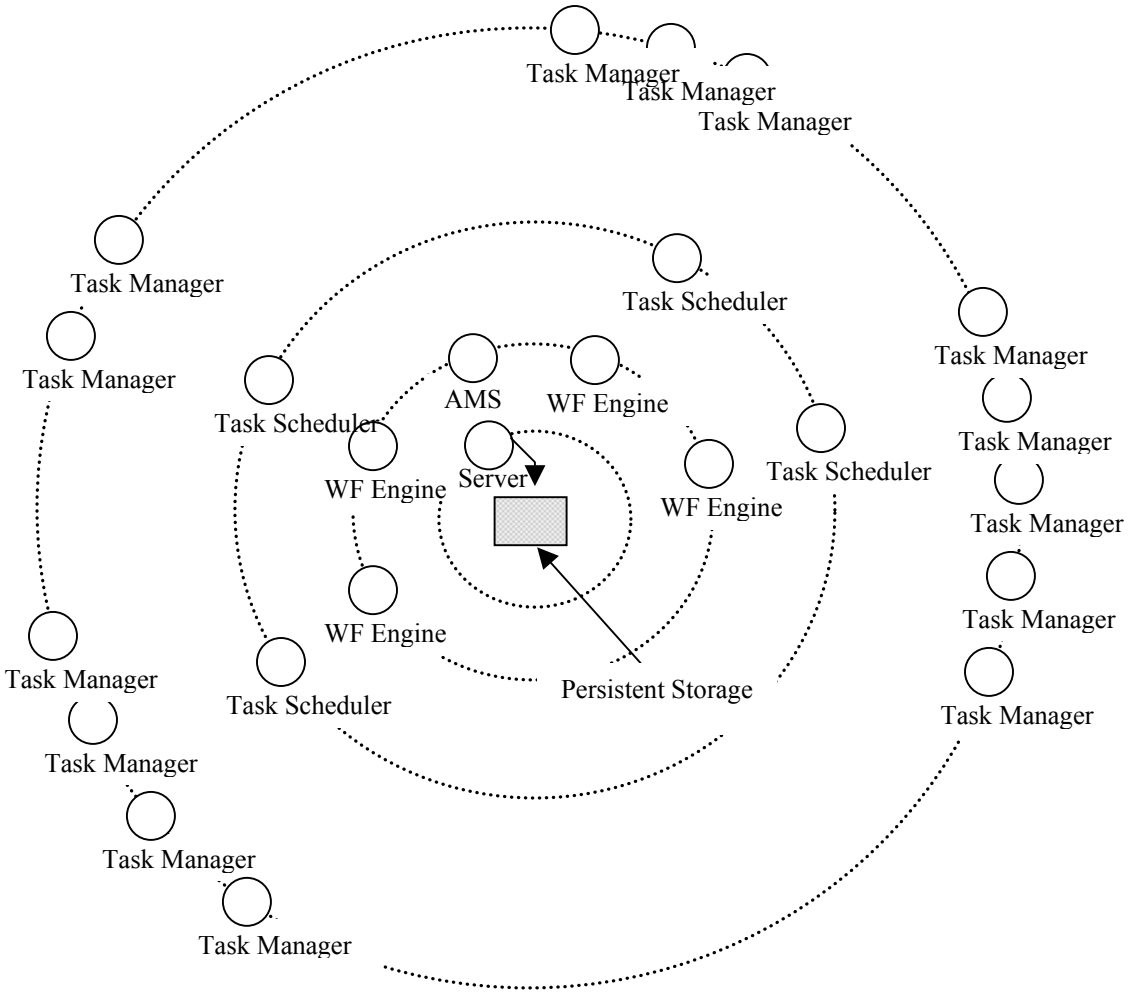
Στη φάση αυτή οι οντότητες του 2$^{ου}$ δακτυλίου (μηχανές εκτέλεσης, AMS) διαβάζουν από το PersistentStorage module το τελευταίο σωσμένο στιγμιότυπό τους και

αρχικοποιούν την κατάστασή τους βάσει της κατάστασης που διαβάσαν. Αντίστοιχα με πριν οι οντότητες του δεύτερου δακτυλίου αφού αρχικοποιηθούν αναλαμβάνουν να δημιουργήσουν τις οντότητες του τρίτου δακτυλίου (δρομολογητές εργασιών) και στη συνέχεια μεταφέρουν τον έλεγχο στο δακτύλιο αυτόν.




**Σχήμα 6.14c.** Ανάνηψη έπειτα από αποτυχία – Γ' φάση

Όπως και πριν έτσι και τώρα οι οντότητες των δρομολογητών εργασιών διαβάζουν από το PersistentStorage το τελευταίο στιγμιότυπό τους και αρχικοποιούν την κατάστασή τους. Αφού αρχικοποιηθούν, οι δρομολογητές εργασιών αναλαμβάνουν να δημιουργήσουν τους εκτελεστές εργασίας που ήταν ενεργοί και να περάσουν τον έλεγχο σε αυτούς.




**Σχήμα 6.14d.** Ανάνηψη έπειτα από αποτυχία – Δ' φάση

Τέλος, οι εκτελεστές εργασίας αναλαμβάνουν να διαβάσουν τη σωσμένη κατάστασή τους στην οποία συμπεριλαμβάνεται κι η ουρά εργασιών τους και να αρχικοποιήσουν την τρέχουσα κατάστασή τους βάσει αυτής. Στη συνέχεια οι εκτελεστές εργασιών είναι έτοιμοι να συνεχίσουν την εκτέλεση των εργασιών τους ξεκινώντας από την πρώτη ανολοκλήρωτη εργασία τους, χωρίς να απαιτείται η εκτέλεση όλων των ενεργειών από την αρχή.

Με την ολοκλήρωση της δημιουργίας και του 4$^{ου}$ δακτυλίου το runtime σύστημα είναι σε θέση να συνεχίσει τη λειτουργία του από μία συνεπή κατάσταση και μάλιστα αποφεύγοντας την ανάγκη επανεκτέλεσης των ήδη εκτελεσθέντων εργασιών. Επιπλέον,

το runtime σύστημα είναι σε θέση να δεχτεί και νέες αιτήσεις προς εξυπηρέτηση από τους χρήστες.

### 6.4.2.3 Persistent Storage

Όπως αναφέρθηκε τα συνεπή στιγμιότυπα των οντοτήτων του συστήματος σώζονται σε μόνιμο αποθηκευτικό μέσο από όπου και διαβάζονται εν συνεχεία στην περίπτωση ανάνηψης του server. Τη λειτουργικότητα του σωσίματος των καταστάσεων καθώς και της ανάκτησής τους την υλοποιεί το Persistent Storage module παρέχοντας κατάλληλο interface προς τις άλλες οντότητες.

### 6.4.2.4 Locking service

Η χρήση του συστήματος για την υποστήριξη web services που θα επιτρέπει σε πολλούς χρήστες ταυτόχρονα να εκτελούν ροές εργασίας εισάγει την ανάγκη για την προστατία της συνέπειας των παραγόμενων αποτελεσμάτων. Προς αυτόν το σκοπό έχει ληφθεί μέριμνα για τα προγράμματα εκείνα για τα οποία επιτρέπεται να υπάρχει μόνο μία διεργασία που τα εκτελεί κάθε στιγμή (επειδή π.χ. χρησιμοποιούν κάποια συγκεκριμένα προσωρινά αρχεία).

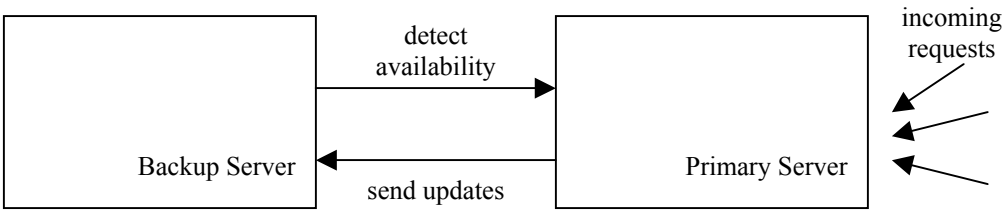
Το runtime σύστημα υλοποιεί έναν μηχανισμό κλειδώματος για την υποστήριξη εκτέλεσης προγραμμάτων που πρέπει να εκτελούνται σε απομόνωση, δηλ. προγραμμάτων που απαιτείται για τη σωστή εκτέλεσή τους να μην υπάρχουν παραπάνω από μία διεργασίες που να τα εκτελούν ταυτόχρονα στο ίδιο μηχάνημα. Η ανάγκη αυτή που προκύπτει για ένα πρόγραμμα δηλώνεται σαφώς στην περιγραφή της ροής εργασίας και συγκεκριμένα στην περιγραφή του προγράμματος που πρέπει να εκτελεστεί σε απομόνωση. Η δυνατότητα εκτέλεσης ενός τέτοιου προγράμματος αντιμετωπίζεται ως πόρος που παρέχεται κεντρικά σε έναν το πολύ εκτελεστή εργασίας κάθε στιγμή. Καθώς ένα τέτοιο πρόγραμμα μπορεί να συμμετέχει σε πολλά instances ροών εργασίας που εκτελούνται ταυτόχρονα σε διαφορετικές μηχανές εκτέλεσης απαιτείται ένας τέτοιος μηχανισμός να λειτουργεί σε επίπεδο runtime συστήματος επιτρέποντας το πολύ σε μία μηχανή εκτέλεσης να εκτελεί ένα «ευαίσθητο» πρόγραμμα σε κάθε στιγμή.

Ένας εκτελεστής εργασίας που θα αναλάβει την εκτέλεση ενός προγράμματος το οποίο πρέπει να εκτελεστεί σε απομόνωση θα ζητήσει αρχικά να «κλειδώσει» το δικαίωμα εκτέλεσης του προγράμματος από το μηχανισμό κλειδωμάτων. Αν το εν λόγω πρόγραμμα δεν έχει ήδη κλειδωθεί από κάποιον άλλον εκτελεστή εργασίας τότε ο εκτελεστής εργασίας αποκτά το δικαίωμα εκτέλεσης του προγράμματος και προχωρεί στη δρομολόγηση της εκτέλεσής του. Μετέπειτα αιτήσεις για την εκτέλεση του ίδιου προγράμματος από άλλους εκτελεστές εργασίας προτού απελευθερωθεί το κλείδωμα που έχει προηγουμένως τεθεί προκαλεί το μπλοκάρισμα αυτών οι οποίοι απέστειλαν αίτηση και την εισαγωγή τους σε μια ουρά εξυπηρέτησης με αλγόριθμο FCFS (First Come First Served). Με τον τερμάτισμο της εκτέλεσης του «ευαίσθητου» προγράμματος ο αντίστοιχος εκτελεστής εργασίας απελευθερώνει τον πόρο ο οποίος και δίνεται στον πρώτο εκτελεστή εργασίας που τυχόν είναι μπλοκαρισμένος στην ουρά για τον συγκεκριμένο πόρο.

Σκόπιμο είναι να τονιστεί πως η εισαγωγή κλειδωμάτων και το μπλοκάρισμα των εκτελεστών εργασίας που συνεπάγεται αυτή δεν οδηγεί στον κίνδυνο πιθανών deadlocks. Αυτό ισχύει γιατί δεν είναι δυνατό το ενδεχόμενο απαιτήσεων που οδηγούν σε κυκλικές σχέσεις μεταξύ των οντοτήτων. Ένας εκτελεστής εργασίας αναλαμβάνει την εκτέλεση μίας και μόνο εργασίας και μπορεί να ζητήσει το πολύ ένα κλείδωμα. Ο πόρος που θα δεσμεύσει ένας εκτελεστής εργασίας εν τέλει θα αποδεσμευτεί χωρίς αυτός να ζητήσει στον μεσοδιάστημα και κάποιο επιπλέον κλείδωμα για να συνεχίσει την εκτέλεσή του.

### 6.4.3 Fault Tolerance – Replication

Προκειμένου να επιτευχθεί υψηλή διαθεσιμότητα των υπηρεσιών του συστήματος υπάρχει η δυνατότητα χρησιμοποίησης ενός εφεδρικού server που αδιαφανώς αναλαμβάνει τη συνέχιση της λειτουργίας του κεντρικού server όταν διαπιστώσει πως αυτός παύει να ανταποκρίνεται. Χρησιμοποιείται η primary-backup τεχνική [98]. Σε αυτήν ο εφεδρικός server ενημερώνεται για όλες τις αλλαγές της κατάστασης στον κύριο server χωρίς όμως να εξυπηρετεί ο ίδιος αιτήσεις για όσο διάστημα λειτουργεί ως εφεδρικός. Η υπόθεση που κάνουμε είναι πως το δίκτυο που συνδέει τον κύριο με τον εφεδρικό server είναι αξιόπιστο έτσι ώστε ο εφεδρικός server να μπορεί να εντοπίσει επιτυχώς και με ασφάλεια τη μη διαθεσιμότητα του κύριου server.




**Σχήμα 6.15.** Σχήμα πρωτεύοντος – εφεδρικού server

Η σύνδεση του εφεδρικού server μπορεί να γίνει οποιαδήποτε στιγμή κατά τη διάρκεια λειτουργίας του κύριου server. Με την καταχώρηση του εφεδρικού server αποστέλλεται σε αυτόν από τον κύριο η τελευταία συνεπής κατάσταση που είναι διαθέσιμη για τις οντότητες του συστήματος. Από εκεί και πέρα ξεκινά η αποστολή "heartbeats" ανά τακτά χρονικά διαστήματα τα οποία και υποδηλώνουν πως ο κύριος server βρίσκεται «εν ζωή». Ο εφεδρικός server ορίζει ένα παραμετροποιήσιμο timeout που by default αντιστοιχεί σε 5 περίπου heartbeats μέσα στο οποίο αν δε λάβει κάποιο heartbeat θεωρεί τον κύριο ως μη διαθέσιμο πλέον. Όσο διάστημα ο κύριος server είναι ενεργός αποστέλει στον εφεδρικό τα στιγμιότυπα των οντοτήτων που σώζουν την κατάστασή τους διατηρώντας με αυτόν τον τρόπο αντίγραφο (replica) της σωσμένης κατάστασης.

Διαπιστώνοντας ο εφεδρικός server τη μη διαθεσιμότητα του κύριου server εισάγεται σε κατάσταση αναβάθμισης όπου προετοιμάζεται έτσι ώστε να αναλάβει τη συνέχιση της παροχής των υπηρεσιών ως ο νέος κύριος server. Η διαδικασία αυτή έχει πολλά κοινά στοιχεία με τη διαδικάσια ανάνηψης ενός server έπειτα από αποτυχία καθώς και στις δύο περιπτώσεις ανασυντίθεται η τελευταία συνεπής κατάσταση του server από ήδη υπάρχουσα κατάσταση. Από τη στιγμή που ο εφεδρικός server έχει εκτελέσει όλες τις απαραίτητες αρχικοποιήσεις έτσι ώστε να αναβαθμιστεί θεωρείται ως ο νέος κύριος server και διατηρεί το σύστημα διαθέσιμο. Προφανώς, ανά πάσα στιγμή μπορεί να συνδεθεί κάποιος εφεδρικός server στον νέο κύριο server.

### 6.4.4 Υποστήριξη Java Applets

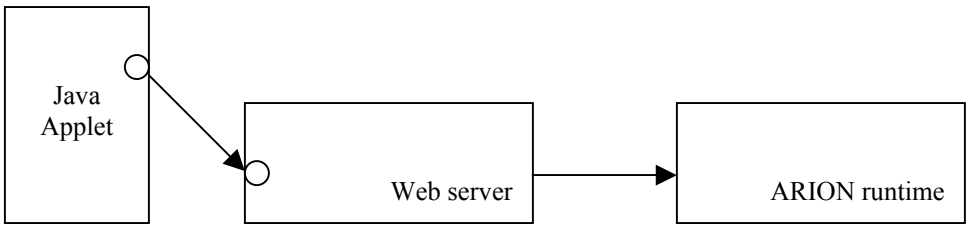
#### 6.4.4.1 Εισαγωγή

Το runtime σύστημα μπορεί να υποστηρίξει τη χρήση Java Applets [99] ως συστατικά στοιχεία μιας ροής εργασίας. Τα Java Applets παρουσιάζουν κάποιες ιδιαιτερότητες ως προς τα υπόλοιπα προγράμματα τα οποία μπορούν να εκτελεστούν κατά τη διάρκεια της εκτέλεσης μιας ροής εργασίας. Η ουσιαστική διαφοροποίηση σε σχέση με τα υπόλοιπα προγράμματα έγκειται στο γεγονός ότι τα Java Applets πρέπει να εκτελεστούν από τους χρήστες καθώς συνήθως περιλαμβάνουν ένα γραφικό περιβάλλον εργασίας μέσω του οποίου γίνονται κάποιες επιλογές από το χρήστη. Στα σενάρια τα οποία εξετάσαμε και θεωρήσαμε ως τα πλέον ρεαλιστικά κατά τη σχεδίαση του runtime συστήματος ένα Java Applet συνήθως χρησιμοποιείται στην αρχή της εκτέλεσης της ροής εργασίας, χωρίς αυτό να είναι δεσμευτικό, μέσω του οποίου ο χρήστης επιλέγει/ορίζει κάποιες αρχικές παραμέτρους ή/και παράγονται κάποια αρχεία εισόδου για τις επόμενες εργασίες.

#### 6.4.4.2 Σύνδεση με το runtime σύστημα

##### 6.4.4.2.1 Γενική περιγραφή

Η εκτέλεση των Java Applets καθώς και η επικοινωνία τους με το runtime σύστημα δε γίνεται με τη χρήση πρακτόρων όπως σε όλες τις υπόλοιπεςπεριπτώσεις εκτέλεσης εργασιών. Αντί πρακτόρων χρησιμοποιούμε έναν lightweight μηχανισμό επικοινωνίας που είναι εγκατεστημένος στο μηχάνημα όπου τρέχει ο web server από τον οποίον προέρχεται το applet. Αυτός ο μηχανισμός χρησιμοποιείται λόγω των περιορισμών ασφαλείας που εισάγει η γλώσσα προγραμματισμού Java στα Java Applets και οι οποίοι επιτρέπουν σε ένα applet να μπορεί να επικοινωνήσει μόνο με το μηχάνημα από το οποίο προήλθε, δηλαδή το μηχάνημα στο οποίο τρέχει ο web server. Έτσι, ο μηχανισμός επικοινωνίας δρα ως γέφυρα που ενώνει το applet με το runtime σύστημα.




**Σχήμα 6.16.** Επικοινωνία Java Applet με το runtime σύστημα

Μέσω αυτού του συνδέσμου το applet μπορεί:

- Να ενημερώνει το runtime σύστημα για τον τερματισμό της εκτέλεσής του. Καθώς η εκτέλεση του applet δεν είναι αυτοματοποιημένη χρειάζεται η ειδοποίηση για τον τερματισμό της λειτουργίας του να δωθεί προς το runtime σύστημα προκειμένου αυτό να προχωρήσει στη δρομολόγηση των επόμενων εργασιών.

- Να δημιουργεί αρχεία εξόδου τα οποία γράφονται στο μηχάνημα προέλευσης. Ένα applet που αποτελεί κομμάτι μιας ροής εργασίας χρειάζεται να μπορεί να παράγει κάποια αρχεία εξόδου τα οποία θα αποτελέσουν είσοδο σε επόμενες εργασίες. Καθώς περιορισμοί ασφαλείας δεν επιτρέπουν τη χρήση του συστήματος αρχείων του μηχανήματος του χρήστη ο οποίος χρησιμοποιεί το applet ο ενδιάμεσος κόμβος επιτρέπει τη χρήση του συστήματος αρχείων του μηχανήματος όπου τρέχει ο web server για την αποθήκευση της εξόδου του applet.

#### 6.4.4.2.2 Communication object

Η επικοινωνία του applet με τον ενδιάμεσο μηχανισμό επικοινωνίας πραγματοποιείται μέσω ενός communication object το οποίο αναλαμβάνει να προωθεί τα μηνύματα του applet στον communication server. Το communication object είναι ένα Java αντικείμενο που έχει ενσωματωμένη τη λειτουργικότητα της επικοινωνίας με τον communication server. Ο προγραμματιστής του applet μέσω ενός απλού API μπορεί να επικοινωνήσει με τον communication server και κατ'επέκταση με το runtime σύστημα χωρίς να απασχολείται με τις λεπτομέρειες του μηχανισμού επικοινωνίας.

Όταν αρχικοποείται το applet θα πρέπει να γίνονται γνωστά σε αυτό κάποια στοιχεία που αφορούν κάποια αναγνωριστικά που πρέπει να παρουσιαστούν προκειμένου το runtime σύστημα να είναι σε θέση να γνωρίζει σε ποιο instance ροής εργασίας αναφέρεται ένα μήνυμα από κάποιο applet. Τα στοιχεία αυτά παρέχονται από την Workflow Database που ανέλαβε να δημιουργήσει το instance της ροής εργασίες το οποίο έχει αποστείλλει προς εκτέλεση στο runtime σύστημα. Στο 7$^{ο}$ κεφάλαιο παρουσιάζεται ένα σενάριο χρήσης που ενσωματώνει ένα Java Applet όπου και γίνεται σαφής ο μηχανισμός απόδοσης των τιμών αυτών στο applet.

#### 6.4.4.2.3 Communication server

Ο communication server αποτελεί τον ενδιάμεσο σύνδεσμο του applet με το runtime σύστημα. Ο communication server πρέπει να είναι ενεργός στο μηχάνημα όπου τρέχει ο web server από τον οποίο προήλθε το applet. Ο ρόλος του είναι η υποστήριξη των λειτουργιών του applet καθώς και η ενημέρωση του runtime συστήματος για την πορεία εκτέλεσης του applet.

Η ακριβής λειτουργικότητα που παρέχεται από έναν communication server εξαρτάται από τις ανάγκες του applet που πρέπει να υποστηριχθεί. Από αυτό γίνεται σαφές ότι ένας communication server είναι προσαρμοσμένος πάνω στο συγκεκριμένο applet που υποστηρίζει κρύβοντας την πολυπλοκότητά του από το runtime σύστημα το οποίο μπορεί να χρησιμοποιεί τα applets χωρίς την ανάγκη κάποιων προσθηκών/αλλαγών σε αυτό. Τα στοιχεία που απαιτεί να γνωρίζει το runtime σύστημα σχετικά με τη λειτουργία του applet είναι:

i. τα παραγόμενα αρχεία εξόδου τα οποία γράφονται στο μηχάνημα του communication server και

ii. το πότε τερματίζει η εκτέλεση του applet προκειμένου να μπορέσει να προχωρήσει η εκτέλεση των επόμενων εργασιών.

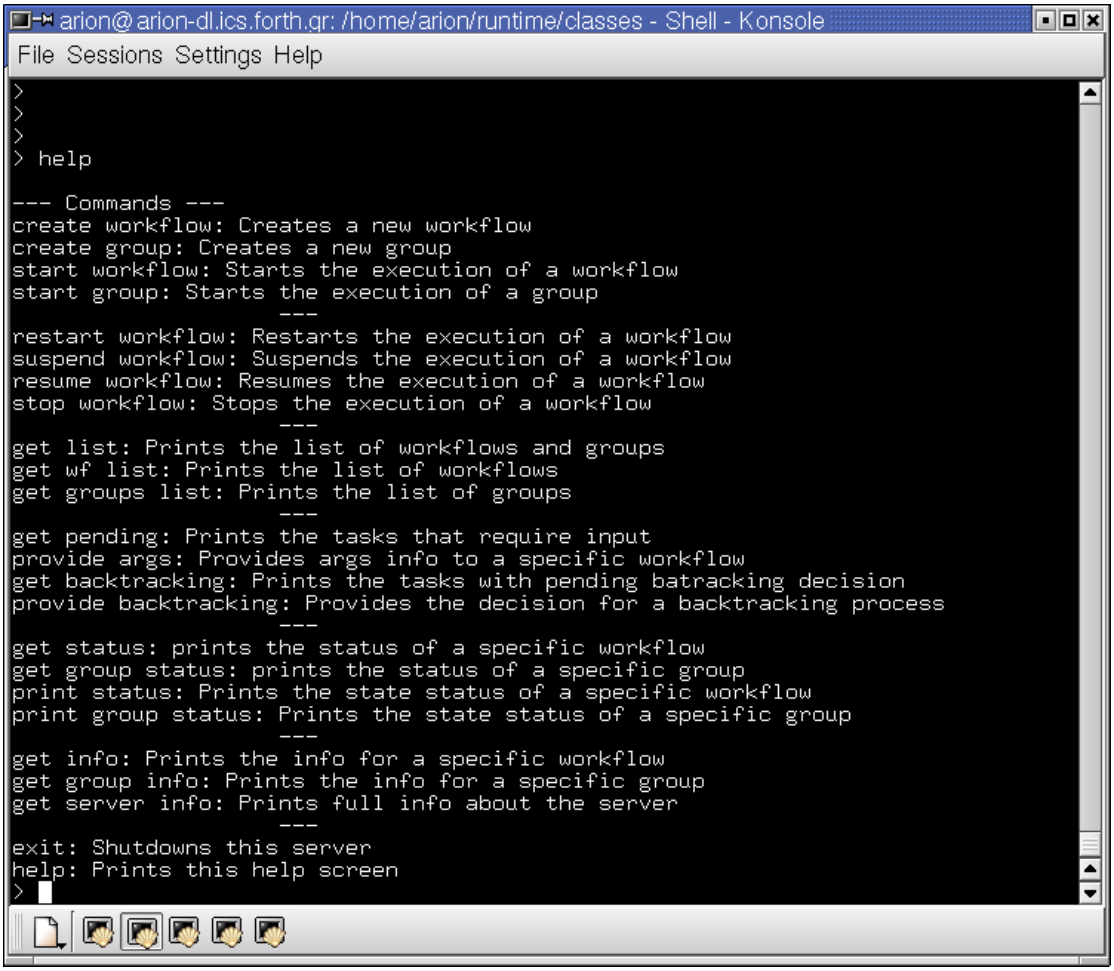
Αυτά τα στοιχεία παρέχονται από κάθε communication server προς το runtime σύστημα πέραν των άλλων λειτουργιών που μπορεί να επιτελεί ένας communication server.

## 6.5 Διαχείριση συστήματος

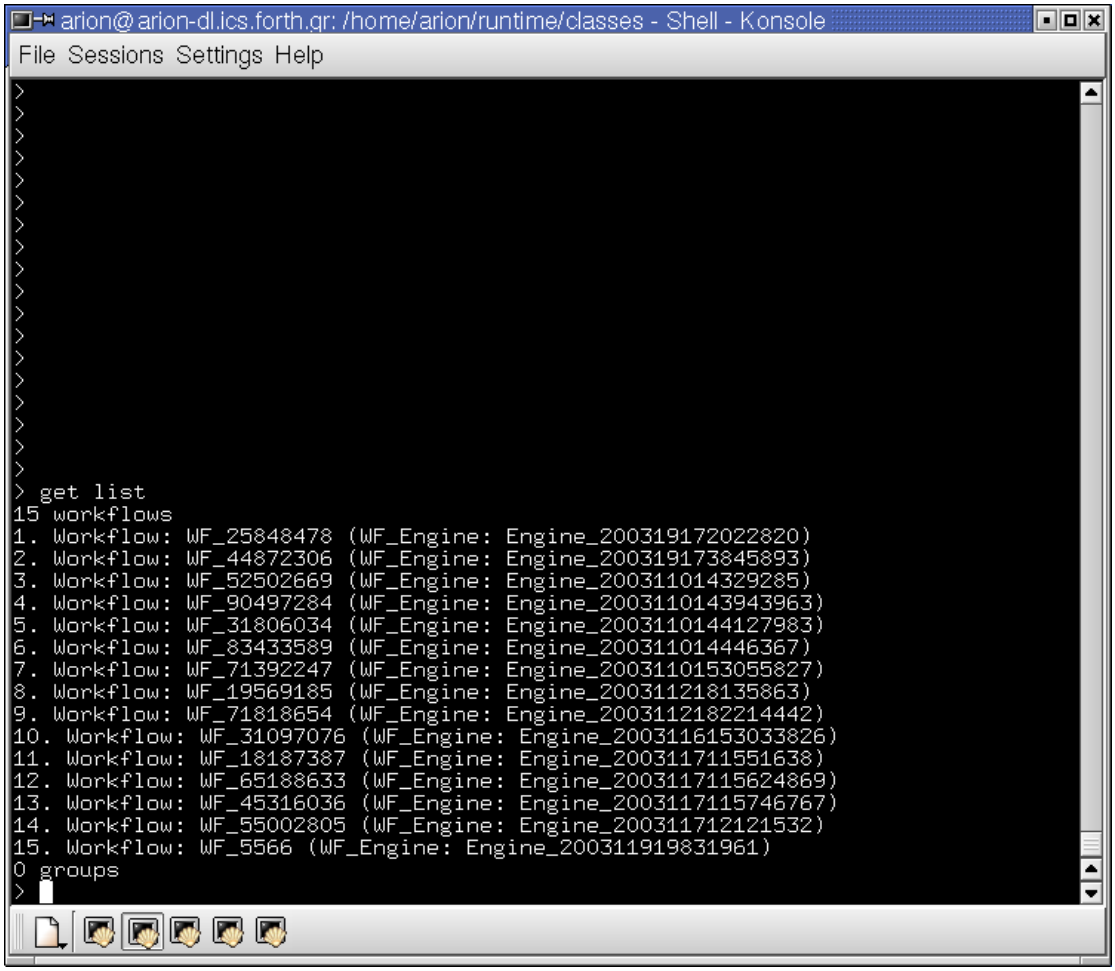
### 6.5.1 Command line utility

Η διαχείρηση του server κατά τη διάρκεια της λειτουργίας του μπορεί να γίνει από το διαχειριστή του συστήματος μέσω ένος command line utility που έχει υλοποιηθεί. Μέσω αυτού ο διαχειριστής μπορεί να λάβει πληροφορίες σχετικές με την τρέχουσα κατάσταση του server και των εκτελέσεων ροών εργασίας που εξυπηρετεί καθώς και να επέμβει στη λειτουργία του server και/ή κάποιας ροής εργασίας που εξυπηρετείται.

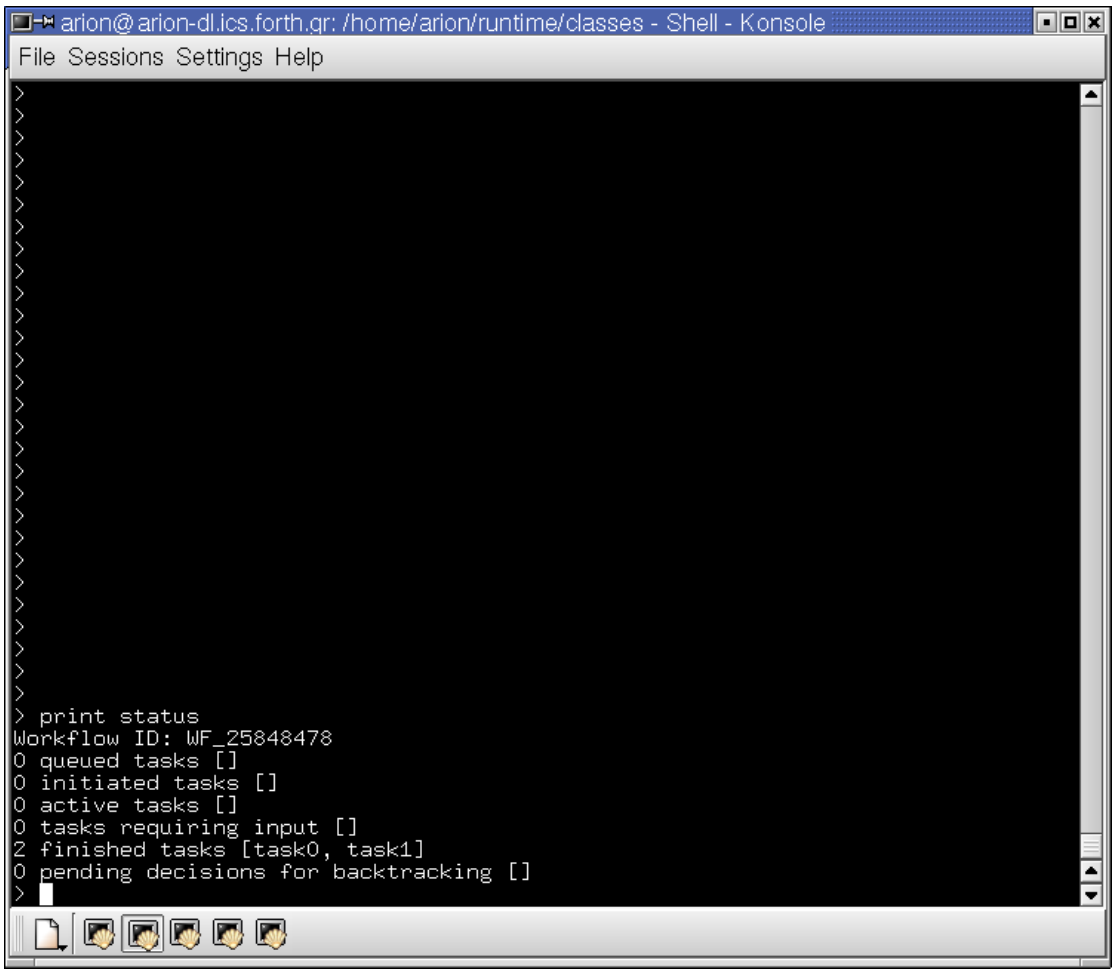
Μέσω αυτού μπορεί να παρακολουθεί την κατάσταση του συστήματος και να επεμβαίνει όποτε αυτό κρίνεται σκόπιμο, π.χ. σταματώντας την εκτέλεση μίας ροής εργασίας που καταναλώνει υπερβολικά πολλούς πόρους. Ακολουθούν κάποια screenshots από τη χρήση του εργαλείου.



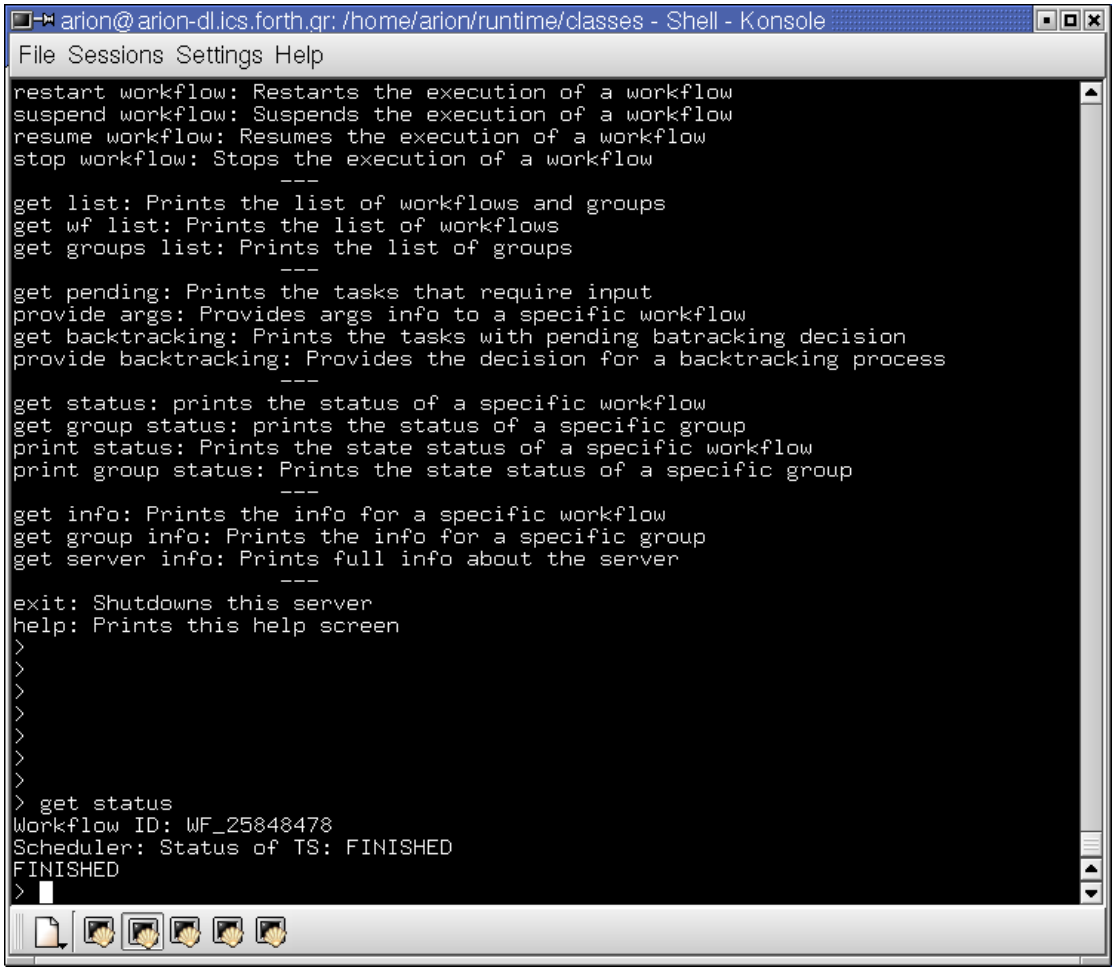

**Εικόνα 6.4.** Διαθέσιμες εντολές του command-line εργαλείου




**Εικόνα 6.5.** Απαρίθμηση ροών εργασίας στο σύστημα




**Εικόνα 6.6.** Τρέχουσα κατάσταση μίας ροής εργασίας




**Εικόνα 6.7.** Εμφάνιση της κατάστασης μίας ροής εργασίας

### 6.5.2 Logger

Ο server φροντίζει για την καταγραφή όλων των λειτουργιών που γίνονται κατά τη διάρκεια της λειτουργίας του. Η καταγραφή αφορά αφενός τις επιμέρους λειτουργίες που γίνονται κατά τη διάρκεια εκτέλεσης των ροών εργασίας από τις μηχανές εκτέλεσής τους και αφετέρου τις λειτουργίες διαχείρισης του συστήματος από τον διαχειριστή καθώς και τις απόπειρες μη έγκυρης χρήσης του.

Το ιστορικό εκτέλεσης μιας ροής εργασίας είναι διαθέσιμο τόσο για το διαχειριστή του συστήματος όσο και για τον χρήστη που ζήτησε την εκτέλεσή του. Σε αυτό καταγράφονται όλες οι βασικές λειτουργίες οι οποίες αφορούν την εκτέλεση της ροής

εργασίας, όπως αρχή-τέλος εκτέλεσης των επιμέρους εργασιών που απαρτίζουν τη ροή εργασίας, είσοδος παραμέτρων από το χρήστη κ.α.

Το ιστορικό του server περιλαμβάνει τις εντολές διαχείρισης που έχει δεχθεί, όπως suspend, resume, τις αιτήσεις για εκτέλεση ροών εργασιών κ.α.

## 6.6 Στοιχεία υλοποίησης

Η υλοποίηση του runtime συστήματος έγινε στη γλώσσα προγραμματισμού Java. Το σύνολο του κώδικα που χρησιμοποιείται για το σύστημα είναι περίπου 17000 γραμμές για το runtime σύστημα (420KB) και 2000 γραμμές για τα Java Servlets (50KB).

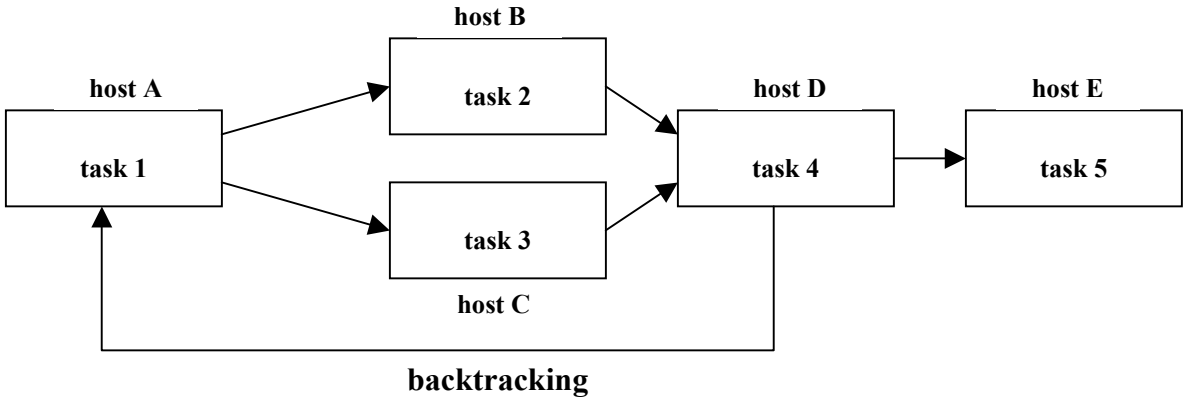
# Κεφάλαιο 7ο: Σενάρια Χρήσης

Το σύστημα υλοποιήθηκε και ελέγχθηκε βάσει παραδειγμάτων ροών εργασίας που συνδυάζουν προγράμματα μεταξύ τους με σχέσης εξάρτησης (control flow) οι οποίες και δεν αλλάζουν κατά τη διάρκεια της εκτέλεσής του. Τέτοιου είδους ροές εργασίας ονομάζονται στατικές σε αντιδιαστολή με τις λεγόμενες δυναμικές ροές εργασίας των οποίων οι περιγραφές, άρα και η συσχέτιση μεταξύ των διαφόρων εργασιών είναι δυνατόν να μεταβάλλεται κατά τη διάρκεια εκτέλεσής τους. Η υποστήριξη δυναμικών ροών εργασίας απαιτεί σύνθετους μηχανισμούς αυξημένης πολυπλοκότητας. Το σύστημα που υλοποιήθηκε στα πλαίσια της παρούσας εργασίας υποστηρίζει στατικές ροές εργασίας όπου επιπλέον υπάρχει και η δυνατότητα ανάδρασης (backtracking) η οποία και αποδίδει έναν δυναμικό χαρακτήρα στο σύστημά μας χωρίς να εισάγεται ιδιαιτέρως υψηλή πολυπλοκότητα σε αυτό.

## 7.1 Πρώτο σενάριο: Αλληλεπιδραστική εκτέλεση

### 7.1.1 Εισαγωγή

Η αλληλεπιδραστική εκτέλεση ροών εργασίας επιτρέπει στο χρήστη να αλληλεπιδρά με το σύστημα παρέχοντας την απαιτούμενη είσοδο (παραμέτρους προγραμμάτων, επιλογές backtracking) κατά τη διάρκεια της εκτέλεσής τους. Θα θεωρήσουμε ένα τυπικό παράδειγμα που παρουσιάζει με απλό τρόπο την υποστηριζόμενη λειτουργικότητα ως προς την εκτέλεση ροών εργασίας.




**Σχήμα 7.1.** Παράδειγμα ροής εργασίας σε αλληλεπιδραστική εκτέλεση

Στο σενάριο που εμφανίζεται στο σχήμα 7.1 παρουσιάζεται μία ροή εργασίας η οποία αποτελείται από 5 εργασίες (προγράμματα). Έστω ότι οι εργασίες αυτές δέχονται ορισμένες παραμέτρους κατά την κλήση τους, απαιτούν κάποια αρχεία εισόδου, ενώ παράγουν και κάποια αρχεία εξόδου :

- **task 1:**
    - arg1 (integer)
    - arg2 (double)
    - arg3 (string)

    Input: file1 (location: A)
    Output: file2, file3

- **task 2:**
    - arg1 (integer)
    - arg2 (integer)

    Input: file2, file3
    Output: file4

- **task 3:**
    - arg1 (double)
    - arg2 (double)

    Input: file2, file3
    Output: file5

- **task 4:**
    - arg1 (string)
    - arg2 (integer)

    Input: file4, file5
    Output: file6

- **task 5:**
    - arg1 (string)

    Input: file6
    Output: file7, file8

Σύμφωνα με το παραπάνω σχήμα στη ροή εργασίας εκτελείται πρώτα το task1, στη συνέχεια μπορούν να εκτελεστούν παράλληλα τα task2 και task3. Αφού τερματίσουν και οι δύο εργασίες μπορεί να εκτελεστεί το task4. Μετά το τέλος του task4 υπάρχει η δυνατότητα επιστροφής σε κάποια προηγούμενη εργασία (εδώ task1).

### 7.1.2 Εκτέλεση ροής εργασίας

Με την εκκίνηση εκτέλεσης της παραπάνω ροής εργασίας ο δρομολογητής εργασιών θα δρομολογήσει την εκτέλεση του task1. Ο εκτελεστής εργασίας που θα αναλάβει την εκτέλεση της εργασίας θα ζητήσει από τον δρομολογητή εργασιών τιμές για τις τρεις παραμέτρους του προγράμματος και θα παγώσει τη συνέχιση της εκτέλεσης της εργασίας. Ο χρήστης θα ειδοποιηθεί μέσω του Notification System για την απαίτηση παραμέτρων και μέσω του web interface θα μπορέσει να εισάγει τις επιθυμητές τιμές. Έστω, ότι δίνει τις τιμές 5, 1.1 και "string1" για τις τρεις παραμέτρους αντίστοιχα. Η εκτέλεση της εργασίας συνεχίζεται και αφού διαπιστωθεί ότι δεν απαιτείται μεταφορά κάποιων αρχείων (το απαιτούμενο αρχείο εισόδου είναι ήδη στο host A) στη συνέχεια ένας κινούμενος πράκτορας θα εκτελέσει το πρόγραμμα task1 με παραμέτρους που δώθηκαν:

*%> task1 5 1.1 "string1"*

Η εκτέλεση θα έχει ως αποτέλεσμα την παραγωγή δύο αρχείων (file2, file3). Αφού τερματιστεί η εκτέλεση της πρώτης εργασίας στη συνέχεια θα δρομολογηθούν οι εργασίες task2 και task3. Αντίστοιχα με πριν οι εκτελεστές εργασίας θα ζητήσουν τις απαιτούμενες παραμέτρους που όταν παρασχεθούν από τους χρήστες οι εκτελεστές

εργασίας θα προχωρήσουν στη μεταφορά των αρχείων file2 και file3 από το host A στο host B και στο host C.

Στη συνέχεια τα δύο προγράμματα θα εκτελεστούν με τις παραμέτρους που δώθηκαν από το χρήστη, π.χ.

*%> task2 5 6*
*%> task3 1.2 1.4*

και θα παραχθούν τα αρχεία file4 και file5.

Στη συνέχεια και αφού τερματίσει η εκτέλεση και των δύο εργασιών θα δρομολογηθεί η εκτέλεση του task4. Θα ζητηθούν οι παράμετροι εκτέλεσής του και στη συνέχεια θα μεταφερθούν τα αρχεία file4 και file5 στο host D από το host B και host C αντίστοιχα. Η εκτέλεση του task4 θα δώσει ως έξοδο το αρχείο file6.

*%> task4 "string2" 14*

Καθώς, σύμφωνα με το σχήμα της ροής εργασίας το task4 έχει οριστεί να αποτελεί checkpoint μετά την περάτωση της εκτέλεσης της εργασίας ζητείται από το χρήστη να επιλέξει μέσω του web interface να επιλέξει αν επιθυμεί να συνεχιστεί η εκτέλεση με το task5 ή αν επιθυμεί να επιστρέψει η εκτέλεση της ροής εργασίας πίσω στο task1. Αν ο χρήστης επιλέξει την επιστροφή της εκτέλεσης στο task1 τότε ο θα εκτελεστεί εκ νέου η πορεία από το task1 ως και ο task4 και ο κύκλος της εκτέλεσης θα επαναλαμβάνεται όσο ο χρήστης επιλέγει την επιστροφή στο task1 με την ολοκλήρωση της εκτέλεσης του task4. Μέσω αυτής της επαναληπτικής διαδικασίας είναι δυνατή η λειτουργία του calibration των παραμέτρων εισόδων των προγραμμάτων προκειμένου να ληφθεί μία επιθυμητή έξοδος από το task4.

Με την επιλογή της συνέχισης από το task4 θα δρομολογηθεί η τελευταία εργασία (task5) και αφου ζητηθεί και παρασχεθεί η απαιτούμενη παράμετρος θα μεταφερθεί το αρχείο file6 από το host D στο host E και θα εκτελεστεί το task 5 το οποίο θα έχει ως

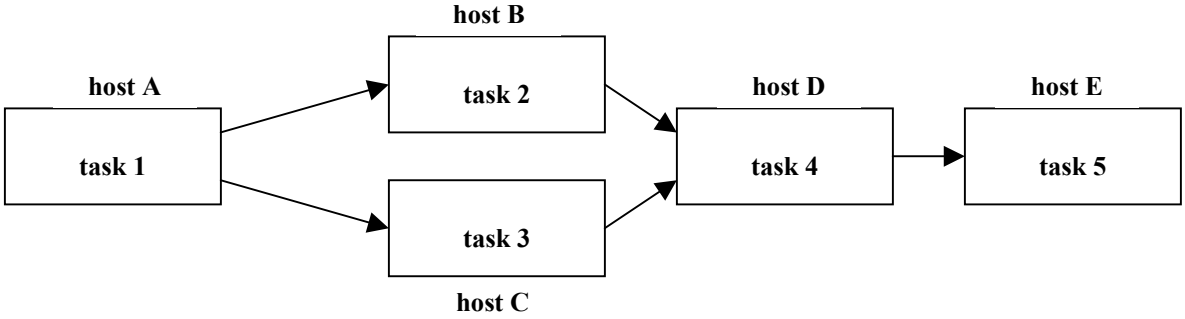
έξοδο τα file7 και file8. Από αυτό το σημείο η εκτέλεση της συγκεκριμένης ροής εργασίας θα έχει τερματιστεί.

*%> task5 "string3"*

## 7.2 Δεύτερο σενάριο: Προγραμματισμένη εκτέλεση

### 7.2.1 Εισαγωγή

Το runtime σύστημα υποστηρίζει και την προγραμματισμένη εκτέλεση ροών εργασίας κατά την οποία ο χρήστης δε χρειάζεται να παρέχει είσοδο. Οι τιμές των απαιτούμενων παραμέτρων έχουν δωθεί εξ'αρχής και χρησιμοποιούνται καθώς αυτές ζητούνται από τους εκτελεστές εργασίας. Στην αυτοματοποιημένη κατάσταση εκτέλεσης η λειτουργία του backtracking απενεργοποιείται. Έτσι, η ροή εργασίας που παρουσιάστηκε στην προηγούμενη παράγραφο σε κατάσταση αυτοματοποιημένης λειτουργίας μετατρέπεται στην εξής μορφή:




**Σχήμα 7.2.** Παράδειγμα ροής εργασίας σε αυτοματοποιημένη εκτέλεση

Στο παραπάνω σχήμα έχει αφαιρεθεί η δυνατότητα του backtracking. Προκειμένου να δωθεί η δυνατότητα εκτέλεσης της λειτουργίας του calibration των παραμέτρων εισόδου, αντίστοιχα όπως και στην κατάσταση αλληλεπιδραστικής εκτέλεσης, ο χρήστης έχει τη δυνατότητα να ορίσει έναν αριθμό από instances ροών εργασίας όπου το κάθε instance θα χρησιμοποιεί διαφορετικές παραμέτρους προκειμένου να είναι δυνατόν να εξεταστεί η παραγόμενη έξοδος της ροής εργασίας για ένα εύρος τιμών παραμέτρων εισόδου.

### 7.2.2 Εκτέλεση group ροών εργασίας

Η εκτέλεση του group των instances ροών εργασίας που έχει ορίσει ο χρήστης απαιτεί την παροχή των απαιτούμενων παραμέτρων για κάθε instance εξ'αρχής. Αν ένας χρήστης απαιτούσε την εκτέλεση π.χ. πέντε instances της παραπάνω ροής τότε θα πρέπει να παρείχε και τις απαιτούμενες παραμέτρους για καθένα από αυτά. Αυτό θα είχε ως αποτέλεσμα τη δημιουργία των παρακάτω δεδομένων που περνιούνται ως παράμετροι στην αίτηση εκτέλεσης του group που αποστέλλεται από την Workflow Database:

*#all lines starting with # are treated as comments*

*# parameters for workflow instance 1*
*task1 arguments arg1="5 " arg2="1.1" arg3="string1"*
*task2 arguments arg1="1" arg2="2"*
*task3 arguments arg1="5.1" arg2="5.2"*
*task4 arguments arg1="string2" arg2="2"*
*task5 arguments arg1="string3"*
*-------*
*# parameters for workflow instance 2*
*task1 arguments arg1="6 " arg2="1.2" arg3="string1b"*
*task2 arguments arg1="1" arg2="2"*
*task3 arguments arg1="5.1" arg2="5.2"*
*task4 arguments arg1="string2" arg2="2"*
*task5 arguments arg1="string3"*
*-------*
*# parameters for workflow instance 3*
*task1 arguments arg1=" 7" arg2="1.3" arg3="string1c"*
*task2 arguments arg1="2" arg2="0"*
*task3 arguments arg1="5.5" arg2="5.5"*
*task4 arguments arg1="string2" arg2="2"*
*task5 arguments arg1="string3"*
*-------*
*# parameters for workflow instance 4*
*task1 arguments arg1="7 " arg2="1.4" arg3="string1d"*
*task2 arguments arg1="2" arg2="1"*

*task3 arguments arg1="5.5" arg2="5.7"*

*task4 arguments arg1="string2" arg2="2"*

*task5 arguments arg1="string3"*

*-------*

*# parameters for workflow instance 5*

*task1 arguments arg1="7 " arg2="1.5" arg3="string1e"*

*task2 arguments arg1="3" arg2="3"*

*task3 arguments arg1="5.55" arg2="5.8"*

*task4 arguments arg1="string2" arg2="2"*

*task5 arguments arg1="string3"*

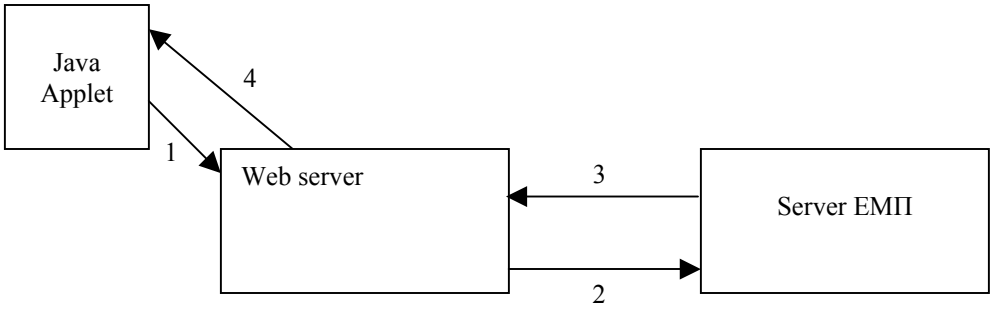
Μετά το πέρας της εκτέλεσης όλων των instances ο χρήστης ειδοποιείται για τον τερματισμό εκτέλεσης του group και μπορεί να εξετάσει τα αποτελέσματα που έχουν παραχθεί και από αυτά να μπορέσει να συγκρίνει την καταλληλότητα των παραμέτρων που χρησιμοποιήθηκαν.

## 7.3 Τρίτο σενάριο: Χρήση Java Applet

### 7.3.1 Εισαγωγή

Το τρίτο σενάριο που παρουσιάζεται αφορά τη χρήση Java applet στην περιγραφή μίας ροής εργασίας. Όταν αρχικοποιείται το applet διαβάζει ορισμένα αναγνωριστικά IDs που του έχουν περαστεί ως παράμετροι από την HTML σελίδα που το περιείχε. Αυτά τα αναγνωριστικά παρουσιάζονται στη συνέχεια όταν το applet επικοινωνήσει με το runtime σύστημα ώστε να είναι σαφές σε ποιο instance ροής εργασίας αναφέρεται το applet.

Ένα από τα πραγματικά σενάρια που εκτελούνται με τη χρήση του συστήματός μας αφορά μία ροή εργασίας που έχει οριστεί από μία ερευνητική ομάδα του Εθνικού Μετσόβειου Πολυτεχνείου (ΕΜΠ) που ασχολείται με ωκεανογραφία.




**Σχήμα 7.3.** Παράδειγμα ροής εργασίας που ενσωματώνει ένα Java Applet

### 7.3.2 Εκτέλεση σεναρίου

Το σενάριο περιλαμβάνει αρχικά τη χρήση ενός Java applet για την επιλογή από το χρήστη κάποιων γεωγραφικών σημείων για τα οποία επιθυμεί να παραχθούν στατιστικά στοιχεία για παραμέτρους όπως ύψη κύματος κ.α. Αφού μέσω του γραφικού περιβάλλοντος του applet ο χρήστης επιλέξει τα σημεία που επιθυμεί να μελετήσει τότε παράγεται ένα αρχείο που περιέχει σχετικά στοιχεία με τις επιλογές του χρήστη το οποίο και αποστέλλεται σε έναν server του ΕΜΠ όπου υπάρχει εγκατεστημένο το πρόγραμμα στατιστικής ανάλυσης γεωγραφικών σημείων. Αυτό το πρόγραμμα παράγει μία σειρά από αρχεία εξόδου (ASCII αρχεία + αρχεία εικόνων) τα οποία στη συνέχεια μεταφέρονται στο μηχάνημα από όπου προέρχεται το applet προκειμένου τα αρχεία αυτά να παρουσιαστούν στο χρήστη.

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο για την επικοινωνία του applet με το runtime χρησιμοποιείται ένας communication server στο μηχάνημα όπου τρέχει ο web server από όπου προέρχεται το applet. Μέσω αυτού το applet γράφει στο μηχάνημα όπου τρέχει ο web server τα δεδομένα με τις επιλογές του χρήστη και λαμβάνει τα αποτελέσματα που παράγονται τελικά.

# Κεφάλαιο 8ο: Συμπεράσματα,Συζήτηση– Μελλοντικές Κατευθύνσεις

## 8.1 Ανακεφαλαίωση

Η παρούσα εργασία ασχολήθηκε με την ανάπτυξη ενός συστήματος εκτέλεσης ροών εργασίας το οποίο επιτρέπει την παρακολούθηση και τη διαχείριση τους από το χρήστη μέσω ενός απλού web interface. Το σύστημα που υλοποιήθηκε υποστηρίζει χαρακτηριστικά που αφενός εγγυώνται τη σωστή λειτουργία του και αφετέρου προσδίδουν σημαντική αξία σε αυτό μέσω της παροχής εξελιγμένων δυνατοτήτων. Δώθηκε μεγάλη βαρύτητα στην απόκρυψη της πολυπλοκότητας της σχεδίασης από το χρήστη και της παροχής απλότητας στη χρήση απαιτώντας μόνο την ύπαρξη ενός web browser.

Τα κυριότερα σημεία που θεωρούμε ότι προσδίδουν ιδιαίτερη αξία στην εργασία αυτή είναι τα ακόλουθα:

- **Υποστήριξη δύο καταστάσεων εκτέλεσης ροών εργασίας (αλληλεπιδραστική και αυτοματοποιημένη).** Οι δύο αυτές διαφορετικές καταστάσεις επιτρέπουν στους χρήστες την εκτέλεση ροών εργασίας είτε επιδρώντας άμεσα με τη μηχανή εκτέλεσης της ροής εργασίας, επηρρεάζοντας ακόμα και την πορεία εκτέλεσής του είτε προγραμματίζοντας την εκτέλεση ενός πλήθους ροών εργασίας οι οποίες εν συνεχεία εκτελούνται χωρίς την ανάγκη παρέμβασης

- **Υποστήριξη Java Applet.**

- **Ύπαρξη εφεδρικού server.** Η δυνατότητα χρήσης εφεδρικού server επιτρέπει στο runtime σύστημα να ανεβάζει τη διαθεσιμότητά του σε αρκετά ψηλά ποσοστά.

- **Ανοχή σε λάθη.** Έχει υλοποιηθεί λειτουργικότητα που επιτρέπει την ανάνηψη ενός server και της συνέχιση της εκτέλεσης του από την τελευταία συνεπή κατάσταση που έχει σωθεί, αποφεύγοντας την ανάγκη επανεκτέλεσης χρονοβόρων διαδικασιών.

## 8.2 Μελλοντικές Επεκτάσεις

Η εργασία αυτή επιδέχεται αρκετές προσθήκες/βελτιώσεις που θα επιτρέπουν στο σύστημα να είναι περισσότερο λειτουργικό και εύχρηστο από τον τελικό χρήστη. Η υλοποίηση που έγινε στα πλαίσια της εργασίας επικεντρώθηκε κυρίως στην υποστήριξη της απαιτούμενης λειτουργικότητας, ενώ δεν ασχολείται σε σημαντικό βαθμό με θέματα που αφορούν τους τρόπους παρουσίασης της παραγόμενης πληροφορίας.

Συγκεκριμένα, στην παρούσα φάση η παρουσίαση της κατάστασης εκτέλεσης ή του ιστορικού εκτέλεσης μιας ροής εργασίας γίνεται μέσω της χρήσης κειμένου. Εδώ θα μπορούσε να υποστηριχθεί η δυνατότητα της χρησιμοποίησης κάποιων γραφικών εργαλείων που θα αναπαριστούσαν με πιο εύληπτο τρόπο τη σχετική πληροφορία. Μια ιδέα προς αυτήν την κατεύθυνση είναι η χρήση ενός χάρτη εκτέλεσης της ροής εργασίας που θα παρουσιάζει γραφικά την κατανομή των εργασιών της ροής εργασίας, τονίζοντας τα ενεργά στοιχεία κάθε στιγμή καθώς και τις μετακινήσεις που προκαλεί το control και το data flow σε πράκτορες και data sets αντίστοιχα.

Επίσης, στο παρόν σύστημα η παρουσίαση των ενδιάμεσων αποτελεσμάτων που παράγονται από τη εκτέλεση των ροών εργασίας δεν υποστηρίζεται. Ο χρήστης μπορεί να ελέγξει τα ενδιάμεσα αποτελέσματα που παράγουν οι ροές εκτός ARION και εν συνεχεία να επιλέξει π.χ. στην περίπτωση του backtracking προς τα ποια κατεύθυνση θα κινηθεί η πορεία εκτέλεσης μιας ροής εργασίας. Βέβαια, η υποστήριξη του visualiazation της εξόδου προγραμμάτων γενικά δεν είναι μια τετριμμένη διαδικασία. Θα μπορούσε σαφώς όμως να υποστηριχθεί η παρουσίαση ορισμένων τύπων δεδομένων, όπως ASCII αρχεία, εικόνες bmp, jpeg, gif κ.α. με τη χρήση ίσων κάποιων φίλτρων που θα αναπροσαρμόζουν την προς απεικίνιση πληροφορία σε μορφή κατάλληλη για απεικόνιση μέσω ενός web browser ή ακόμα και Java applet.

# REFERENCES


1. Workflow Management Coalition-WFMC homepage, http://www.wfmc.org.

2. P. Lawrence. *WFMC Workflow Handbook 1997*. John Wiley & Sons, ISBN 0-471-96947, 1997.

3. G. Alonso, D. Agrawal, A. El Abbadi, C. Mohan, M. Kamath and R. Guuenthoer. Exotica/FMQM: A Persistent Message-based Architecture for Distributed Workflow Management. In *Proc. IFIP Working Conference on Information Systems Development for Decentralized Organizations*, 1995.

4. S. Ceri, P. Grefen and G. Sanchez. WIDE: A Distriduted Architecture for Workflow Management. In *Proc 7th Int'l Workshop on Research Issues in Data Engineering*, 1997.

5. Corba Technology homepage, http://www.corba.org.

6. A. Sheth, K.Kochut, J. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch and I. Schevchenko. Supporting State-Wide Immunization Tracing Using Multi-Paradigm Workflow Technology. In *Proc. VLDB Conference*, 1996.

7. J. Weissenfels, D. Wodtke, G. Weikum, A. Kotz Dittrich. The Mentor Architecture for Enterprise-wide Workflow Management. In: *NSF Workshop on Workflow and Process Automation in Information Systems*, Athens, GA, May 1996.

8. D. Georgakopoulos, M. Hornick, A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2), 1995.

9. C. Bauzer Medeiros, G. Vossen, M. Weske. WASA: A Workflow-Based Architecture to Support Scientific Database and ExpertSystems (Extended Abstract). In *Proc. of 6th International Conference on Database and ExpertSystems Applications (DEXA) 1995*, London, Springer LNCS 978, 574-583, 1995.

10. J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, G. Yost et. al.. The PIF Process Interchange Format and Framework Version 1.2. The Knowledge Engineering Review, 13 (1998) 1, Cambridge University Press, March 1998, pp. 91-120.

11. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits-Systems and Computer*, 8(1): 21-66, 1998.

12. N. Adam, V. Atluri and W. Huang. Modeling and Analysis of Workflows using Petri Nets. *Journal of Intelligent Information Systems*, 10(2): 131-158, 1998.

13. C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan editor, *Application and Theory of Petri Nets 1993*, 691:1-16, 1993.

14. C. Reisig and G. Rosenberg, editors, Lecture on Petri Nets I : Basic Models, Volume 1491 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998.

15. Carlsen. Conceptual Modeling and Composition of Flexible Workflow Models. PhD Thesis, Information Systems Group. Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim 1997.

16. F. Leymann and W. Altenhuber. Managing Business Processes as an Information Resource. *IBM Systems Journal*, 33 (1994) 2, pp. 326-348.

17. R. Medina-Mora, T. Winograd, R. Flores, F. Flores. The Action Workflow Approach to Workflow Management Technology. In: *CSCW'92: Proceedings of the Conference on Computer Supported Cooperative Work*, ACM Publishers, New York, pp. 281-288.

18. N.S. Glance, D.S. Pagani, R. Pareschi. Generalized Process Structure Grammars (GPSG) for Flexible Representations of Work. In: M. Ackerman (Ed): *CSCW'96: Proceedings of the Conference on Computer Supported Cooperative Work*. Boston, MA, 1996, pp. 180-189.

19. P. Senge: *The 5th Discipline-The art and practice of the learning organization*. Century Business Publishers, London, 1990.

20. O.A. Buhkres, J. Chen, A.K. Elmagarmid, X. Liu and J.G. Mullen. Interbase: A Multi-Database Prototype System. In *Proceeding of the ACM Sigmod Conference on Management of Data*, 1993.

21. N. Krishnakumar and A. Sheth. Managing Heterogeneous Multi-System Tasks to Support Enterprise-Wide Operations. *Distributed and Parallel Databases*, 3(2), April 1995.

22. F. Leyman and D. Roller. Business Process Management with FlowMark. In *Proc. 39th IEEE Computer Society Int'I Conf. (CompCon)*, 1994.

23. The IBM homepage. http://www.ibm.com.

24. D. Chan, J. Vonk, G. Sanchez, P. Grefen, P. Apers. A Specification Language for the WIDE Workflow Model. In *Proc. 7th Int'l Workshop on Advanced Information Systems Engineering (CaiSE)*. 1997.

25. W.M.P. van der Aalst and A. Kumar. XML Based Schema Definition for Support of Inter-Organizational Workflow. University of Colorado and University of Eindhoven Report, 2001.

26. Workflow Definition Language 2, http://www.cs.helsinki.fi/group/design/wdl2-toteutus-1.1.html.

27. Workman – Workflow Manager, http://www.cs.helsinki.fi/group/workflow/design.

28. XLink specification, http://www.w3.org/TR/xlink/.

29. World Wide Web Consortium homepage, http://www.w3.org/.

30. *Workflow Process Definition Interface – XML Process Definition Language*, a Workflow Management Coalition Specification, Document Number WFMC-TC-1025.

31. Workflow Management Coalition. Workflow Client Application (Interface 2) Application Programming Interface (WAPI) Specification. Document Number WFMC TC-1009. Winchester 1997.

32. J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, G. Yost et. al.. The PIF Process Interchange Format and Framework Version 1.2. The Knowledge Engineering Review, 13 (1998) 1, Cambridge University Press, March 1998, pp. 91-120.

33. M. R. Genesereth: Knowledge Interchange Format – draft proposed American National Standard (dpANS). NCITS.T2/98-004. Online Paper as of 1999-09-04. http://logic.stanford.edu/kif/dpans.html.

34. National Institute of Standards and Technology-NIST homepage, http://www.nist.giv/.

35. N.S. Glance, D.S. Pagani, R. Pareschi. Generalized Process Structure Grammars (GPSG) for Flexible Representations of Work. In: M. Ackerman (Ed): *CSCW'96: Proceedings of the Conference on Computer Supported Cooperative Work*. Boston, MA, 1996, pp. 180-189.

36. Rational, Inc. et al.: UML Notation Guide version 1.1 OMG Document ad/97-08-05. Framingham, MA 1997.

37. Rational, Inc. et al.: UML Semantics version 1.1. OMG Document ad/97-08-04. Framingham, MA 1997.

38. Rational, Inc. et al.: UML Summary version 1.1. OMG Document ad/97-08-03. Framingham, MA 1997.

39. P. Hruby. Specification of Workflow Management Systems with UML. *Proceedings of the 1998 OOPSLA Workshop on Implementation and Application of Object-oriented Workflow Management Systems*, Vancouver, BC 1998.

40. O. Wiegert. Business Process Modeling and Workflow Definition with UML-Deficiencies and Actions to Improve. Presentation at the OMG Meeting Manchester, 1998-03-31.

41. Michigan http://www.si.umich.edu/UMDL.

42. The Intelligent Integration of Information Initiative. http://mole.dc.isx.com/I3

43. THETIS: A Data Management and Data Visualization System for Coastal Zone Management of the Mediterranean Sea. Contact person C. Houstis http://kos.ics.forth.gr:8000/

44. POSEIDON: A Distributed Information System for Ocean Processes. Contact person N. M. Patrikalakis. http://czms.mit.edu/poseidon/

45. V. Christophides, C. Houstis, S. Lalis, H. Tsalapata, "Ontology-driven Integration of Scientific Repositories", NGITS'99, New Generation Information Technologies, Lecture Notes in Computer Science, Elsevier, Habart Habaron, Israel, July 1999

46. Houstis, S. Lalis, N.M. Patrikalakis, W. Cho, "Federated Scientific Information Systems", position paper for the invitational workshop for the EU-NSF cooperation on Large Scientific Database Systems, http://www.carc.caltech.edu/euus/documents/houstis.html

47. The DECIMA project http://www.hrwallingford.co.uk/projects/DESIMA/

48. The DECAIR project http://www-air.inria.fr/decair

49. SIMS http://www.isi.edu/sims

50. Mikrokosmos Ontology http://crl.nmsu.edu/users/mahesh/onto-intro-page.html

51. The Extensible Markup Language (XML) standard. http://www.w3.org/XML

52. Description Logic http://www.cs.rmit.edu.au/dl

53. Microsoft, Microsoft ODBC 3.0 Software Development Kit and Programmer's Reference. Microsoft Press, February 1997.

54. The JDBC Data Access API. http://www.javasoft.com/products/idbc/index.html

55. The Digital Library Initiative. http://www.cise.nsf.gov/iis/dli_home.html

56. Information Manifold http://www.research.att.com/~levy/imhome.html

57. The Knowledge Sharing Effort. http://www-ksl.stanford.edu/knowledge-sharing

58. The HDF standard. http://hdf.ncsa.uiuc.edu

59. InfoSleuth http://www.mcc.com:80/projects/infosleuth

60. Ontolingua http://www-ksl.stanford.edu/knowledge-sharing/ontolingua

61. The Resource Description Framework. http://www.w3.org/Metadata/RDF

62. Description Logics. http://dl.kr.org

63. Denning, OLE Controls Inside Out. Microsoft Press, 1995

64. D. Curtis, Java, RMI and CORBA. http://www.omg.org/library/wpjava.htm

65. The JDBC Data Access API. http://www.javasoft.com/products/idbc/index.html

66. Ontosaurus http://www.isi.edu/isd/ontosaurus.html

67. Knowledge Interchange Format http://logic.stanford.edu/kif/kif.html

68. Santa Barbara http://alexandria.sdc.ucsb.edu

69. The CDF standard. http://nssdc.gsfc.nasa.gov/cdf

70. The Dublin Core. http://purl.org/metadata/dublin-core

71. Stanford http://www-diglib.stanford.edu

72. GARLIC http://www.almaden.ibm.com/cs/showtell/garlic

73. TSIMMIS http://www-db.stanford.edu/tsimmis

74. EnviroTech on-line. http://www.envirotech.org

75. Java: http://java.sun.com

76. Dave Zeltserman and Gerard Puoplo. “Building Network Management Tools With Tcl/Tk”. Prentice Hall. ISBN: 0130807273, April 1998

77. R. Kent Dybvig and Kent Dybbig. “The Scheme Programming Language: ANSI Scheme”. Prentice Hall. ISBN: 0134546466, April 1996

78. Grasshopper, http://www.grasshopper.de/

79. Crystaliz Inc, General Magic, GMD FOKUS, IBM, TOG: OMG Joint Submission “Mobile Agent System Interoperability Facility”, November 1997

80. FIPA: http://www.fipa.org/

81. Mike Bradley. “*IIOP: OMG’s Internet Inter-ORB Protocol: A Brief Description*”, May 1997

82. Java RMI:http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html

83. SSL 3.0 Specification: http://wp.netscape.com/eng/ssl3/

84. X.509 Public Key Infrastructure (PKI) for the Internet. http://www.ipa.go.jp/security/rfc/RFC3280-01EN.html

85. Aglets URL: http://www.trl.ibm.co.jp/aglets/

86. Lange, D. B., and M. Oshima, *Programming and Developing Java Mobile Agents with Aglets*, Addison-Wesley, Menlo Park, CA (1998).

87. Lange, D. B., M. Oshima, G. Karjoth and K. Kosaka, ``Aglets: Programming Mobile Agents in Java", *Proceedings of the International Conference on Worldwide Computing and Its Applications*, Tsukuba, Japan (March 1997), Lecture Notes in Computer Science 1274, Springer-Verlag, Berlin, Germany, 253-266.

88. Java Security Model: http://java.sun.com/products/jdk/1.2/docs/guide/security/

89. April Agent Platform: http://pellucid.ui.sav.sk/TR-2002-06.pdf

90. Jim Waldo. JiniTMtechnology architectural overview. Technical report, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 U.S.A., 1999. Available as http://www.sun.com/jini/whitepapers/architecture.html.

91. Tim Kindberg, John Barton, Je_ Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic. People, places, things: Web presence for the real world, 2000.

92. Ian Foster, Carl Kesselman, Je_rey M. Nick, and Steven Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. Draft 5, Mathematics and Computer Science Division, Argonne National Laboratory and Department of Computer Science, University of Chicago and Information Sciences Institute, University of Southern California and IBM Corporation, November 2002.

93. Sun Microsystems. Java message service, 2002.

94. Peter R. Pietzuch and Jean M. Bacon. Hermes: A distributed event-based middleware architecture. In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02), 2002.

95. Chris Grennhalgh. Equip: a software platform for distributed interactive systems. Technical report, The Mixed Reality Laboratory, University of Nottingham, 2001.

96. C. Mascolo, L. Capra, and W. Emmerich. An xml-based middleware for peer-to-peer computing, 2001.

97. Li Gong. JXTA: A network programming environment. IEEE Internet Computing, V 5:88-95, 2001.

98. R. Guerraoui, A. Schiper “Fault-Tolerance by Replication in Distributed Systems”, Proc. Reliable Software Technologies – Ada-Europe ’96, Springer Verlag, LNCS 1088, 1996.

99. http://java.sun.com/applets/