

Πληροφοριακό σύστημα Arethusa

Σχεδίαση & Υλοποίηση

Κρητικός Απόστολος (914)

akritiko@gmail.com

Προπτυχιακός φοιτητής

Επιβλέπων:

Σταμέλος Ιωάννης

stamelos@csd.auth.gr

Επίκουρος Καθηγητής
Τμήμα Πληροφορικής

2008

ΠΛΗΡΟΦΟΡΙΑΚΟ ΣΥΣΤΗΜΑ ARETHUSA

Σχεδίαση και Υλοποίηση

Κρητικός Απόστολος

(προπτυχιακός φοιτητής του

Τμήματος Πληροφορικής, Α.Π.Θ.)

Επιβλέπων

Ιωάννης Σταμέλος

(επίκουρος καθηγητής του

Τμήματος Πληροφορικής, Α.Π.Θ.)

Περιεχόμενα

Ευχαριστίες.....	7
1. Εισαγωγή	8
1.1. Η υπόσχεση των ευέλικτων μεθοδολογιών	8
1.2. Μέθοδοι και πρακτικές	9
1.2.1. Ακραίος Προγραμματισμός	9
1.2.2. Προγραμματισμός σε ζεύγη	10
1.2.3. Σχεδίαση με προέλεγχο	10
1.2.4. Μικρές παραδόσεις	10
2. Το έργο Arethusa	11
2.1. Εισαγωγή	11
2.2. Ο χώρος του διυλιστηρίου – Ορολογία και Παραδοχές	11
2.2.1. Περιγραφή βασικών στοιχείων και οντοτήτων	12
2.2.2. Καταστάσεις Δεξαμενών	12
2.2.3. Κανόνες Λειτουργίας	13
2.3. Λειτουργικές απαιτήσεις	15
2.3.1. Με βάση τα δεδομένα	15
2.3.2. Με βάση τη λειτουργικότητα	16
2.4. Μη λειτουργικές απαιτήσεις	18
2.4.1. Βάσεις δεδομένων	18
2.4.2. OPC Clients / Servers	18
2.4.3. Περιβάλλον ανάπτυξης	18
2.5. Κέρδος	18
3. Σχεδιασμός	20
3.1. Η ανάθεση του έργου	20
3.2. Μια πρώτη προσέγγιση	21
3.3. Εισάγοντας ευφυΐα στο έργο	22
3.4. Arethusa – Αρχιτεκτονική	23
3.4.1. Γνωρίζοντας την δικτυακή υπηρεσία	23
3.4.2. Γνωρίζοντας τη διεπαφή χρήστη El Greco	25
3.4.3. Προϋπάρχοντα συστήματα	25
4. Υλοποίηση	27
4.1. Φάση 1 ^η – Πυροδότηση του έργου	27

4.1.1.	Η γέννηση του γραφικού περιβάλλοντος	28
4.1.2.	Αξιολόγηση της υλοποίησης κατά την πρώτη φάση	28
4.2.	Φάση 2 ^η – Επικοινωνία με Oracle Server	29
4.2.1.	Oracle 10g – Μονόδρομος	30
4.2.2.	Σχεδιάζοντας με βάση τις απαιτήσεις	30
4.2.3.	Παρακολούθηση σε πραγματικό χρόνο	32
4.2.4.	Άλλοι πίνακες	32
4.2.5.	Υλοποίηση και κώδικας	33
4.2.6.	Αξιολόγηση υλοποίησης κατά τη δεύτερη φάση	35
4.3.	Φάση 3 ^η – Επικοινωνία με OPC Server	35
4.3.1.	Σύστημα παρακολούθησης δεξαμενών	36
4.3.2.	OPC Server & OPC Clients	36
4.3.3.	Υλοποίηση και κώδικας	37
4.3.4.	Αξιολόγηση υλοποίησης κατά την τρίτη φάση	40
4.4.	Φάση 4 ^η – Μετατροπή εφαρμογής σε υπηρεσία	41
4.4.1.	Υλοποίηση και κώδικας	41
4.4.2.	Αξιολόγηση υλοποίησης κατά την τρίτη φάση	43
4.5.	Φάση 5 ^η – Ανάπτυξη ευφυούς συστήματος και ενσωμάτωση	44
4.5.1.	Γεγονότα	44
4.5.2.	Κανόνες	45
4.5.3.	RuleML	49
4.5.4.	NxBRE και Microsoft Visio	49
4.5.5.	Υλοποίηση και κώδικας	54
4.5.6.	Αξιολόγηση υλοποίησης της πέμπτης φάσης	54
5.	Έλεγχος	57
5.1.1.	Ανάθεση Λειτουργιών στο Μοντέλο	57
5.1.2.	Έλεγχος με Θεωρητικά και Πραγματικά Δεδομένα.	58
6.	Συμπεράσματα	61
6.1.1.	Πλεονεκτήματα	61
6.1.2.	Μειονεκτήματα	63
7.	Παραρτήματα	64
A.	Ονοματολογία	64
B.	Φόρμες Εργασίας	65
8.	Ευρετήρια	80

8.1.	Εικόνες.....	80
8.2.	Σχήματα	80
9.	Βιβλιογραφία.....	81

Ευχαριστίες

Μπορεί η εκπόνηση της παρούσας πτυχιακής εργασίας να ήταν ένα έργο δημιουργικό και ενδιαφέρον αλλά υπήρξαν στιγμές που έγινε κουραστικό και η περάτωσή του έμοιαζε απλησίαστη. Σε αυτό το σημείο θα ήθελα να αφιερώσω λίγες γραμμές σε όλους όσους πίστεψαν στην παρούσα εργασία και βοήθησαν στην περάτωσή της έμμεσα ή άμεσα.

Στον συμφοιτητή μου, φίλο και συνεργάτη Στέφανο Σκαλιστή που υπήρξε άψογος κατά τη διάρκεια της συνεργασίας μας στο τελευταίο αυτό κομμάτι των σπουδών μας και που με στήριξε όχι μόνον στα πλαίσια της εκπόνησης του έργου που αναλάβαμε αλλά και σε προσωπικό επίπεδο αποτρέποντας τον εκτροχιασμό μου. Τον ευχαριστώ γιατί χωρίς τη συμβολή του το αποτέλεσμα δεν θα έφθανε ποτέ, ποιοτικά, στο παρόν.

Στον κύριο Ιωάννη Σταμέλο, επίκουρο καθηγητή του τμήματος Πληροφορικής του Α.Π.Θ., που πίστεψε σε εμένα και μου έδωσε την ευκαιρία να δουλέψω πάνω σε μια δική μου πρόταση. Η βοήθειά του υπήρξε καταλυτικός παράγοντας στην ορθή και εμπρόθεσμη παράδοση της εργασίας μου αυτής.

Στους κυρίους Αποστολόπουλο Γρηγόριο, Κουτρολίκου Ευστάθιο και Τσιρίζη Αθανάσιο, στελεχών του ομίλου των Ελληνικών Πετρελαίων των Βιομηχανικών Εγκαταστάσεων Θεσσαλονίκης, που μου πρότειναν αυτήν τη συνεργασία, μου έδωσαν την ευκαιρία για μια δυναμική αρχή και με στήριξαν καθ' όλη τη διάρκεια εκπόνησης του έργου κάνοντας τα πάντα για να με διευκολύνουν στην υλοποίησή του. Κοντά τους μπόρεσα να δω την θεωρία που τόσα χρόνια μάθαινα να εφαρμόζεται στην πράξη.

Τέλος, θα ήθελα να πω ένα μεγάλο ευχαριστώ στην οικογένειά μου που στηρίζει πάντοτε τις προσωπικές μου επιλογές, τόσο υλικά όσο και ψυχολογικά και που ανέχεται την ιδιόρρυθμη σχέση που έχω με τον ηλεκτρονικό υπολογιστή μου με ότι αυτή συνεπάγεται. Χωρίς τη βοήθειά σας δεν θα έφθανα ποτέ εδώ που είμαι.

1. Εισαγωγή

1.1. Η υπόσχεση των ευέλικτων μεθοδολογιών

Είναι γνωστό ότι η επιστήμη της Πληροφορικής εξελίσσεται ραγδαία. Χρόνο με τον χρόνο διευρύνει κάποιους από τους τομείς της, κάνει την είσοδό της σε νέους και αποδεσμεύει αυτούς των οποίων ο κύκλος ζωής ολοκληρώθηκε εγκαταλείποντάς τους διακριτικά. Η ανάπτυξη λογισμικού, είναι καταδικασμένη να ακολουθεί προσπαθώντας να συγχρονίζεται με τις εξελίξεις της Πληροφορικής, παρέχοντας κάθε στιγμή κατάλληλα εργαλεία και υπηρεσίες για την ορθότερη, ταχύτερη και ευκολότερη σχεδίαση και παραγωγή λογισμικού, ή βελτιώνοντας ήδη υπάρχοντα.

Οι Ευέλικτες Μέθοδοι (Agile Methods), αποτελούν ένα σύνολο φρέσκων προτάσεων ανάπτυξης λογισμικού και στοχεύουν στην παράγωση κώδικα υψηλής ποιότητας ενώ υπόσχονται, σε όσους τις ακολουθήσουν, ταχύτητα και αποτελεσματικότητα μέσα σε ένα περιβάλλον ανάπτυξης ανθρωποκεντρικό και επικοινωνιακό. Παρά το ελκυστικό της πρότασής τους, οι ευέλικτες μέθοδοι αντιμετωπίζονται με σκεπτικισμό. Αυτό οφείλεται κυρίως στο γεγονός ότι προωθούν την προσαρμοστικότητα αφήνοντας την προβλεπτικότητα, που αποτελεί βασικό χαρακτηριστικό των κλασικών μεθόδων ανάπτυξης, να περάσει σε δεύτερη μοίρα. Επιπλέον εισάγουν την έννοια της *σχεδίασης με προέλεγχο*, η οποία επιβάλλει τη συγγραφή των tests του κώδικα πριν αυτός ακόμη παραχθεί, πράγμα το οποίο αντιβαίνει πλήρως στην πρακτική ελέγχου των συμβατικών μεθόδων ανάπτυξης. Τα χαρακτηριστικά αυτά αποτελούν δύο μόνο από μια πλειάδα λόγων που οδηγούν σε αμφισβήτηση της δυνατότητας των ευέλικτων μεθόδων να δώσουν λύσεις σε κρίσιμες εφαρμογές που επιβάλλουν τον περιορισμό του παράγοντα λάθους στο ελάχιστο δυνατό.

Η παρούσα εργασία παρακολουθεί την εφαρμογή της ευέλικτης μεθόδου του *Ακραίου Προγραμματισμού* (Extreme – XP – Programming) και των πρακτικών της, *Προγραμματισμός σε ζεύγη* (Pair Programming), *Σχεδίαση με προέλεγχο* (Test - First - Design) και *Μικρές Παραδόσεις* (Small Releases) κατά τον σχεδιασμό και υλοποίηση του πληροφοριακού συστήματος Arethusa και προσπαθεί να πιστοποιήσει την αποδοτικότητα, την αποτελεσματικότητα, την ευελιξία και την παραγωγικότητά τους. Το έργο Arethusa αποτελεί ένα προϊόν της συνεργασίας του Αριστοτέλειου Πανεπιστημίου Θεσσαλονίκης (Α.Π.Θ.) με τον όμιλο Ελληνικά Πετρέλαια (ΕΛ.ΠΕ) και προορίζεται για χρήση στις Βιομηχανικές Εγκαταστάσεις Θεσσαλονίκης (Β.Ε.Θ.).

1.2. Μέθοδοι και πρακτικές

Παρακάτω περιγράφονται σύντομα οι μέθοδοι και πρακτικές που αναφέρθηκαν στην εισαγωγή. Για εκτεταμένη θεωρητική θεμελίωση ο αναγνώστης μπορεί να ανατρέξει στην πτυχιακή εργασία του προπτυχιακού φοιτητή Σκαλιστή Στέφανου με τίτλο «*Εφαρμογή Ακραίου Προγραμματισμού στο Πληροφορικό Σύστημα Arethusa*» που αποτέλεσε συνεργάτη κατά την ανάπτυξη του έργου Arethusa.

1.2.1. Ακραίος Προγραμματισμός

Ο Ακραίος Προγραμματισμός (Extreme Programming, XP) αποτελεί μία μεθοδολογία ανάπτυξης λογισμικού, ίσως την πλέον διαδεδομένη από τις ευέλικτες μεθόδους. Διαφέρει από τις παραδοσιακές μεθοδολογίες στο γεγονός ότι θέτει την προσαρμοστικότητα (adaptability), αντί της προβλεπτικότητας (predictability), στο επίκεντρο καθ' όλη τη διάρκεια εκπόνησης του έργου. Στηρίζεται στις αρχές των μικρών κύκλων ανάπτυξης, των σαφέστατων αναφορών αποτελεσμάτων και της διαρκούς επικοινωνίας μεταξύ των συμβαλλόμενων. Περιλαμβάνει δε τις ακόλουθες πρακτικές:

- ✓ Το παιχνίδι της σχεδίασης (the planning game)
- ✓ Μικρά παραδοτέα (small releases)
- ✓ Αρχιτεκτονική εικόνα (metaphor)
- ✓ Απλή σχεδίαση (simple design)
- ✓ Σχεδίαση με προέλεγχο (test - first –design)
- ✓ Αναθεώρηση κώδικα (refactoring)
- ✓ Προγραμματισμός σε ζεύγη (pair programming)
- ✓ Συλλογική ιδιοκτησία (collective ownership)
- ✓ Διαρκής ενσωμάτωση κώδικα (continuous integration)
- ✓ Υποφερτός ρυθμός εργασίας (sustainable pace ή 40-hour week)
- ✓ Διαρκής παρουσία πελάτη (on-site customer)
- ✓ Προδιαγραφές κωδικοποίησης (coding standards)

1.2.2. Προγραμματισμός σε ζεύγη

Ο Προγραμματισμός σε ζεύγη (Pair Programming) είναι η ταυτόχρονη συνεργασία δύο ατόμων μπροστά στον ίδιο υπολογιστή. Το πρώτο ασχολείται με το σχεδιασμό, τη συγγραφή κώδικα και ελέγχων και ονομάζεται οδηγός Driver ενώ το άλλο είναι επιφορτισμένο με την τακτική επόπτευση των κινήσεων του οδηγού ενώ παράλληλα σκέφτεται "ένα βήμα μπροστά" και ονομάζεται Navigator.

1.2.3. Σχεδίαση με προέλεγχο

Περιλαμβάνει μια σειρά από ελέγχους που ονομάζονται *έλεγχοι αποδοχής* και πρέπει να είναι αυτοματοποιημένοι, δηλαδή να μην χρειάζεται η παρέμβαση κάποιου μέλους της ομάδας. Το πιο σημαντικό σε αυτή την πρακτική, είναι ότι οι έλεγχοι προέρχονται από τον πελάτη και ότι ένα μεγάλο μέρος τους γράφετε πριν ακόμη γραφεί ο κώδικας τον οποίο ελέγχουν.

1.2.4. Μικρές παραδόσεις

Ο Ακραίος Προγραμματισμός θεωρεί ότι πρέπει να γίνονται συχνά παραδόσεις στο πελάτη με σκοπό να δει όσο πιο νωρίς κάποιο κομμάτι του τελικού συστήματος έτσι ώστε να ανατροφοδοτήσει την παραγωγική διαδικασία. Έτσι με τις μικρές παραδόσεις ο ενδιαφερόμενος παρακολουθεί την πρόοδο του έργου από πρώτο χέρι και είναι σε θέση να ζητήσει κάποια αλλαγή σε περίπτωση που κάτι δεν τον ικανοποίησε.

Κάθε παράδοση πρέπει να περιέχει μόνο τις επιχειρησιακές απαιτήσεις οι οποίες έχουν συμφωνηθεί και χωρίς να περιέχει χαρακτηριστικά τα οποία δεν λειτουργούν απόλυτα σωστά. Το διάστημα μεταξύ διαδοχικών παραδόσεων πρέπει να είναι όσο πιο μικρό γίνεται χωρίς αυτό βέβαια να απαγορεύει στον πελάτη να αλλάξει κάποια από τις απαιτήσεις στην διάρκεια ενός τέτοιου κύκλου.

2. Το έργο Arethusa

2.1. Εισαγωγή

Όπως ήδη αναφέρθηκε, το έργο Arethusa αποτελεί μια συνεργασία του τμήματος Πληροφορικής του Αριστοτέλειου Πανεπιστημίου Θεσσαλονίκης με τον όμιλο Ελληνικά Πετρέλαια (ΕΛ.ΠΕ.), Βιομηχανικές Εγκαταστάσεις Θεσσαλονίκης (Β.Ε.Θ.), στα πλαίσια της πτυχιακής εργασίας των φοιτητών Κρητικού Απόστολου και Σκαλιστή Στέφανου.

Στόχος της συνεργασίας αυτής είναι η ανάπτυξη ενός πληροφοριακού συστήματος το οποίο θα χρησιμοποιηθεί από το τμήμα Προγραμματισμού του διυλιστηρίου υπό την επίβλεψη του τμήματος Πληροφορικής των ΕΛ.ΠΕ., Β.Ε.Θ. για να καλύψει ανάγκες παρακολούθησης, ελέγχου και καταγραφής στοιχείων, που αφορούν την τροφοδοσία του διυλιστηρίου.

Εν συντομία, το σύστημα θα πρέπει να συγκεντρώνει στοιχεία που σχετίζονται με τη λειτουργία των δεξαμενών αργού πετρελαίου, να καταγράφει αναλυτικό ιστορικό της κίνησης των δεξαμενών καθώς και των συνθέσεων τους σε μια βάση δεδομένων, να ελέγχει την σύμφωνη λειτουργία τους με τον ημερήσιο προγραμματισμό και τέλος στην περίπτωση που το διυλιστήριο τροφοδοτείται με κάποια σύνθεση που έχει χαρακτηριστεί ως «επικίνδυνη» να το αναφέρει στους χειριστές.

Το τμήμα του συστήματος που αφορά την καταγραφή και τον έλεγχο θα πρέπει να λειτουργεί συνεχώς και αυτόνομα σε αντίθεση με αυτό της παρουσίασης των στοιχείων στους χειριστές.

2.2. Ο χώρος του διυλιστηρίου – Ορολογία και Παραδοχές

Δεδομένης της πολυπλοκότητας και της έκτασης του έργου κρίνεται σκόπιμη η παράθεση της βασικής ορολογίας του χώρου του διυλιστηρίου καθώς και η καταγραφή κάποιων παραδοχών, έστω και σε αφαιρετικό επίπεδο, που χρησιμοποιούνται για την ορθή μοντελοποίηση του συστήματος.

2.2.1. Περιγραφή βασικών στοιχείων και οντοτήτων

Το σύστημα που καλούμαστε να υλοποιήσουμε είναι επιφορτισμένο με την παρακολούθηση διακίνησης M τύπων αργού πετρελαίου (ωστόσο μπορεί εύκολα να τροποποιηθεί ώστε να υποστηρίζει οποιονδήποτε τύπο υγρού). Με βάση τα παραπάνω το σύστημα αυτό οφείλει να έχει υπό την εποπτεία του:

- ✓ N αριθμό Δεξαμενών όπου η κάθε μία περιέχει μια συγκεκριμένη ποσότητα, κάθε δεδομένη χρονική στιγμή, από κάθε τύπο υγρού. Επίσης κάθε μία χαρακτηρίζετε είτε ως "Προς Εξωτερική Πηγή" είτε ως "Προς Διύλιση". Ο διαχωρισμός αυτός αναλύεται παρακάτω.
- ✓ Μια είσοδο (συγκεκριμένα το "Αγκυροβόλιο") από την οποία έρχεται ένας μόνο τύπος υγρού και κατευθύνεται προς μία κύρια δεξαμενή και μία δευτερεύουσα¹ η οποία λαμβάνει μικρές ποσότητες της εισόδου.
- ✓ Μια έξοδο (συγκεκριμένα η "Εξωτερική Πηγή") για αποστολή υγρών (αργών) σε κάποιο άλλον φορέα/πηγή στην οποία αποστέλλουν αποκλειστικά και μόνο οι δεξαμενές που χαρακτηρίζονται ως "Προς Εξωτερική Πηγή".
- ✓ Μια έξοδο (συγκεκριμένα η "Αποστακτική Στήλη") όπου γίνεται η επεξεργασία των υγρών (αργών) την οποία τροφοδοτούν αποκλειστικά και μόνο οι δεξαμενές που χαρακτηρίζονται ως "Προς Διύλιση".

Σημείωση: Στην υπόλοιπη έκταση του κείμενου, χάριν απλότητας, θα χρησιμοποιούνται αποκλειστικά οι όροι "αργό" ή "αργό πετρέλαιο", "Αγκυροβόλιο", "Εξωτερική Πηγή" "Αποστακτική Στήλη" αντί των όρων "υγρό", είσοδος, έξοδοι αντίστοιχα.

2.2.2. Καταστάσεις Δεξαμενών

Κάθε χρονική στιγμή όλες οι δεξαμενές βρίσκονται σε μια από τις καταστάσεις που αναφέρονται παρακάτω. Στην πλειονότητά τους μεταβάλλουν τα περιεχόμενα τους ή/και τα περιεχόμενα κάποιας άλλης δεξαμενής.

Οι καταστάσεις (αναφορικά) στις οποίες μπορεί να βρίσκεται κάποια δεξαμενή καθώς και η επίδρασή τους στην δεξαμενή αυτή είναι οι ακόλουθες:

¹ Η δευτερεύουσα αυτή δεξαμενή χρησιμοποιείται για να εξομαλύνει τις απότομες αλλαγές πίεσης κατά την λήψη από το Αγκυροβόλιο. Σε περίπτωση που δεν υφίστανται τέτοιες αλλαγές τότε απλώς αγνοείται.

1. Σε αναμονή (Stand By): Η δεξαμενή βρίσκεται σε αναμονή και δεν γίνεται καμία μεταβολή των περιεχομένων.
2. Προς Αποστακτική Στήλη (Refinery Feed - μόνο για δεξαμενές που χαρακτηρίζονται ως *Προς Αποστακτική Στήλη*): Η δεξαμενή τροφοδοτεί την *Αποστακτική Στήλη*. Ο όγκος κάθε τύπου αργού μειώνεται χωρίς να αλλάζει η αρχική σύσταση.
3. Προς Εξωτερική πηγή (Third Party Tank Feed - Μόνο για δεξαμενές που χαρακτηρίζονται ως *Προς Εξωτερική πηγή*): Η δεξαμενή αποστέλλει προς την *Εξωτερική πηγή*. Ο όγκος κάθε τύπου αργού μειώνεται χωρίς να αλλάζει η αρχική σύσταση.
4. Παραλαβή από Αγκυροβόλιο (Receiving): Η δεξαμενή παραλαμβάνει έναν τύπο αργού από το *Αγκυροβόλιο*. Ο όγκος του συγκεκριμένου τύπου αυξάνεται ενώ τον υπολοίπων παραμένουν σταθεροί.
5. Ενδοδιακίνηση Προς (Inner Transport To): Η δεξαμενή αποστέλλει προς μία άλλη δεξαμενή. Ο όγκος κάθε τύπου αργού μειώνεται χωρίς να αλλάζει η αρχική σύσταση.
6. Ενδοδιακίνηση Από (Inner Transport From): Η δεξαμενή παραλαμβάνει από μία άλλη δεξαμενή. Ο όγκος κάθε τύπου αργού αυξάνεται με βάση το συνολικό όγκο που παραλαμβάνεται πολλαπλασιασμένο με το ποσοστό του συγκεκριμένου τύπου αργού της δεξαμενής από την οποία παραλαμβάνει.
7. Παραλαβή Slop (Slop Return): Η δεξαμενή παραλαμβάνει υπολείμματα (Slop) τις επεξεργασίας τα οποία θα σταλθούν εκ νέου για διύλιση. Αυξάνεται μόνο ο όγκος του τύπου Slop.
8. Saab Off (Saab Off): Τη συγκεκριμένη χρονική στιγμή το υποσύστημα μέτρησης του όγκου (αναλύεται παρακάτω) δεν είναι ενεργό ή επέστρεψε μη αποδεκτή τιμή οπότε δεν μπορούμε να έχουμε πληροφόρηση σχετικά με τον συνολικό όγκο.
9. Εκτός Λειτουργίας / Συντήρηση (Error / Maintenance): Η δεξαμενή είτε συντηρείται είτε είναι εκτός λειτουργίας. Αυτό συμβαίνει όταν το σύστημα μέτρησης του όγκου δεν ήταν ενεργό ή επέστρεψε μη αποδεκτή τιμή συνεχόμενα για κάποιες διαδοχικές χρονικές στιγμές.

2.2.3. Κανόνες Λειτουργίας

Κάθε περιβάλλον διέπεται από τους κανόνες λειτουργίας του. Με στόχο τη μοντελοποίηση του περιβάλλοντος που καλείται να παρακολουθήσει το σύστημα Arethusa πρέπει να καθοριστούν οι κανόνες που διέπουν την λειτουργία όλων των εμπλεκόμενων οντοτήτων. Παρά το γεγονός ότι κάποιοι από αυτούς έχουν ήδη αναφερθεί, ακολουθεί μια πλήρης και αναλυτική περιγραφή όλων των κανόνων.

Όσον αφορά τα αργά, θεωρείται δεδομένο ότι δεν αλληλεπιδρούν μεταξύ τους σε καμιά περίπτωση, είτε πρόκειται για αργά που βρίσκονται σε κάποια δεξαμενή που βρίσκεται σε κατάσταση αναμονής, είτε για αργά που εισέρχονται, εξέρχονται ή ενδοδιακινούνται. Αν και το πλήθος των τύπων αργών είναι μεταβλητό θα πρέπει να περιλαμβάνεται και ο τύπος των υπολειμμάτων (Slor). Ο όγκος τους επηρεάζεται από την εξωτερική θερμοκρασία, όμως θεωρείται (βάσει Φυσικής) ότι καθίσταται δυνατό να υπολογισθεί σε συγκεκριμένη θερμοκρασία (15 °C).

Σχετικά με τις δεξαμενές, είναι γνωστό ότι το μίγμα αργών που περιέχουν είναι ομογενές. Από αυτό συνεπάγεται ότι κατά την οποιαδήποτε αποστολή αν και μεταβάλλεται ο όγκος κάθε τύπου αργού, το ποσοστό που περιέχει η δεξαμενή από τον κάθε τύπο αργού παραμένει σταθερό. Συνεπώς, καθίσταται εφικτός ο υπολογισμός των νέων όγκων με βάση το συνολικό όγκο που περιέχει η δεξαμενή. Ακόμη, όπως έχει ήδη αναφερθεί, μόνο οι δεξαμενές που έχουν χαρακτηριστεί ως *Προς Αποστακτική Στήλη* μπορούν να τροφοδοτούν με αργό πετρέλαιο την *Αποστακτική Στήλη*. Το ίδιο ισχύει και για την διάδραση των δεξαμενών που χαρακτηρίζονται ως *Προς Εξωτερική πηγή* με την *Εξωτερική πηγή*.

Στη διαδικασία τις ενδοδιακίνησης συμμετέχουν μόνο δύο δεξαμενές, δηλαδή πρόκειται για μια διεργασία «ένα προς ένα». Κατά την διάρκεια μίας ενδοδιακίνησης μεταξύ δύο δεξαμενών δεν είναι δυνατό να πραγματοποιηθεί κάποια άλλη ενδοδιακίνηση· οι υπόλοιπες διεργασίες όμως δεν επηρεάζονται.

Κατά τη διαδικασία διύλισης μπορούν να συμμετέχουν πολλές δεξαμενές, δηλαδή πρόκειται για μια διεργασία «πολλά προς ένα». Το ίδιο όμως δεν συμβαίνει για την αποστολή προς *Εξωτερική Πηγή* όπου μόνο μία δεξαμενή μπορεί να αποστέλλει κάθε χρονική στιγμή.

Κατά τη διαδικασία λήψης από το *Αγκυροβόλιο* συμμετέχουν, όπως έχει ήδη αναφερθεί, μία ή δύο δεξαμενές. Τόσο η κύρια όσο και η δευτερεύουσα δεξαμενή παραλαμβάνουν αυστηρά τον ίδιο τύπο αργού, αλλά σε διαφορετικές ποσότητες.

Τέλος, πρέπει να γίνει σαφές ότι μία δεξαμενή μπορεί να συμμετέχει μόνο σε μία διαδικασία κάθε δεδομένη χρονική στιγμή. Για παράδειγμα, δεν επιτρέπεται σε μια δεξαμενή να αποστέλλει προς τη *Αποστακτική Στήλη* και παράλληλα να ενδοδιακινεί.

2.3. Λειτουργικές απαιτήσεις

Παρακάτω παρατίθενται οι λειτουργικές απαιτήσεις του έργου στην τελική τους μορφή. Αξίζει, ωστόσο, να σημειωθεί ότι υπήρξε πλήθος αλλαγών όσον αφορά τις απαιτήσεις αυτές προτού καταλήξουν στη μορφή που αναλύεται παρακάτω. Οι αλλαγές αυτές προκάλεσαν κωλυσιεργίες στο έργο, καθυστερώντας το, άλλες περισσότερο και άλλες λιγότερο.

Είναι επίσης αξιοσημείωτο το γεγονός ότι τόσο κατά τη φάση του σχεδιασμού όσο και κατά τη φάση της υλοποίησης του έργου οι αλλαγές των απαιτήσεων που προέκυψαν αντιμετωπίστηκαν χωρίς να χρειασθεί να σχεδιαστεί εκ νέου κάποιο ήδη υλοποιημένο τμήμα του συστήματος. Έτσι οι επεμβάσεις περιορίστηκαν σε επίπεδα κώδικα.

Οι λειτουργικές απαιτήσεις χωρίζονται σκόπιμα σε δύο ομάδες με βάση τον προσανατολισμό τους στα δεδομένα ή τη λειτουργικότητα. Αυτός ο διαχωρισμός αφενός κάνει την καταγραφή τους ευκολότερη και βοηθά στην σφαιρικότερη αντίληψη και καλύτερη κατανόηση από τον αναγνώστη ενώ παράλληλα σκιαγραφεί τα πραγματικά ζητούμενα του πελάτη.

2.3.1. Με βάση τα δεδομένα

Συλλογή δεδομένων

Το σύστημα θα πρέπει να συλλέγει δεδομένα που αφορούν την διαφορά πίεσης (Δp) στην περιοχή προθέρμανσης του αργού καθώς και την ογκομετρική (m^3) τροφοδοσία του διωλιστηρίου. Επίσης θα πρέπει να αντλούνται δεδομένα (μέσω του OPC Server) σχετικά με τις στάθμες ή τον όγκο των δεξαμενών αργού μετά το πέρας κάποιου προαποφασισμένου παράθυρου χρόνου. Τέλος με βάση την καταγραφή ιστορικού θα πρέπει να ελέγχεται αν η σύνθεση αργών που τροφοδοτείται είναι παρόμοια με κάποια που έχει χαρακτηριστεί ως «επικίνδυνη».

Εισαγωγή δεδομένων

Το σύστημα θα πρέπει να δέχεται, μέσω κατάλληλης διεπαφής, τον ημερήσιο προγραμματισμό και σε περίπτωση οποιουδήποτε λάθους να είναι δυνατή η εισαγωγή όλων των απαραίτητων δεδομένων ώστε να συνεχίσει ορθά η λειτουργία του συστήματος.

Καταγραφή ιστορικού

Το σύστημα θα πρέπει να εισάγει στη βάση δεδομένων τη σύνθεση των αργών ανά δεξαμενή, την πληρότητά της δεξαμενής (με βάση τα κυβικά μέτρα που περιέχει), την χρονική στιγμή της μέτρησης και την τρέχουσα κατάστασή της.

Μεταβολή Ιστορικού

Το σύστημα θα πρέπει να είναι σε θέση να μεταβάλλει, στη βάση δεδομένων, τη σύνθεση των αργών ανά δεξαμενή, την πληρότητά της και την τρέχουσα κατάστασή της βάσει των στοιχείων που εισήχθησαν κατά την Εισαγωγή δεδομένων ως συμπληρωματικά για την ανάνηψη του συστήματος μετά από λάθος.

Παρουσίαση στοιχείων

Το σύστημα θα πρέπει να παρουσιάζει χρησιμοποιώντας μια διεπαφή τα ακόλουθα δεδομένα:

- ✓ Μία εποπτική αναπαράσταση του τμήματος του διυλιστηρίου το οποίο παρακολουθείται από το σύστημά μας. Αναλυτικότερα παρουσιάζονται χρωματικά οι καταστάσεις των δεξαμενών και οι ροές μεταξύ όλων των οντοτήτων (αγκυροβόλιο, δεξαμενές αργού, αποστακτική στήλη, ΟΚΤΑ).
- ✓ Λεπτομερή περιγραφή των στοιχείων της/των επιλεγμένης/ων οντότητας/ων.
- ✓ Όλων των ειδών τις επισημάνσεις (alerts).
- ✓ Ημερολόγιο (log) κινήσεων του χειριστή.
- ✓ Μεταβολή στοιχείων συστήματος
- ✓ Αυξομείωση του αριθμού των δεξαμενών.
- ✓ Αλλαγή του χαρακτηρισμού των δεξαμενών (τροφοδοτούσα, προς ΟΚΤΑ).

2.3.2. Με βάση τη λειτουργικότητα

Με γνώμονα τα παραπάνω το σύστημα είναι αναγκαίο να γνωρίζει σε πραγματικό χρόνο:

- ✓ Επ' ακριβώς το περιεχόμενο των δεξαμενών, και να το καταγράφει. Απαιτείται για τον υπολογισμό της σύνθεσης που τροφοδοτεί την αποστακτική στήλη, της σύνθεσης αργού που αποστέλλεται προς την *Εξωτερική Πηγή* και τις αλλαγές συνθέσεων σε περίπτωση ενδοδιακίνησης.

- ✓ Επ' ακριβώς τη σύνθεσης της τροφοδοσίας, και να την καταγράφει. Απαιτείται για τον έλεγχο προγραμματισμού – αποτελέσματος και για την αναφορά «κακής σύνθεσης» στην περίπτωση που μοιάζει πολύ σε κάποια σύνθεση που έχει χαρακτηριστεί ως επικίνδυνη.
- ✓ Επ' ακριβώς τα χαρακτηριστικά της αποστακτικής στήλης (διαφορά πίεσης (Δp) στην περιοχή προθέρμανσης του αργού και ογκομετρική (m^3) τροφοδοσία διυλιστηρίου) και να τα καταγράφει. Η ογκομετρική τροφοδοσία απαιτείται για την ανακάλυψη των δεξαμενών που τροφοδοτούν την αποστακτική στήλη και επαλήθευση του αν η παροχή των δεξαμενών αυτών συμφωνεί με την καταγεγραμμένη εισροή της αποστακτικής στήλης. Η διαφορά πίεσης απαιτείται για μελλοντική πρόβλεψη² επικίνδυνων συνθέσεων.

Επίσης απαιτείται να ελέγχονται τα παρακάτω:

- ✓ Αν ακολουθείται ο προγραμματισμός.
- ✓ Αν οι δεξαμενές λειτουργούν ορθά.
- ✓ Αν η τρέχουσα σύνθεση της τροφοδοσίας δεν πλησιάζει κάποια από τις χαρακτηρισθείσες ως επικίνδυνες.

Σε περίπτωση αστοχιών το σύστημα θα πρέπει να ενημερώνει κατάλληλα τους υπεύθυνους χειριστές.

Τέλος το σύστημα είναι επιφορτισμένο με την οπτική απεικόνιση των παρακάτω σε πραγματικό χρόνο:

- ✓ Τη στάθμη των δεξαμενών.
- ✓ Την κατάσταση των δεξαμενών.
- ✓ Των ροών, ποσοτικά και ποιοτικά, που εισέρχονται στο διυλιστήριο (από *Αγκυροβόλιο*).
- ✓ Των ροών, ποσοτικά και ποιοτικά, που εξέρχονται από το διυλιστήριο (προς *Εξωτερική Πηγή*).
- ✓ Των ροών, ποσοτικά και ποιοτικά, που ενδοδιακινούνται στο διυλιστήριο (από δεξαμενή σε δεξαμενή).
- ✓ Των ροών, ποσοτικά και ποιοτικά, που τροφοδοτούν την αποστακτική στήλη.

² Το συγκεκριμένο χαρακτηριστικό αποτελεί μελλοντική επέκταση του συστήματος για την εύρεση επικίνδυνων συνθέσεων εφαρμόζοντας τεχνικές ανακάλυψης γνώσης και ξεφεύγει από τα ζητούμενα του παρόντος έργου.

- ✓ Των ροών, ποσοτικά και ποιοτικά, που επιστρέφουν από την αποστακτική στήλη (υπό μορφή slop).

Μια προαιρετική απαίτηση είναι η εξαγωγή αναφορών (reports), ανά τακτά χρονικά διαστήματα ή μετά από αίτηση, της λειτουργίας του διυλιστηρίου σε μορφή ιστοσελίδας.

2.4. Μη λειτουργικές απαιτήσεις

Όσον αφορά τις μη λειτουργικές απαιτήσεις τα συστήματα που παρατίθενται στη συνέχεια επελέχθησαν από το τμήμα Πληροφορικής των ΕΛ.ΠΕ., Β.Ε.Θ. με βάση την τεχνολογία που χρησιμοποιεί ο όμιλος στο σύνολό του. Ενδεικτικά αναφέρουμε ότι η πρότασή μας ήταν να γίνει χρήση τεχνολογιών ανοικτού κώδικα και δωρεάν εργαλείων αλλά δεν έγινε αποδεκτή.

2.4.1. Βάσεις δεδομένων

- ✓ Oracle 10g
- ✓ Oracle 10g – Express Edition (χρήση για λόγους προσομοίωσης)

2.4.2. OPC Clients / Servers

- ✓ Matrikon OPC Simulation Server (χρήση για λόγους προσομοίωσης)
- ✓ Matrikon OPC Explorer (χρήση για λόγους προσομοίωσης)

2.4.3. Περιβάλλον ανάπτυξης

- ✓ Microsoft Visual Studio .NET 2005
- ✓ Microsoft .NET Framework 2.0

2.5. Κέρδος

Με γνώμονα την παραπάνω ανάλυση προκύπτουν τα παρακάτω κέρδη που αποτελούν και κίνητρα για την ανάπτυξη της εφαρμογής:

- ✓ Ταχύτερη και αποδοτικότερη παραγωγή του φύλλου προγραμματισμού.

Πλέον ο προγραμματισμός εισάγεται απευθείας στο σύστημα (μέσω κατάλληλης φόρμας) και αυτό αναλαμβάνει να παράγει το κατάλληλο έγγραφο (φύλλο

προγραμματισμού). Επίσης διασφαλίζει τον χειριστή ειδοποιώντας τον για το αν η σύνθεση που εισήγαγε χαρακτηρίζεται ως επικίνδυνη ή όχι.

✓ Δυνατότητα εποπτείας του διυλιστηρίου σε πραγματικό χρόνο.

Μέσω του γραφικού περιβάλλοντος του συστήματος ο χειριστής:

- Έχει άμεση εποπτεία όλων των πληροφοριών, οι οποίες απεικονίζονται συγκεντρωμένες, που του προσφέρει το σύστημα.
- Απολαμβάνει ποιοτική αναπαράσταση των παραπάνω πληροφοριών και ως εκ τούτου είναι σε θέση να αντιδρά ταχύτερα και αποτελεσματικότερα.
- Αυτό επιτρέπει στα τμήματα Λειτουργίας, Προγραμματισμού, Off-sites και Διεύθυνσης διυλιστηρίου να ενημερώνονται για την τροφοδοσία σε πραγματικό χρόνο.

✓ Δυνατότητα ακριβέστερου ελέγχου της πραγματικής τροφοδοσίας σε σχέση με την προγραμματισμένη.

Ο χειριστής ειδοποιείται σε πιθανή απόκλιση της πραγματικής από την προγραμματισμένη τροφοδοσία. Ως άμεσο επακόλουθο του 2 του δίνεται η δυνατότητα να επέμβει σε πραγματικό χρόνο και να κάνει τις απαραίτητες διορθώσεις.

✓ Δυνατότητα μέτρησης των απωλειών λόγω ανθρώπινου λάθους και επέμβασης κατά τη διάρκεια υλοποίησης των εντολών προγραμματισμού.³

✓ Δυνατότητα ανακάλυψης γνώσης που βοηθά στην έγκαιρη αναγνώριση επικίνδυνων συνθέσεων και αποφυγής τους κατά τον προγραμματισμό.⁴

³ Γνωρίζοντας το ανθρώπινο λάθος και το μέγεθός του, δίνεται η δυνατότητα στο τμήμα προγραμματισμού να σχεδιάσει τον επόμενο προγραμματισμό βασιζόμενο στην πραγματική τρέχουσα κατάσταση του διυλιστηρίου.

⁴ Το κομμάτι των ήδη καταγεγραμμένων «επικίνδυνων συνθέσεων» καθώς και η εισαγωγή νέων λαμβάνεται υπόψη από το προς υλοποίηση έργο, ενώ η ανακάλυψη νέων αποτελεί μελλοντικό χαρακτηριστικό που δύναται να υλοποιηθεί από το τμήμα Πληροφορικής των ΕΛ.ΠΕ., Β.Ε.Θ.

3. Σχεδιασμός

Ο σχεδιασμός του πληροφοριακού συστήματος Arethusa πέρασε από διάφορα στάδια πριν καταλήξει στην τελική του μορφή. Το κεφάλαιο αυτό μελετά την εξέλιξη του σχεδιασμού από την αφηρημένη αρχικά περιγραφή αναγκών σε καταγραφή συγκεκριμένων απαιτήσεων που μπορούσαν πλέον να μοντελοποιηθούν σε πληροφοριακό σύστημα.

3.1. Η ανάθεση του έργου

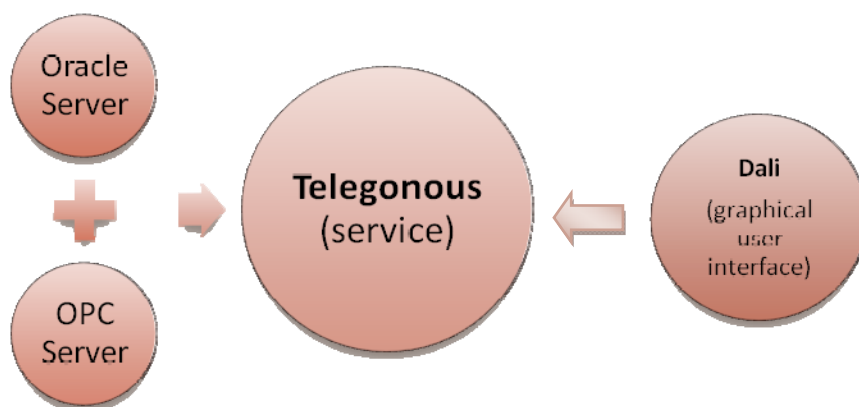
Προτού μιλήσουμε για τις πρώτες συνεντεύξεις που οδήγησαν στην καταγραφή των ουσιαστικότερων λειτουργικών απαιτήσεων είναι σημαντικό να περιγράψουμε σύντομα τους παράγοντες που πυροδότησαν τη συνεργασία των ΕΛ.ΠΕ. με το Α.Π.Θ. και οδήγησαν τελικά στην ανάθεση του έργου σε φοιτητές του.

Κατά το δίμηνο Ιουλίου – Αυγούστου του 2006 ο Κρητικός Απόστολος, προπτυχιακός φοιτητής του τμήματος Πληροφορικής του Α.Π.Θ., προσελήφθει από το τμήμα Πληροφορικής των ΕΛ.ΠΕ., Β.Ε.Θ., και απασχολήθηκε ως υπάλληλος με την ιδιότητα του ασκούμενου φοιτητή. Κατά τη διάρκεια της περιόδου αυτής ήρθε σε επαφή, μέσω του προϊσταμένου του κ. Γρηγορίου Αποστολόπουλου, με το τμήμα προγραμματισμού, το οποίο από καιρό μελετούσε τη δυνατότητα ανάπτυξης ενός πληροφοριακού συστήματος, εντός Β.Ε.Θ., με σκοπό κυρίως την ακριβέστερη παρακολούθηση της σωστής εφαρμογής του προγραμματισμού. Μετά από μια σειρά συζητήσεων και μια πρώτη εκτίμηση του μεγέθους του έργου αποφασίστηκε η υλοποίηση του να πραγματοποιηθεί στα πλαίσια της πτυχιακής εργασίας του Κρητικού Απόστολου εφόσον το τμήμα Πληροφορικής του Α.Π.Θ. ήταν σύμφωνο .

Η πυροδότηση της διαδικασίας εκπόνησης του έργου συνέβη με την αποδοχή του ρόλου ως επιβλέποντα, από τον κ. Ιωάννη Σταμέλο, επίκουρο καθηγητή του τμήματος Πληροφορικής του Α.Π.Θ. Μετά από συνάντηση του τελευταίου με το επιτελείο των ΕΛ.ΠΕ. και τον Κρητικό Απόστολο και αφού εξετάσθηκε εκ νέου το προς υλοποίηση πληροφοριακό σύστημα, αποφασίστηκε να ανατεθεί τελικά από κοινού στον Κρητικό Απόστολο και Σκαλιστή Στέφανο.

3.2. Μια πρώτη προσέγγιση

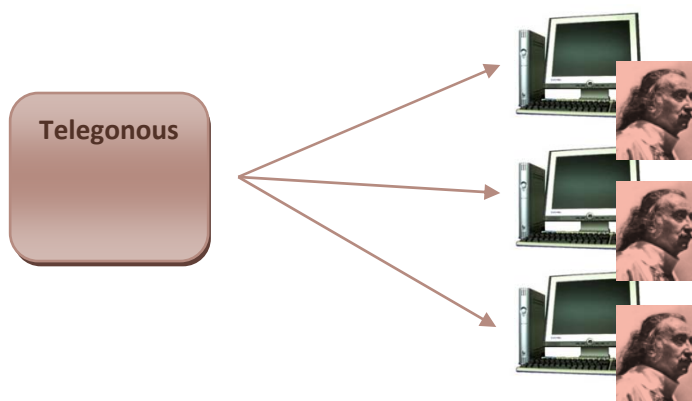
Οι πρώτες συνεντεύξεις, που κατέληξαν σε συγκεκριμένες απαιτήσεις, τοποθετούνται μεταξύ Δεκεμβρίου του 2006 και Ιανουαρίου του 2007. Σε αυτή τη φάση είχε γίνει ξεκάθαρο ότι το έργο θα έπρεπε να αποτελείται από δύο οντότητες, μια διαδικτυακή υπηρεσία και μια διεπαφή χρήστη. Η διαδικτυακή υπηρεσία θα συγκέντρωνε το σύνολο σχεδόν της λειτουργικότητας ενώ η διεπαφή χρήστη θα περιοριζόταν σε θέματα παρουσίασης πληροφορίας και ανάδρασης με τους χρήστες.



Σχήμα 3-1: Η Arethusa στην αρχική της μορφή.

Αναλυτικότερα ο ρόλος της διαδικτυακής υπηρεσίας θα ήταν η επικοινωνία με το σύστημα μέτρησης του όγκου των δεξαμενών, η εξαγωγή των επιθυμητών συμπερασμάτων και κατόπιν η παρουσίασή τους στο χρήστη καθώς και η καταχώρησή τους στη βάση δεδομένων εξυπηρετώντας σκοπούς τήρησης ιστορικού. Επιπλέον θα έπρεπε να είναι σε θέση να επικοινωνεί, μέσω δικτύου, με όλα τα στιγμιότυπα της διεπαφής χρήστη και να τους παρέχει τα δεδομένα προς απεικόνιση.

Η διεπαφή χρήστη η οποία, όπως τονίστηκε στην προηγούμενη παράγραφο, θα έπρεπε να επιτρέπει τη συνύπαρξη πολλαπλών στιγμιότυπων της, όφειλε να έχει τη δυνατότητα να εκτελείται σε κάποιον απομακρυσμένο υπολογιστή.

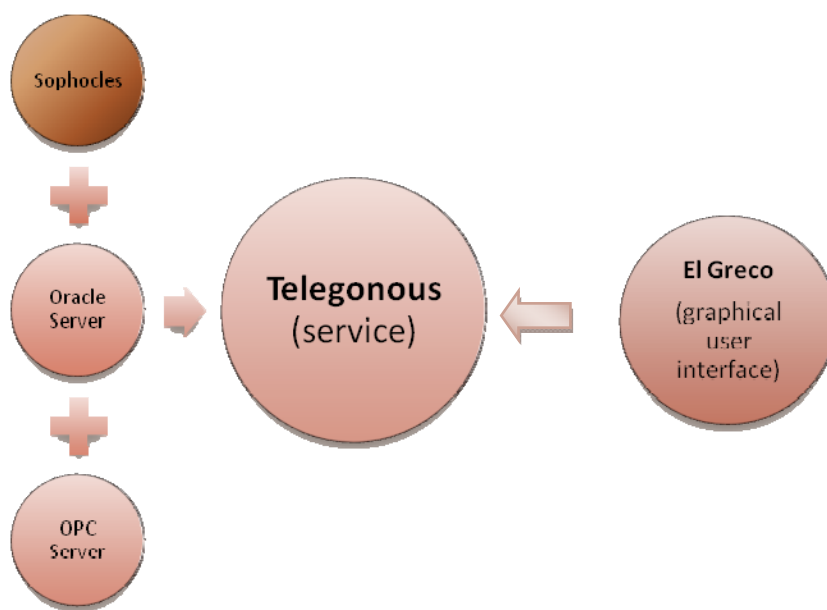


Σχήμα 3-2: Ο Telegonous τροφοδοτεί τρία στιγμιότυπα του Dalí

3.3. Εισάγοντας ευφυΐα στο έργο

Ενώ η υλοποίηση της παραπάνω αρχιτεκτονικής είχε ήδη ξεκινήσει, έκανε την εμφάνισή της, σχεδόν αμέσως, μια καινούρια απαίτηση. Η δημιουργία ενός ευφυούς συστήματος (ή καλύτερα υποσυστήματος) το οποίο θα αναλάμβανε τον χαρακτηρισμό όλων των δεξαμενών αργού πετρελαίου με βάση την τρέχουσα κατάστασή τους (βλ. *Οι καταστάσεις των Δεξαμενών - §2.2.2.*). Με την εισαγωγή της απαίτησης αυτής, όπως γίνεται εύκολα αντιληπτό, η αρχιτεκτονική του έργου χρειάστηκε να επαναπροσδιοριστεί. Το γεγονός ότι ο μέχρι τότε σχεδιασμός κατέληγε σε ένα πλήρως αρθρωτό σύστημα καθώς και η ορθογωνικότητα του υλοποιημένου κώδικα, απόρροια και τα δύο της υλοποίησης με χρήση ευέλικτων μεθόδων, επέτρεψαν την ομαλή προσθήκη των νέων στοιχείων στο ήδη υπάρχον μοντέλο αρχιτεκτονικής.

Τα παραπάνω είχαν σαν αποτέλεσμα το έργο, από άποψη αρχιτεκτονικής, να πάρει την ακόλουθη μορφή, η οποία και παρέμεινε σταθερή μέχρι το πέρας της υλοποίησης. Το σύστημα Arethusa χωρίζεται, όπως και πριν, στις ίδιες συνιστώσες. Η διαδικτυακή υπηρεσία (Telegonous), που αποτελεί τον πυρήνα εσωκλείει πλέον το ευφυές σύστημα (Sophocles) καθώς και δύο δυναμικές βιβλιοθήκες – «περιτυλίγματα» συστημάτων που προϋπήρχαν και αναλαμβάνουν τη σύνδεση και ορθή χρήση τους. Τα συστήματα Oracle Server και OPC Server (OLE for Process Control Server) προϋπήρχαν και χρησιμοποιούνται από την Arethusa για άντληση ή/και καταχώρηση πληροφοριών. Η διεπαφή χρήστη (που μετονομάστηκε από Dalí σε El Greco) διατήρησε το ρόλο και τη λειτουργία της.



Σχήμα 3-3: Η Arethusa μετά την εισαγωγή του ευφυούς συστήματος Sophocles.

3.4. Arethusa – Αρχιτεκτονική

Σε αυτό το σημείο θα ήταν καλό να ρίξουμε μια πιο κοντινή ματιά στο πληροφοριακό σύστημα. Αυτό θα μας επιτρέψει αφενός να κατανοήσουμε καλύτερα και σε μεγαλύτερο βαθμό τη λειτουργία των συστημάτων και υποσυστημάτων του έργου Arethusa και αφετέρου να επαληθευτεί η πληρότητα και ορθότητα της διαπραγμάτευσης των λειτουργικών απαιτήσεων όπως αυτές καταγράφονται στο δεύτερο κεφάλαιο.

3.4.1. Γνωρίζοντας την δικτυακή υπηρεσία

Πρόκειται ουσιαστικά για μια υπηρεσία φτιαγμένη για να τρέχει σε λειτουργικό σύστημα Windows (απαιτείται η ύπαρξη .NET Framework 2.0). Χωρίζεται σε δύο υποσυστήματα τον Telegonous, που αποτελεί τον πυρήνα του έργου, και τον Sophocles, το ευφύες σύστημα.

✓ Πυρήνας συστήματος (Telegonous)

Αναλαμβάνει να επικοινωνήσει με όσα συστήματα βρίσκονται εκτός του έργου Arethusa μέσω βιβλιοθηκών (components) – περιτυλιγμάτων. Έτσι πραγματοποιεί αφενός μια σύνδεση με το σύστημα SAAB, χρησιμοποιώντας έναν OPC Server για να πάρει από αυτόν πληροφορίες σχετικά με την κατάσταση και τη στάθμη της εκάστοτε

δεξαμενής αργού, και αφετέρου με το σύστημα PHD, χρησιμοποιώντας ως μεσάζοντα τον Oracle Server, για να αλιεύσει δεδομένα που αφορούν τη διαφορά πίεσης (Δp) στην περιοχή προθέρμανσης του αργού καθώς και την ογκομετρική (m^3) τροφοδοσία διυλιστηρίου. Ακόμη, μέσω του υποσυστήματος El Greco (βλ. παρακάτω), το οποίο αποτελεί την διεπαφή χρήστη, λαμβάνει τον τρέχοντα προγραμματισμό του διυλιστηρίου. Όσον αφορά το λειτουργικό κομμάτι, το σύστημα Telegonous είναι επιφορτισμένο με τα ακόλουθα:

1. Υπολογίζει τις νέες συνθέσεις με βάση την τρέχουσα κατάσταση της κάθε δεξαμενής.
2. Ελέγχει αν η λειτουργία του διυλιστηρίου όπως συμβαίνει σε πραγματικό χρόνο συμφωνεί με το αρχικό σχέδιο προγραμματισμού που δόθηκε από τον χειριστή του προγράμματος.
3. Καταγράφει στην βάση δεδομένων για κάθε δεξαμενή αργού τη σύνθεση, τη στάθμη καθώς και την κατάσταση κάθε δεξαμενής.
4. Καταγράφει στην βάση δεδομένων τη διαφορά πίεσης (Δp) καθώς και την ογκομετρική τροφοδοσία διυλιστηρίου.
5. Αναδιαμορφώνει τη βάση δεδομένων για συγκεκριμένο χρονικό διάστημα ύστερα από αίτηση των χειριστών.
6. Εξάγει αναφορά, ύστερα από αίτηση του χειριστή.
7. Παράγει ειδοποιήσεις (alerts) όπου κάτι τέτοιο κρίνεται απαραίτητο.

Σημείωση:

Γίνεται φανερό ότι οι λειτουργίες 3 και 4 που αναφέρθηκαν παραπάνω εξυπηρετούν στην τήρηση ιστορικού. Το σύστημα εγγράφει στη βάση δεδομένων τα στοιχεία που αναφέρονται στις λειτουργίες 3 και 4 ανά πέντε λεπτά για τις τελευταίες 2 ώρες, ενώ για χρονικό διάστημα παλαιότερο από αυτό κρατούνται ιστορικά στοιχεία ανά ώρα.

✓ Ευφυές υποσύστημα (Sophocles)

Ανακαλύπτει την τρέχουσα κατάσταση των δεξαμενών αργού στηριζόμενο στις προηγούμενες καταστάσεις τους, όπως αυτές υπάρχουν στο ιστορικό, και στα τρέχοντα δεδομένα.

Αναλαμβάνει επίσης να αποφανθεί αν η τρέχουσα σύνθεση τροφοδοσίας της αποστακτικής στήλης είναι παρόμοια με κάποια από αυτές που αναφέρονται ως «επικίνδυνες» (black listed).

3.4.2. Γνωρίζοντας τη διεπαφή χρήστη El Greco

Αποτελεί μια διεπαφή χρήστη που έχει ως στόχο αφενός την παρουσίαση με έναν κατάλληλο εποπτικά τρόπο δεδομένων στο χρήστη και αφετέρου λαμβάνει σαν είσοδο δεδομένα που θέλει να εισάγει κάποιος χειριστής.

Αναλυτικότερα οι λειτουργίες του αφορούν:

- ✓ Εποπτική αναπαράσταση του διυλιστηρίου.
- ✓ Δυνατότητα εισαγωγής του προγραμματισμού από τον χειριστή.
- ✓ Εμφάνιση επισημάνσεων (warnings & alerts).
- ✓ Εμφάνιση ημερολογίων (logs).
- ✓ Διαχείριση χρηστών του πληροφοριακού συστήματος (δημιουργία, επεξεργασία, διαγραφή).
- ✓ Χαρακτηρισμό των δεξαμενών αργού.
- ✓ Προσθαφαίρεση δεξαμενών

3.4.3. Προϋπάρχοντα συστήματα

Πρόκειται για συστήματα που προϋπήρχαν στις εγκαταστάσεις των Ελληνικών Πετρελαίων και βρίσκονται ήδη σε λειτουργία. Το παρόν έργο αντλεί πληροφορίες από αυτά αλλά δεν επηρεάζει περαιτέρω τη λειτουργία τους με κανέναν τρόπο.

✓ Oracle Server

Πρόκειται για ένα server που είναι φτιαγμένος για να φιλοξενεί μια Oracle 10g βάση δεδομένων και προσφέρει δικτυακά της υπηρεσίες της στους υπολογιστές – πελάτες του τοπικού δικτύου. Το παρόν πληροφοριακό σύστημα χρησιμοποιεί τον Oracle Server αφενός για να αντλεί πληροφορίες και αφετέρου δεσμεύει κάποιους πίνακές του για την τήρηση ιστορικού.

✓ OPC Server

Ο OPC Server συνδέεται απευθείας με το SAAB, ένα σύστημα που είναι επιφορτισμένο με τη μέτρηση της στάθμης των δεξαμενών καθώς και την κατάσταση στην οποία αυτές βρίσκονται. Το πληροφοριακό μας σύστημα χρησιμοποιεί τα παραπάνω δεδομένα για την εξαγωγή συμπερασμάτων και έλεγχο.

4. Υλοποίηση

Παρόλο που οι φάσεις της Σχεδίασης και Υλοποίησης εξετάζονται σε διαφορετικά κεφάλαια πρέπει να γίνει σαφές στον αναγνώστη ότι κατά την ανάπτυξη του πληροφοριακού συστήματος Arethusa εξελίσσονταν παράλληλα όπως προβλέπουν οι πρακτικές του Ακραίου Προγραμματισμού, «*Το παιχνίδι του σχεδιασμού*» και «*Μικρές παραδόσεις*». Επιλέχθηκε ωστόσο να παρουσιαστούν χωριστά στην τεκμηρίωση διότι, εν γένει, η Σχεδίαση και η Υλοποίηση συμμετέχουν με διαφορετικό τρόπο στην περάτωση του έργου.

Στο προηγούμενο κεφάλαιο καλύψαμε ζητήματα αρχιτεκτονικής σχεδίασης και είδαμε πόσες και ποιες λειτουργικές απαιτήσεις καλύφθηκαν από τις επιλογές μας αυτές. Στο παρόν κεφάλαιο θα παρακολουθήσουμε την υλοποίηση του έργου, σε επίπεδο κώδικα πια, καθώς αυτό περνούσε από διάφορες φάσεις και έπαιρνε την τελική του μορφή. Παράλληλα θα εξετάσουμε τα προβλήματα που εμφανίστηκαν κατά την υλοποίηση του εκάστοτε τμήματος του πληροφοριακού συστήματος και θα αναφέρουμε πως βοήθησε η χρήση των ευέλικτων μεθόδων στο να ξεπεραστούν, όπου αυτό κρίνεται απαραίτητο.

Εν γένει, η συνολική πορεία του έργου μπορεί να χωρισθεί σε πέντε διακριτές φάσεις υλοποίησης, τις οποίες σηματοδοτούν αντίστοιχες παραδόσεις. Αξίζει να σημειωθεί ότι η διάρκεια κάθε φάσης ήταν περίπου δύο μήνες. Ας τις δούμε αναλυτικά.

4.1. Φάση 1^η – Πυροδότηση του έργου

Μετά την αναλυτική καταγραφή του πρώτου σετ απαιτήσεων, η ομάδα ανάπτυξης άρχισε να δουλεύει σε ένα πρώτο σκαρίφημα του συστήματος ώστε να τις καλύπτει. Είχε ήδη γίνει φανερό ότι θα χρειαζόνταν να υλοποιηθούν δύο εφαρμογές εκ των οποίων η μία θα είχε ρόλο υπηρεσίας και η άλλη ρόλο διεπαφής χρήστη. Καθώς όμως η μελέτη και οργάνωση των ήδη καταγεγραμμένων απαιτήσεων προχωρούσε εμφανίστηκαν ασάφειες και σκοτεινά σημεία. Έτσι, παρόλο που ήταν προφανές ότι η υπηρεσία αποτελούσε την σημαντικότερη από τις δύο προς ανάπτυξη εφαρμογές, η ομάδα αποφάσισε να μην ξεκινήσει με την υλοποίηση αυτής προτού αποσαφηνισθούν όλες οι απαραίτητες λεπτομέρειες. Αντί αυτού, θέλοντας να εκμεταλλευτεί όλο το χρόνο που δέθηκε, προχώρησε στη δημιουργία ενός πρώιμου προτύπου διεπαφής χρήστη με σκοπό να διαμορφώσει μια πρώτη εντύπωση για το πώς την οραματίζονταν ο πελάτης βολιδοσκοπώντας τις αντιδράσεις του. Εκτός αυτού, έχει παρατηρηθεί ότι ο πελάτης δίνει πάντοτε περισσότερη σημασία σε κομμάτια της

εφαρμογής που θα κληθεί να χειριστεί και λιγότερη ή καθόλου σε αυτά που υπάρχουν και λειτουργούν αυτόνομα στο παρασκήνιο. Έτσι, με την παρουσίαση ενός προτύπου για το γραφικό περιβάλλον σαν πρώτο παραδοτέο, ο πελάτης αισθάνεται πως συνεισφέρει στην ανάπτυξη του έργου στο σύνολό του και γίνεται πιο δεκτικός και συνεργάσιμος.

4.1.1. Η γέννηση του γραφικού περιβάλλοντος

Αφότου λοιπόν επελέγη η παραπάνω στρατηγική, η ομάδα προχώρησε στη δημιουργία ενός προτύπου του κεντρικού παραθύρου. Η οθόνη χωρίζονταν σε τέσσερα πλαίσια (panels), όπως φαίνεται παρακάτω (Εικόνα 4-1).

Ο αριστερός χώρος, που καταλάμβανε περισσότερο από τα 2/3 του παραθύρου, καλούνταν να απεικονίσει τις οντότητες που συμμετέχουν στην διαδικασία της τροφοδοσίας. Χρησιμοποιήθηκαν εικονίδια για τον χαρακτηρισμό των δεξαμενών, του αγκυροβολίου και των εξόδων (αποστακτική στήλη και εξωτερική πηγή). Επίσης οι διάφορες ροές χαρακτηρίζονταν μονοσήμαντα από τον χρωματισμό τους. Τέλος, με γνώμονα τη real time φύση του έργου, προσφέρονταν ένα κουμπί ανανέωσης (refresh) στον χρήστη με τη χρήση του οποίου θα μπορούσε χειροκίνητα (χωρίς να είναι αναγκασμένος να περιμένει δηλαδή την αυτόματη ανανέωση) να ζητήσει απεικόνιση της τρέχουσας κατάστασης του διυλιστηρίου.

Δεξιά δημιουργήθηκαν τρία πλαίσια με στόχο να παρέχουν αναλυτικότερες πληροφορίες για τις δεξαμενές που απεικονίζονταν αριστερά. Σχολιάζοντας από πάνω προς τα κάτω το πρώτο πλαίσιο φιλοξενούσε βασικά χαρακτηριστικά της εκάστοτε δεξαμενής όπως την κατάσταση, το ποσοστό πληρότητας και άλλα. Ακριβώς από κάτω, στο δεύτερο πλαίσιο παρουσιάζονταν η σύνθεση του αργού που περιέχονταν στην δεξαμενή που απεικονίζονταν ενώ στο τελευταίο πλαίσιο παρουσιάζονταν επιπλέον σχόλια που σχετίζονταν με την εκάστοτε λειτουργία της δεξαμενής (για παράδειγμα στην εικόνα 4-1 η λειτουργία της επιλεγμένης δεξαμενής χαρακτηρίζεται ως τροφοδοτούσα. Στο τρίτο πλαίσιο λοιπόν εμφανίζεται η ποσοστιαία συμμετοχή της στην τροφοδοσία της αποστακτικής στήλης).

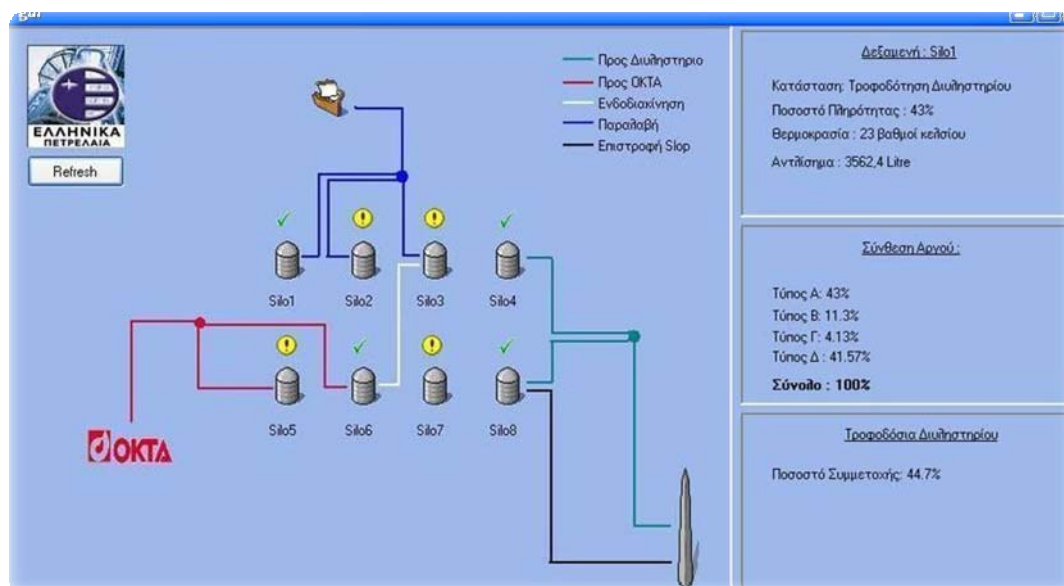
4.1.2. Αξιολόγηση της υλοποίησης κατά την πρώτη φάση

Η υλοποίηση των παραπάνω πραγματοποιήθηκε σε τρεις συνεδρίες και διήρκεσε, σε σύνολο επτά (7) ώρες. Δεν υπήρξαν σημαντικές κωλυσιεργίες και προβλήματα ενώ η συνεργασία της ομάδας ήταν αρμονική και έφθασε στο επιθυμητό αποτέλεσμα εγκαίρως (η

παρουσίαση του πρότυπου γραφικού περιβάλλοντος έγινε στις 19/01/2007 όπως και είχε προσυμφωνηθεί).

Αξίζει να σημειωθεί ότι το τελικό αποτέλεσμα δεν ικανοποίησε απόλυτα την ομάδα. Η ύπαρξη των στατικών πλαισίων, όπως αυτά περιγράφονται παραπάνω, πιθανότατα θα καθιστούσε το περιβάλλον μη ευέλικτο. Επιπλέον εκτός του κεντρικού παραθύρου θα έπρεπε να υλοποιηθούν και άλλα στο μέλλον. Θα ήταν λοιπόν καλό να υπάρχει ένας τρόπος ομαδοποίησής τους.

Η παρουσίαση του προτύπου ήρθε να επιβεβαιώσει τον σκεπτικισμό της ομάδας. Ο πελάτης εξέφρασε παρόμοιες ανησυχίες σχετικά με την ομαδοποίηση των παραθύρων και έθεσε σχεδιαστικές απαιτήσεις για τον τρόπο απεικόνισης των οντοτήτων του διυλιστηρίου που καλείται να παρακολουθεί η εφαρμογή.



Εικόνα 4-1. Διεπαφή χρήστη. Μια πρώτη προσέγγιση υπο μορφήν προτύπου

4.2. Φάση 2^η – Επικοινωνία με Oracle Server

Μετά την συνάντηση για την επίδειξη του πρότυπου γραφικού περιβάλλοντος οι απαιτήσεις για το σύνολο του έργου είχαν ωριμάσει αρκετά και οι ασάφειες που προβληματίζουν την ομάδα ανάπτυξης είχαν αποσαφηνισθεί, στο μεγαλύτερο τουλάχιστον μέρος τους. Πλέον ήταν ξεκάθαρο ότι η εφαρμογή – υπηρεσία θα αντλούσε και θα αποθήκευε δεδομένα κάνοντας χρήση δύο συστημάτων που προϋπήρχαν στις εγκαταστάσεις του πελάτη.

Το πρώτο αποτελούσε έναν server με ρόλο συστήματος διαχείρισης βάσεων δεδομένων (DBMS) ενώ το δεύτερο έναν server που ήταν επιφορτισμένος με την αλίευση στοιχείων για τις δεξαμενές κάνοντας χρήσης κάποιων αισθητήρων που βρίσκονται ενσωματωμένοι σε αυτές. Δεδομένου ότι η ομάδα αισθάνονταν περισσότερο εξοικειωμένη με το αντικείμενο των βάσεων δεδομένων αποφάσισε να ασχοληθεί πρώτα με την υλοποίηση του τμήματος της εφαρμογής που θα αναλάμβανε να συνδεθεί με τη βάση δεδομένων και να τη χρησιμοποιήσει για αποθήκευση και διαχείριση δεδομένων.

4.2.1. Oracle 10g – Μονόδρομος

Όπως φαίνεται και από τις μη λειτουργικές απαιτήσεις (βλ. §2.1.) η επιλογή του συστήματος που θα σχολιαστεί στην παρούσα παράγραφο όσο και όλων των άλλων συστημάτων που χρησιμοποιήθηκαν έγινε από τον ίδιο τον πελάτη (τμήμα Πληροφορικής των ΕΛ.ΠΕ., Β.Ε.Θ.). Έτσι η χρήση του Oracle 10g ως συστήματος διαχείρισης βάσεων δεδομένων (DBMS) για το έργο Arethusa αποτέλεσε μονόδρομο.

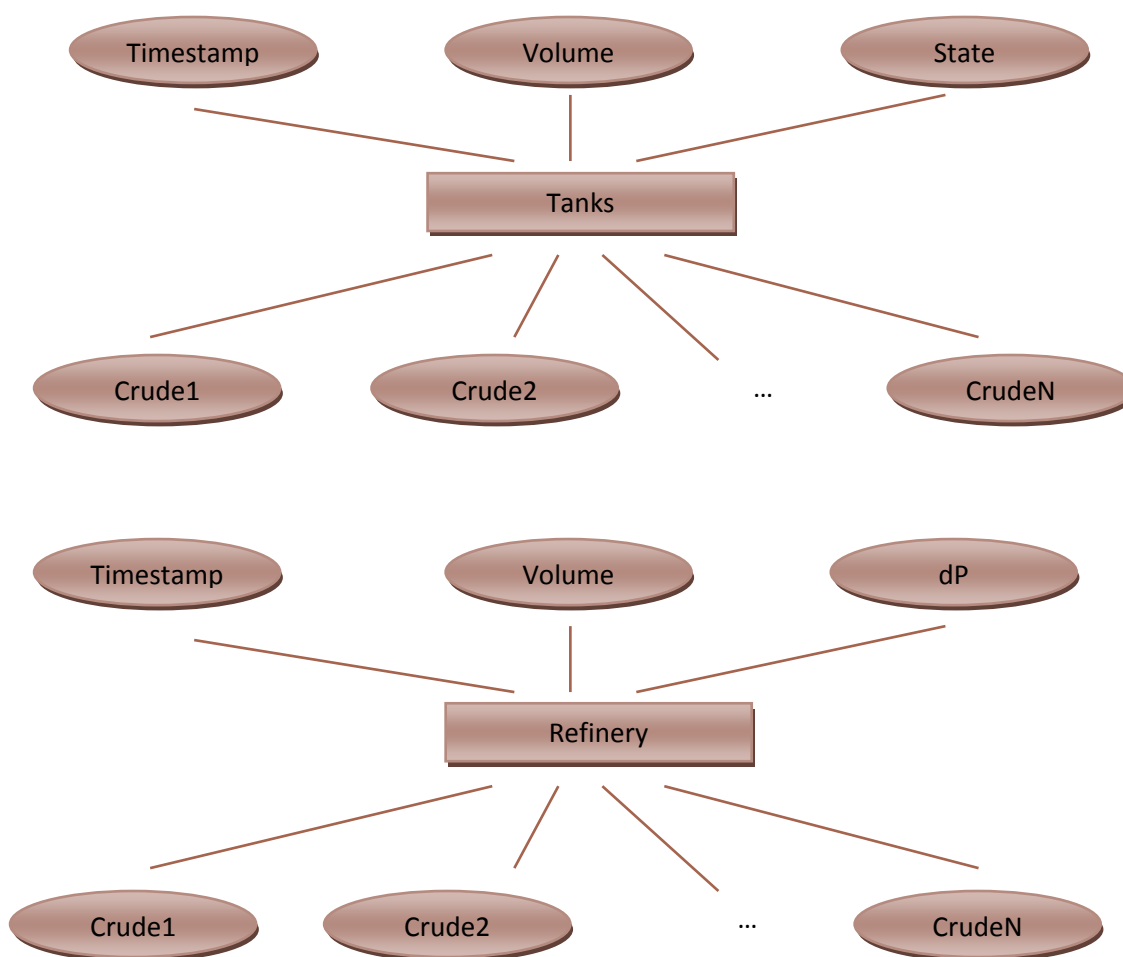
Αξίζει να σημειωθεί ότι η ομάδα ανάπτυξης χρησιμοποίησε μια δωρεάν έκδοση της προαναφερθείσας βάσης δεδομένων και συγκεκριμένα την Oracle 10g Express Edition. Αυτό δεν επηρέασε καθόλου την ανάπτυξη του έργου αφού οι δυο εκδόσεις Oracle που προαναφέρθηκαν δεν διέφεραν καθόλου στην αρχιτεκτονική. Διαφορές υπήρχαν αλλά κυρίως σε επίπεδο χρήσης (η δωρεάν έκδοση επιτρέπει μικρότερα ποσά αποθηκευμένης πληροφορίας, θέτει περιορισμό στον αριθμό χρηστών που μπορούν να τη χρησιμοποιούν ταυτόχρονα, κλπ.).

4.2.2. Σχεδιάζοντας με βάση τις απαιτήσεις

Η ύπαρξη της βάσης δεδομένων στο σύστημα Arethusa χρησιμοποιείται κυρίως για την τήρηση ιστορικού. Η real time φύση του συστήματος επιβάλλει να παρακολουθούνται και να καταγράφονται οι κινήσεις των δεξαμενών και της αποστακτικής στήλης στην πάροδο του χρόνου. Για τον σκοπό αυτό δημιουργήθηκαν δύο μοτίβα πινάκων ένα για τις δεξαμενές (tanks) με χαρακτηριστικά (πεδία) *timestamp* (στιγμιότυπο χρόνου), *volume* (όγκος), *state* (κατάσταση), *crude1*, *crude2*, *crude3*, ... , *crudeN*⁵ (αργό1, αργό2, ... , αργόN) και ένα για την

⁵ Το διυλιστήριο λειτουργεί με N συγκεκριμένους τύπους αργού πετρελαίου. Κάθε δεξαμενή μπορεί να περιέχει οποιοδήποτε συνδυασμό αυτών και σε οποιαδήποτε αναλογία. Η καταγραφή της συμμετοχής κάθε αργού συγκεκριμένου τύπου, για την εκάστοτε δεξαμενή μας επιτρέπει να γνωρίζουμε την ακριβή σύνθεσή της κάθε δεδομένη χρονική στιγμή.

αποστακτική στήλη (refinery), με πεδία *timestamp* (στιγμιότυπο χρόνου), *dp* (διαφορά πίεσης), *volume* (όγκο), *crude1*, *crude2*, *crude3*, ... ,*crudeN*⁶ (αργό1, αργό2, ... , αργόN). Η εφαρμογή, σύμφωνα με την τελική μορφή των απαιτήσεων, πρέπει να κρατά αρχεία ιστορικού σε ένα παράθυρο τριών (3) ετών. Αυτό σημαίνει ότι θα πρέπει να αποθηκεύουμε δεδομένα για το τρέχον έτος (καθώς αυτά διαμορφώνονται) και να κρατούμε ως ιστορικό όλη την πληροφορία που συλλέξαμε έως και τρία χρόνια πριν. Για τον σκοπό αυτό η Arethusa χρησιμοποιεί τέσσερις πίνακες για τις δεξαμενές και τέσσερις για την αποστακτική στήλη με ονόματα *tanks_YEAR*, *refinery_YEAR* αντίστοιχα. Όπου YEAR εμφανίζεται το έτος για το οποίο περιέχει δεδομένα ο αντίστοιχος πίνακας. Θα πρέπει να σημειωθεί ότι στους πίνακες του τρέχοντος έτους πρέπει να είναι δυνατή η χειροκίνητη διόρθωση εγγραφών.



Σχήμα 4-1. ER διαγράμματα για τους πίνακες tanks και refinery.

⁶ Ισχύει ότι και παραπάνω με τη διαφορά ότι σε αυτήν την περίπτωση δεν μιλούμε για μια απλή σύνθεση αργών πετρελαίων αλλά για την προς διύλιση σύνθεση (blend) που δόθηκε την χρονική στιγμή που δηλώνει το εκάστοτε timestamp.

4.2.3. Παρακολούθηση σε πραγματικό χρόνο

Μία από τις σημαντικότερες απαιτήσεις που καλείται να υλοποιήσει το έργο Arethusa είναι η παρακολούθηση του διυλιστηρίου σε πραγματικό χρόνο. Μετά από σειρά συναντήσεων με τον πελάτη αποφασίστηκε το σύστημα να συλλέγει πληροφορίες για τις διάφορες οντότητες που παρακολουθεί κάθε πέντε (5) λεπτά, να τις κρατά προσωρινά στη μνήμη και να τις αποθηκεύει στους πίνακες tanks και refinery για τον τρέχοντα χρόνο κάθε μια ώρα. Δηλαδή κάθε ώρα αποθηκεύονται σε κάθε έναν από τους δύο πίνακες δώδεκα (12) εγγραφές (εφόσον μια ώρα έχει δώδεκα πεντάλεπτα).

Το συγκεκριμένο μοτίβο παρακολούθησης και τήρησης ιστορικού κατέληξε να επικρατήσει μετά από πλήθος αλλαγών. Η αρχική προσέγγιση προέβλεπε συλλογή στοιχείων ανά τέταρτο της ώρας, αποθήκευση στη βάση μία φορά την ημέρα (δηλαδή κάθε 24 ώρες) και τήρηση ιστορικού ανά μήνα. Στη συνέχεια προτάθηκαν τα δέκα (10) λεπτά ως διάστημα συλλογής των δεδομένων, η καταγραφή στη βάση παρέμεινε ημερήσια ενώ η τήρηση ιστορικού άλλαξε σε ετήσια. Αργότερα προτάθηκε μια τρίτη προσέγγιση που μείωσε της συλλογή δεδομένων στα πέντε (5) λεπτά με ημερήσια καταγραφή στη βάση δεδομένων και τήρηση ιστορικού ανά έτος. Τελικά, αφότου μεσολάβησε μια γενικευμένη συνάντηση με τους περισσότερους από αυτούς που προορίζονταν να χρησιμοποιήσουν το σύστημα, οι συζητήσεις κατέληξαν στην τελική φόρμουλα όπως παρουσιάστηκε στην αρχή της παραγράφου.

Οι παραπάνω αλλαγές στις απαιτήσεις πιστοποιούν την αναποφασιστικότητα από μέρος του πελάτη που μπορεί να αντιμετωπίσει μια ομάδα ανάπτυξης ενός πληροφοριακού συστήματος. Αυτό εντείνει την ανάγκη για ανάπτυξη αρθρωτών συστημάτων και ορθογώνιων ως προς τον κώδικα. Παρακάτω παρέχονται σχόλια σχετικά με το πώς αντιμετωπίστηκε το πρόβλημα των συχνών αλλαγών στη συγκεκριμένη απαίτηση (της τήρησης ιστορικού) καθώς και πως βοήθησαν οι ευέλικτες μεθοδολογίες σε αυτό.

4.2.4. Άλλοι πίνακες

Εκτός των πινάκων παρακολούθησης και τήρησης ιστορικού γίνεται χρήση και κάποιων επιπλέον

- ✓ Ο πίνακας tanks_characterization με χαρακτηριστικά tank_id και characterization κρατά τον χαρακτηρισμό των δεξαμενών (Προς διύλωση ή Προς εξωτερική πηγή – βλ. §2.2.1).

- ✓ Ο πίνακας users με χαρακτηριστικά username και password ο οποίος περιλαμβάνει τους χρήστες που έχουν δικαίωμα πρόσβασης στο πληροφοριακό σύστημα (το χαρακτηριστικό username παίζει τον ρόλο του πρωτεύοντος κλειδιού).

4.2.5. Υλοποίηση και κώδικας

Η υλοποίηση της σύνδεσης και διαχείρισης του Oracle Server εμπεριέχεται στο project⁷ Oracle Handler που αποτελεί μέρος του solution⁸ Arethusa. Περιλαμβάνει την ομώνυμη κλάση OracleHandler που είναι γραμμένη σε γλώσσα J# (J sharp).

Η κλάση αυτή χρησιμοποιεί το OracleDataAccess component (.dll) για να πετύχει σύνδεση και διαχείριση της βάσης δεδομένων (λειτουργίες ανάγνωσης, εισαγωγής και ενημέρωσης δεδομένων). Για να μπορέσει να υλοποιήσει την ωριαία τήρηση αλλά και διόρθωση ιστορικού χρησιμοποιεί δύο δομές λεξικού (dictionaries). Έτσι οι προς αποθήκευση νέες εγγραφές που προκύπτουν αποθηκεύονται προσωρινά στο λεξικό insertions ενώ σε όσες εγγραφές πρέπει να επιβληθούν αλλαγές αποθηκεύονται στο λεξικό updates. Η δομή λεξικού περιλαμβάνει δύο πεδία, ένα τύπου String και ένα τύπου ArrayList. Στο πρώτο αποθηκεύεται ένα αναγνωριστικό (κλειδί) όπου στην προκειμένη περίπτωση μπορεί να πάρει τιμές Refinery (αν η εγγραφή αφορά την αποστακτική στήλη), Tank1 (αν η εγγραφή αφορά την πρώτη δεξαμενή), Tank2 (αν η εγγραφή αφορά τη δεύτερη), κ.ο.κ. Η εγγραφή περιέχει τα στοιχεία της αποστακτικής στήλης ή κάποιας δεξαμενής όπως παρουσιάζονται στην παράγραφο 4.2.2. όπου γίνεται περιγραφή της δομής των πινάκων tanks, refinery.

Θα προσπαθήσουμε τώρα να προσεγγίσουμε τη λειτουργικότητα της κλάσης μέσα από τον σχολιασμό των μεθόδων της.

connect ()

Προσπαθεί να συνδεθεί στη βάση δεδομένων του έργου. Σε περίπτωση επιτυχίας επιστρέφει TRUE ενώ σε αντίθετη περίπτωση τυπώνει το κατάλληλο μήνυμα λάθους και επιστρέφει FALSE.

disconnect ()

Αποσυνδέεται από τη βάση δεδομένων.

⁷ Ορολογία του περιβάλλοντος ανάπτυξης MS Visual Studio .NET. (βλ. Παράρτημα Β).

⁸ Ορολογία του περιβάλλοντος ανάπτυξης MS Visual Studio .NET. (βλ. Παράρτημα Β).

```
verifyUser(String username, String password)
```

Προσπαθεί να εντοπίσει κάποιων χρήστη στον πίνακα users με τα συγκεκριμένα credentials. Αν τα καταφέρει επιστρέφει TRUE ενώ σε αντίθετη περίπτωση FALSE.

```
insertRecord(String tableName, Object[] values)
```

Δημιουργεί το κατάλληλο PL-SQL query για εισαγωγή μιας έγγραφής στον πίνακα που προσδιορίζει η μεταβλητή tableName. Οι τιμές της έγγραφής προς εισαγωγή υπάρχουν στον πίνακα values.

```
insertAllPendingRecords()
```

Καταχωρεί τα περιεχόμενα του λεξικού insertions στην βάση δεδομένων κάνοντας χρήση της insertRecord. Ουσιαστικά υλοποιεί τη λειτουργία της Μαζικής Εισαγωγής (Mass Insertion).

```
updateRecord(String tableName, Object[] values)
```

Δημιουργεί το κατάλληλο PL-SQL query για αλλαγή των στοιχείων μιας έγγραφής στον πίνακα που προσδιορίζει η μεταβλητή tableName. Οι τιμές της έγγραφής προς αλλαγή υπάρχουν στον πίνακα values.

```
updateAllPendingRecords()
```

Αλλάζει τις έγγραφές της βάσης δεδομένων οι οποίες εμφανίζονται στο λεξικό updates κάνοντας χρήση της updateRecord. Ουσιαστικά υλοποιεί τη λειτουργία της Μαζικής Αλλαγής (Mass Update).

```
storePendingRecord(String tableName, Object[] values)
```

Εισάγει μια έγγραφή στο λεξικό Insertions. Χρησιμοποιείται από το πρόγραμμα για να στέλνει ανά πεντάλεπτο τα δεδομένα που πρόκειται να καταχωρηθούν στη βάση δεδομένων .

```
storeRecordAwaitingAlternation(String tableName, Object[] values)
```

Εισάγει μια έγγραφή στο λεξικό updates. Χρησιμοποιείται κατά τη χειροκίνητη διόρθωση στοιχείων από τους χειριστές του συστήματος .

4.2.6. Αξιολόγηση υλοποίησης κατά τη δεύτερη φάση

Η υλοποίηση των παραπάνω διήρκεσε τέσσερις (4) συνεδρίες. Παρά το «κακό lead in» από την παρουσίαση του προτύπου γραφικού, το οποίο θα έπρεπε να αλλάξει σχεδόν ριζικά, η ομάδα κατάφερε στις δύο πρώτες συνεδρίες να υλοποιήσει περισσότερο από το 70% του επιθυμητού αποτελέσματος. Η τρίτη συνεδρία χαρακτηρίστηκε από σχεδόν μηδενική παραγωγή για αυτό και διακόπηκε. Η τελευταία συνεδρία, που κάλυψε χρονικά τη διακοπή της προηγούμενης, σηματοδοτήθηκε από σωματική κόπωση και αδιαθεσία του Σκαλιστή Στέφανου. Παρόλα αυτά η ομάδα κατάφερε να λειτουργήσει με μεταφορά ορισμένου workload στον Κρητικό Απόστολο και στο τέλος της συνεδρίας η ομάδα κατάφερε να φθάσει στο επιθυμητό αποτέλεσμα. Η παρουσίαση του εν λόγω τμήματος έγινε στις 15/03/2007 με αφορμή μια συνάντηση για την παρουσίαση της γλώσσας κανόνων RuleML και της μηχανής NxBRE για ανάπτυξη συστημάτων κανόνων στο MS Visual studio .NET (βλ. §4.5).

Είναι σημαντικό στο σημείο αυτό να τονίσουμε ότι παράλληλα με την ανάπτυξη του τμήματος του έργου που εξετάσαμε στην δεύτερη φάση συνέβαιναν αλληπάλληλες αλλαγές απαιτήσεων σχετικά με τους χρόνους συλλογής δεδομένων παρακολούθησης και τήρησης ιστορικού. Το γεγονός ότι η ομάδα οργάνωσε την υλοποίηση κατά τέτοιον τρόπο ώστε να υλοποιούνται μικρές λειτουργίες, απαλλαγμένες από λεπτομέρειες όπως οι παραπάνω, της επέτρεψε να δουλέψει χωρίς να την επηρεάσουν καθόλου οι αλλαγές αυτές.

4.3. Φάση 3^η – Επικοινωνία με OPC Server

Σε αυτήν τη φάση η ομάδα ανάπτυξης κλήθηκε για πρώτη φορά να διαχειριστεί μια οντότητα για τη λειτουργία της οποίας είχε πλήρη άγνοια. Τόσο σε επίπεδο αρχιτεκτονικής όσο και σε επίπεδο φιλοσοφίας, η λειτουργία του συστήματος παρακολούθησης και ογκομέτρησης δεξαμενών (OPC Server) ήταν τελείως ξένη προς την ομάδα ανάπτυξης.

Στόχος της συγκεκριμένης φάσης ήταν η σύνδεση του έργου με το σύστημα ογκομέτρησης και η αλίευση μόνον της πληροφορίας εκείνης που ενδιέφερε το σύστημα Arethus. Για να μπορέσει να φθάσει στο ζητούμενο η ομάδα αποφάσισε να κινηθεί ως εξής. Αρχικά ξόδεψε λίγο χρόνο μιλώντας με ειδικούς της εταιρίας (πελάτης) ώστε να ανακαλύψει τι υπάρχει πίσω από το σύστημα μέτρησης του όγκου των δεξαμενών και πως το τελευταίο αντλεί πληροφορίες. Στη συνέχεια προσπάθησε να προσομοιώσει το σύστημα αυτό στον υπολογιστή που γίνονταν η ανάπτυξη του έργου Arethus ώστε να μπορεί να πειραματίζεται χωρίς να κινδυνεύουν τα αυθεντικά δεδομένα του πελάτη. Τέλος αφότου

κατακτήθηκε η γνώση των αρχών λειτουργίας όλων των παραπάνω η ομάδα στράφηκε αφενός προς την σύνδεση με το σύστημα μέτρησης όγκου και αφετέρου προς την αλίευση και διαχείριση των επιθυμητών δεδομένων από αυτό.

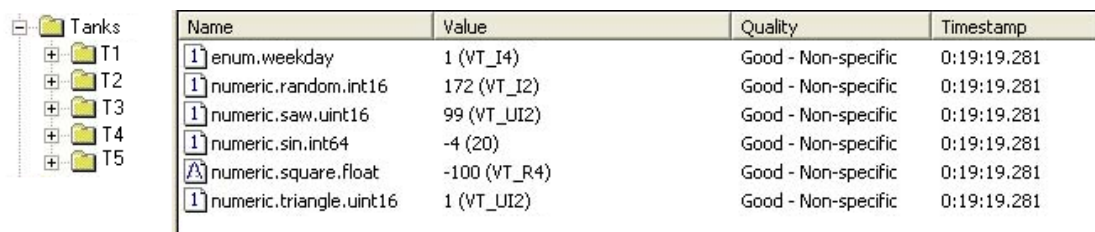
4.3.1. Σύστημα παρακολούθησης δεξαμενών

Θα δούμε σε λίγο πως πραγματοποιείται η παρακολούθηση των δεξαμενών μέσα από το πληροφοριακό σύστημα Arethusa. Θεωρούμε σκόπιμο ωστόσο, προηγουμένως, να περιγράψουμε συνοπτικά τον μηχανισμό που παρακολουθεί την υλική υπόσταση των δεξαμενών, δηλαδή τα ίδια τα σιλό. Η ενδελεχής ανάλυση αυτού του μηχανισμού ξεφεύγει από τα πλαίσια του παρόντος εγγράφου για αυτό η ανάλυση θα περιοριστεί σε μια υπεραπλουστευμένη περιγραφή της πραγματικότητας.

Κάθε δεξαμενή διαθέτει κάποιους αισθητήρες (sensors) οι οποίοι είναι σε θέση να παρέχουν διάφορες πληροφορίες όπως στάθμη, θερμοκρασία και άλλα. Οι αισθητήρες αυτοί είναι συνδεδεμένοι με ένα σύστημα το οποίο αφενός αναλαμβάνει να δέχεται τα σήματα των αισθητήρων και να τα μεταφράζει σε πληροφορία και αφετέρου να παρέχει τα δεδομένα του σε αναγνώσιμη μορφή. Στη συνέχεια μια εφαρμογή εξυπηρέτη (server-side application) αναλαμβάνει να παρέχει πρόσβαση σε όλους όσους έχουν δικαίωμα να δουν τη συγκεκριμένη πληροφορία μέσω μιας αντίστοιχης εφαρμογής πελάτη (client-side application).

4.3.2. OPC Server & OPC Clients

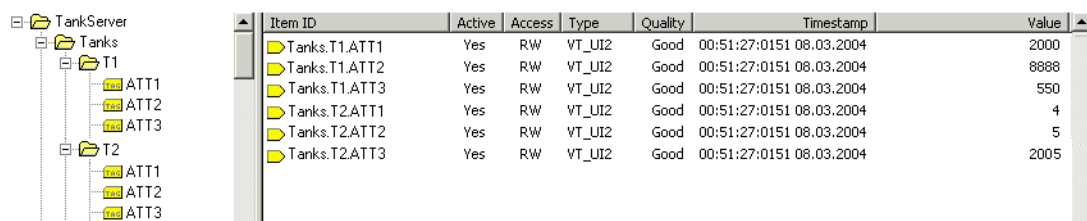
Ο OPC Server είναι η εφαρμογή εξυπηρέτη που καθιστά εφικτή την ανάγνωση των δεδομένων που εξήγαγαν οι αισθητήρες παρακολούθησης των δεξαμενών μέσω ενός OPC Client. Αυτό που ουσιαστικά κάνει ο OPC Server είναι να απεικονίζει τις οντότητες που παρακολουθεί και τα χαρακτηριστικά που γνωρίζει για την κάθε μια με μια δενδρική δομή.



Name	Value	Quality	Timestamp
enum.weekday	1 (VT_I4)	Good - Non-specific	0:19:19.281
numeric.random.int16	172 (VT_I2)	Good - Non-specific	0:19:19.281
numeric.saw.uint16	99 (VT_UI2)	Good - Non-specific	0:19:19.281
numeric.sin.int64	-4 (20)	Good - Non-specific	0:19:19.281
numeric.square.float	-100 (VT_R4)	Good - Non-specific	0:19:19.281
numeric.triangle.uint16	1 (VT_UI2)	Good - Non-specific	0:19:19.281

Εικόνα 4-2. Οι δεξαμενές και τα χαρακτηριστικά τους όπως απεικονίζονται στον Matricon OPC Server.

Στη συνέχεια οι χρήστες αποκτούν πρόσβαση στην πληροφορία που κρατά ο OPC Server χρησιμοποιώντας κάποιον OPC Client. Σε έναν τέτοιο client οι δεξαμενές απεικονίζονται και πάλι με δενδρική δομή ενώ τα χαρακτηριστικά τους απεικονίζονται σε δομή πίνακα.



Item ID	Active	Access	Type	Quality	Timestamp	Value
Tanks.T1.ATT1	Yes	RW	VT_UI2	Good	00:51:27:0151 08.03.2004	2000
Tanks.T1.ATT2	Yes	RW	VT_UI2	Good	00:51:27:0151 08.03.2004	8888
Tanks.T1.ATT3	Yes	RW	VT_UI2	Good	00:51:27:0151 08.03.2004	550
Tanks.T2.ATT1	Yes	RW	VT_UI2	Good	00:51:27:0151 08.03.2004	4
Tanks.T2.ATT2	Yes	RW	VT_UI2	Good	00:51:27:0151 08.03.2004	5
Tanks.T2.ATT3	Yes	RW	VT_UI2	Good	00:51:27:0151 08.03.2004	2005

Εικόνα 4-3. Οι δεξαμενές και τα χαρακτηριστικά τους όπως απεικονίζονται στον Matricon OPC Explorer.

Σημείωση: Πριν ξεκινήσει τη συγγραφή κώδικα η ομάδα εξοικειώθηκε με τις έννοιες των OPCServer και OPCClient κάνοντας χρήση των δωρεάν εργαλείων της εταιρίας Matricon, Matricon OPCServer και Matricon OPCExplorer. Ο Matricon OPCServer δε, χρησιμοποιήθηκε καθ' όλη τη διάρκεια της υλοποίησης του τμήματος του έργου που περιγράφεται στην παρούσα παράγραφο ως προσομοίωση του πραγματικού OPCServer που βρίσκεται στις εγκαταστάσεις των ΕΛ.ΠΕ. Β.Ε.Θ.

4.3.3. Υλοποίηση και κώδικας

Όπως γίνεται φανερό από τις προηγούμενες δύο παραγράφους η ομάδα ανάπτυξης κλήθηκε ουσιαστικά να υλοποιήσει ένα σύστημα που θα επιτύγχανε την σύνδεση με τον OPC Server και στη συνέχεια θα προσομοίωνε μέρος ενός OPC Client που θα επέτρεπε στο πληροφοριακό σύστημα Arethusa να αντλεί τα απαραίτητα δεδομένα.

Για το σκοπό αυτό δημιουργήθηκε ακόμη ένα project με όνομα OPCHandler κάτω από τη solution Arethusa. Περιλαμβάνει την ομώνυμη κλάση OPCHandler που είναι γραμμένη σε γλώσσα J# (J sharp).

Η κλάση αυτή χρησιμοποιεί το OPCAutomationWrapper component (.dll) για να πετύχει σύνδεση και διαχείριση του OPCServer (σε επίπεδο λειτουργιών, σύνδεση και αλίευση δεδομένων).

Για να μπορέσει η κλάση OPCHandler να συνδεθεί επιτυχώς στον OPCServer πρέπει να υλοποιηθεί ένας εικονικός OPCServer που θα λειτουργήσει ως γέφυρα μεταξύ του OPCHandler και του πραγματικού, απομακρυσμένου OPCServer. Το

OPCAutomationWrapper component μας παρέχει τη δυνατότητα αυτή με την κλάση OPCServerClass. Θα μπορούσαμε να παρομοιάσουμε τον εικονικό server με ένα ζευγάρι 3D γυαλιά. Ο θεατής φορά τα γυαλιά και μπορεί να δει την 3D πληροφορία (εικόνα) που του προσφέρει η ειδική αίθουσα. Ομοίως, ο εικονικός server τύπου OPCServerClass επιτρέπει στην Arethusa να δει τα δεδομένα που περιέχει ο απομακρυσμένος OPCServer.

Στη συνέχεια προσεγγίζουμε τη λειτουργικότητα της κλάσης μέσα από τον σχολιασμό των χαρακτηριστικών και μεθόδων της.

OPCServerClass myServer

Αντικείμενο τύπου OPCServerClass το οποίο μας δίνει πρόσβαση στα δεδομένα του απομακρυσμένου OPCServer (αποτελεί πεδίο – χαρακτηριστικό – για την κλάση OPCHandler).

connect()

Επιχειρεί σύνδεση με τον OPCServer (myServer) σαν πελάτης με όνομα Telegonous. Αν τα καταφέρει επιστρέφει TRUE ενώ σε διαφορετική περίπτωση FALSE.

disconnect()

Αποσυνδέεται από τον OPCServer (myServer) και αποδεσμεύει το αντικείμενο myServer.

initialize()

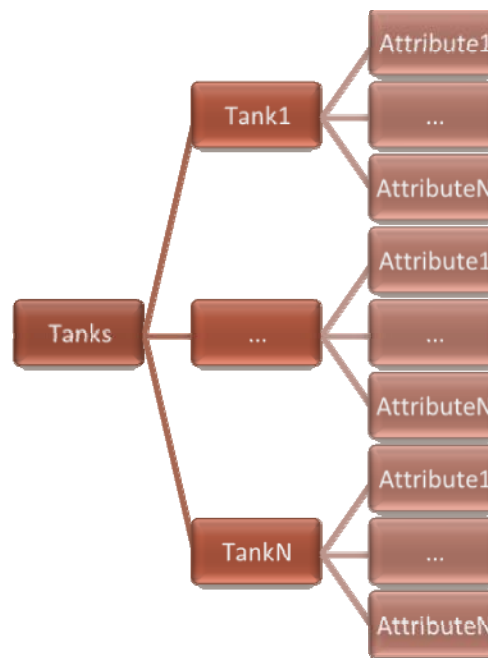
Αποτελεί τον ακρογωνιαίο λίθο του project. Χρησιμοποιεί τον εικονικό OPCServer myServer για να ζητήσει τα δεδομένα που μας ενδιαφέρουν από τον πραγματικό OPCServer. Αναλυτικότερα, η μέθοδος initialize δημιουργεί μια δενδρική δομή ίδια με αυτήν που περιέχει ο αυθεντικός OPCServer με τη διαφορά όμως ότι η δενδρική δομή του εικονικού Server περιλαμβάνει μόνον τις οντότητες και τα χαρακτηριστικά που τον ενδιαφέρουν (στη συγκεκριμένη περίπτωση που ενδιαφέρουν το πληροφοριακό σύστημα Arethusa) αγνοώντας τα υπόλοιπα. Το δένδρου που προκύπτει στην προκειμένη περίπτωση αποτελείται από τα εξής επίπεδα (από τη ρίζα προς τα φύλλα).

Επίπεδο 1 (ρίζα): Δημιουργία του **Group** δεξαμενών με όνομα Tanks.

Επίπεδο 2: Δημιουργία ενός **Group** για κάθε μία από τις N δεξαμενές με ονόματα Tank1, Tank2, ... , Tank N αντίστοιχα.

Επίπεδο 3 (φύλλα): Δημιουργία ενός **Item** για κάθε ένα χαρακτηριστικό δεξαμενής που μας ενδιαφέρει.

Σημείωση: Τα χαρακτηριστικά είναι δεδομένα και ίδια για όλες τις δεξαμενές ενώ δεν προβλέπεται τρόπος μεταβολής τους από το έργο.



Σχήμα 4-2. Προσέγγιση του περιεχομένου του myServer με δενδρική αναπαράσταση.

```
getCurrentData( )
```

Επιστρέφει τα ονόματα των δεξαμενών που βρίσκονται στο group Tanks.

```
readData(String tankName, String tag)
```

Επιστρέφει μια τριπλέτα τιμών του χαρακτηριστικού tag για τη δεξαμενή με όνομα tankName (ο κώδικας της μεθόδου παρατίθεται στο σύνολό του παρακάτω). Η τριπλέτα αυτή περιλαμβάνει την τιμή του συγκεκριμένου χαρακτηριστικού, μια εκτίμηση της ποιότητας της μέτρησης και ένα στιγμιότυπο χρόνου που προσδιορίζει πότε έγινε η μέτρηση. Η ποιότητα της μέτρησης έχει τη λογική του να προσδιορίζει πόσο αξιόπιστη είναι η πληροφορία που μεταφέρει η μέτρηση.

```

public static OPCAutomation.OPCItem readData(String tankName, String
tag)

{

OPCAutomation.OPCItem requestedItem =
myServer.get_OPCTGroups().GetOPCTGroup("Tanks").get_OPCTItems().Item("Ta
nks."+tankName + tag +".CV");

Object tmpValue = new Object();

Object tmpQuality = new Object();

Object tmpTimestamp = new Object();

requestedItem.Read((short)OPCAutomation.OPCDataSource.OPCDevice,
tmpValue, tmpQuality, tmpTimestamp);

return requestedItem;

}

```

Απόσπασμα κώδικα 4-1. Η μέθοδος readData.

4.3.4. Αξιολόγηση υλοποίησης κατά την τρίτη φάση

Η τρίτη φάση υλοποιήθηκε σε τέσσερις (4) συνεδρίες. Σημαντικό πρόβλημα αποτέλεσε η άγνοια της ομάδας γύρω από τη λειτουργία και διαχείριση τέτοιου είδους συστημάτων. Η ύπαρξη όμως εργαλείων προσομοίωσης καθώς και το πλούσιο υλικό του παγκόσμιου ιστού αποτέλεσαν οδηγό για μια επιτυχημένη πορεία. Στην πρώτη συνεδρία η ομάδα μελέτησε τον τρόπο λειτουργίας των εργαλείων προσομοίωσης και στο τέλος προσπάθησε να εγκαταστήσει το απαραίτητο component για να επιτευχθεί αργότερα η σύνδεση και διαχείριση. Η προσπάθεια στέφθηκε με επιτυχία. Η δεύτερη συνεδρία, που διακόπηκε απότομα, συμπλήρωσε τη δουλειά της προηγούμενης. Κατά τη διάρκειά της έγινε προσαρμογή των εργαλείων προσομοίωσης στα δεδομένα του διυλιστηρίου των ΕΛ.ΠΕ., Β.Ε.Θ. και έλεγχος ότι ο client της Matricon επικοινωνούσε ορθά με τον server και έβλεπε τα δεδομένα χωρίς πρόβλημα. Στις δύο τελευταίες συνεδρίες επιχειρήθηκε σύνδεση και ανάγνωση δεδομένων, μέσω του συστήματος Arethusα πια. Μετά από κάποια μικροπροβλήματα στην ανάγνωση των δεδομένων ο στόχος επετεύχθη με τη χρήση ενός προσωρινού γραφικού περιβάλλοντος το οποίο μάλιστα βελτιώθηκε και χρησιμοποιήθηκε στην παρουσίαση του τμήματος που συνδέονταν και διαχειρίζονταν των OPCServer και διεξήχθη στις 18/05/2007.

4.4. Φάση 4^η – Μετατροπή εφαρμογής σε υπηρεσία

Μετά το πέρας της τρίτης φάσης η ομάδα ανάπτυξης ήταν έτοιμη να δημιουργήσει ένα πρώτο παραδοτέο της εφαρμογής – υπηρεσίας. Στόχος της ομάδας ήταν η πρώτη αυτή εκδοχή της τελικής υπηρεσίας να εγκατασταθεί στο φυσικό της περιβάλλον (δηλαδή μέσα στις εγκαταστάσεις των Ελληνικών Πετρελαίων) ώστε αφενός να μπορεί να ελεγχθεί η σωστή επικοινωνία με τα συστήματα Oracle Server και OPCServer και αφετέρου εφόσον δεν υπάρξει κάποιο αναπάντεχο λάθος να καταγραφούν πραγματικά δεδομένα ώστε τα μελλοντικά τεστ να είναι πιο ρεαλιστικά ακόμη και σε περιβάλλον προσομοίωσης. Είχε επίσης προστεθεί η απαίτηση το παραδοτέο να αποτελεί μια διαδικτυακή Windows υπηρεσία.

Προτού ξεκινήσει η μετατροπή του ήδη υπάρχοντος έργου σε διαδικτυακή υπηρεσία έπρεπε να γίνει έλεγχος για το αν τα projects OracleHandler και OPCHandler μπορούσαν να συνεργαστούν αρμονικά. Έτσι δημιουργήθηκε ένα προσωρινό project με τίτλο Main (που αποτελούσε application και όχι service) το οποίο συνδέονταν στον OPCServer, αντλούσε δεδομένα ανά ένα fixed χρονικό διάστημα (της τάξεως των μερικών δευτερολέπτων) και μόλις συμπληρώνονταν ένα δεύτερο fixed χρονικό διάστημα (της τάξεως του λεπτού) προσπαθούσε να αποθηκεύσει τα δεδομένα στην βάση δεδομένων μέσω του OracleHandler. Κατά τη διάρκεια της άτυπης αυτής ενοποίησης προέκυψε η ανάγκη της αναδιαμόρφωσης του OPCHandler. Αυτό που ουσιαστικά χρειάστηκε να γίνει είναι να απομονωθεί ο κώδικας σύνδεσης και χειρισμού από το προσωρινό γραφικό περιβάλλον παρουσίασης με το οποίο μέχρι τότε συγκατοικούσε⁹. Μετά από αυτό η λειτουργία της πρόχειρης εφαρμογής στέφθηκε με απόλυτη επιτυχία.

4.4.1. Υλοποίηση και κώδικας

Για την μετατροπή της προσωρινής εφαρμογής σε δικτυακή υπηρεσία δημιουργήθηκε ακόμη ένα project, με όνομα Telegonous, γραμμένο σε J# που όμως αυτή τη φορά ήταν τύπου Windows Service και όχι application. Εντάχθηκε και αυτό στο γενικότερο solution Arethusa.

⁹ Επειδή κατά την τρίτη φάση το deadline παρουσίασης πίεσε χρονικά, η ομάδα ανάπτυξης αποφάσισε να περιτυλίξει την κλάση OPCHandler με ένα γραφικό αντί να το δημιουργήσει ξεχωριστά.

Από πλευράς χαρακτηριστικών μια Windows Service δεν διαφέρει πολύ από οποιαδήποτε άλλη application. Οι μέθοδοί της όμως παρουσιάζουν κάποιες ιδιαιτερότητες όπως θα δούμε παρακάτω.

Ο Telegonous σε αυτήν την φάση ήταν επιφορτισμένος με τη σύνδεση και λήψη δεδομένων (για κάθε δεξαμενή) από τον OPC Server (μέσω του OPCHandler) ανά πεντάλεπτο, την προσωρινή αποθήκευσή τους και τελικά την καταχώρησή τους στη βάση δεδομένων (μέσω του OracleHandler) ανά ώρα. Δύο χρονόμετρα με ονομασίες viciousTimeCycle και dbTimer κτυπούν ανά πεντάλεπτο και ανά μία (1) ώρα αντίστοιχα. Επιπλέον, ορίζεται μια δομή λεξικού (dictionary) με όνομα tanks το οποίο περιέχει πληροφορίες για όλες τις δεξαμενές που παρακολουθούμε. Τέλος, όπως συμβαίνει με τις περισσότερες υπηρεσίες, έτσι και ο Telegonous περιλαμβάνει ένα ημερολόγιο κινήσεων (EventLog) με όνομα TelegonousEventLog το οποίο καταγράφει σημαντικές κινήσεις κατά τη λειτουργία της υπηρεσίας. Το EventLog χρησιμοποιήθηκε κατά κόρον και κατά την διαδικασία ανάπτυξης αφού μέσω αυτού η υπηρεσία άφηνε ίχνη της λειτουργίας της όπως «Σύνδεση με τον OPCServer», «Σύνδεση με τον OracleServer».

Όσον αφορά την ιδιαιτερότητα των Windows υπηρεσιών ως προς τις μεθόδους, επιβάλλεται η ύπαρξη των OnStart() και OnStop(). Αυτές περιγράφουν τις ενέργειες που θα κάνει η υπηρεσία κατά την εκκίνηση και διακοπή της λειτουργίας της. Από εκεί και πέρα ο προγραμματιστής είναι ελεύθερος να προσθέσει όσες ακόμη μεθόδους θέλει κατά τα γνωστά.

Στη συνέχεια προσεγγίζουμε τη λειτουργικότητα της κλάσης μέσα από τον σχολιασμό των χαρακτηριστικών και μεθόδων της.

OnStart(String[] args)

Αρχικοποιεί τα χρονόμετρα viciousTimeCycle και dbTimer ορίζοντας τους χρόνους απόκρισής τους σε 300.000 και 3.600.000 milliseconds αντίστοιχα και προσθέτει στο λεξικό tanks πληροφορίες για κάθε δεξαμενή με την παρακολούθηση της οποίας είναι επιφορτισμένο το σύστημα. Επιπλέον αποτυπώνει στο EventLog το χρονικό στιγμιότυπο έναρξής του και επιχειρεί σύνδεση με τους Oracle και OPCServer. Αν όλα πάνε καλά αποσυνδέεται από τα δύο υποσυστήματα αλλιώς καταγράφει στο EventLog ποια-ες σύνδεση-εις απέτυχαν.

OnStop()

Καταγράφεται στο EventLog το χρονικό στιγμιότυπο κατά το οποίο σταμάτησε να λειτουργεί η υπηρεσία και απενεργοποιούνται τα δύο χρονόμετρα.

OnViciousCycleTimerTick(Object source, ElapsedEventArgs e)

Οι λειτουργίες του εκτελούνται κάθε 300.000 milliseconds (5 λεπτά). Συνδέεται στον OPCServer, για κάθε δεξαμενή συλλέγει τις κατάλληλες πληροφορίες και αποσυνδέεται. Στη συνέχεια, με τις πληροφορίες που συνέλλεξε, δημιουργεί μια εγγραφή και την αποθηκεύει προσωρινά στη μνήμη κάνοντας χρήση της μεθόδου storePendingRecord του OracleHandler. Κατόπιν αποσυνδέεται από την Oracle.

OnDbTimerTick(Object source, ElapsedEventArgs e)

Η λειτουργία του εκτελείται κάθε 3.600.000 milliseconds (1 ώρα). Συνδέεται στην Oracle και εισάγει όλες τις εγγραφές, που κρατούνται μέχρι εκείνη τη στιγμή στη μνήμη, στους πίνακες του τρέχοντος έτους, κάνοντας χρήση της μεθόδου insertAllPendingRecords και αποσυνδέεται από τη βάση δεδομένων.

writeToEventLog(String message)

Καταγράφει το περιεχόμενο της μεταβλητής message ως μήνυμα στο EventLog.

Κρίνουμε σκόπιμο σε αυτό το σημείο να αναφέρουμε ότι η δημιουργία της κλάσης μιας Windows υπηρεσίας συνοδεύεται από τη δημιουργία μιας κλάσης που παίζει το ρόλο του Installer της υπηρεσίας. Μια τέτοια κλάση, αν και δεν παράγεται αυτόματα από το σύστημα περιορίζεται στην διαχείριση έτοιμων εργαλείων που διατίθενται σε περιβάλλον σχεδίασης από το Visual Studio .NET. Επομένως, εφόσον δεν υπάρχει επέμβαση στον κώδικα της κλάσης αυτής, η περιγραφή της περιορίζεται στην τρέχουσα αναφορά (περιλαμβάνεται ωστόσο στο διάγραμμα κλάσεων που παρέχεται στο τέλος του κεφαλαίου).

4.4.2. Αξιολόγηση υλοποίησης κατά την τρίτη φάση

Σε αυτή τη φάση αναδείχθηκε η ομαδικότητα των συνεργατών περισσότερο από ότι σε οποιαδήποτε άλλη. Αρχικά, μελέτη σχετικά με την υλοποίηση Windows υπηρεσιών είχε πραγματοποιηθεί από τον Στέφανο Σκαλιστή ο οποίος και μεταβίβασε τη γνώση που είχε στον Κρητικό Απόστολο. Στη συνέχεια δούλεψαν μαζί στην πρώτη και δεύτερη συνεδρία

αλλά δεν κατάφεραν να φθάσουν σε ένα λειτουργικό αποτέλεσμα. Στην τρίτη συνεδρία ο Κρητικός Απόστολος έφθασε έχοντας κάνει τη δική του έρευνα πάνω στην υλοποίηση Windows υπηρεσιών, πατώντας πάνω στις πληροφορίες που είχε από τη συζήτηση με τον Σκαλιστή Στέφανο. Η έρευνα αυτή απέδωσε και οδήγησε σε κατανόηση των λαθών στην μέχρι τότε λογική. Αφού κοινοποιήθηκαν τα συμπεράσματά του Κρητικού Απόστολος και στον Σκαλιστή Στέφανο, στην τέταρτη συνεδρία η υπηρεσία είχε πετύχει τον αρχικό της στόχο. Η παράδοση της πρώτης αυτής υλοποίησης της δικτυακής υπηρεσίας έγινε στις 01/09/2007.

4.5. Φάση 5^η – Ανάπτυξη ευφυούς συστήματος και ενσωμάτωση

Από την δεύτερη κιόλας φάση υλοποίησης είχαν αρχίσει συζητήσεις μεταξύ της ομάδας ανάπτυξης και του πελάτη για την ενσωμάτωση ενός ευφυούς υποσυστήματος το οποίο θα ήταν επιφορτισμένο με τον προσδιορισμό της τρέχουσας κατάστασης κάθε δεξαμενής στηριζόμενο σε κάποιους κανόνες που διέπουν την λειτουργία του διυλιστηρίου αλλά και στην κατάσταση στην οποία βρίσκονταν η εκάστοτε δεξαμενή τις προηγούμενες χρονικές στιγμές.

Οι συζητήσεις αυτές κατέληξαν σε συγκεκριμένες απαιτήσεις λίγο πριν το τέλος της τρίτης φάσης. Η ομάδα ανάπτυξης βέβαια είχε αρχίσει πολύ πιο πριν να αναζητά τον τρόπο με τον οποίο θα σχεδίαζε το ευφυές σύστημα. Όταν λοιπόν οι απαιτήσεις του τελευταίου έγιναν κάπως πιο συγκεκριμένες είχε ήδη συμφωνηθεί από τα δύο μέλη της ομάδας ανάπτυξης ότι θα χρησιμοποιούνταν η γλώσσα κανόνων RuleML για τη σύνταξη κανόνων και η μηχανή χειρισμού NxBRE που συνεργάζονταν χωρίς προβλήματα με το περιβάλλον ανάπτυξης Visual Studio .NET 2005(είχαν ήδη πραγματοποιηθεί κάποιες πρώτες – γενικής φύσεως – δοκιμές).

Σε αυτό το σημείο (όπως τονίζεται και στο κεφάλαιο της σχεδίασης) υπήρξε ανάγκη αλλαγής στην αρχιτεκτονική του συστήματος διότι προστέθηκε ένα ακόμη υποσύστημα, το ευφυές υποσύστημα με όνομα Sophocles.

4.5.1. Γεγονότα

Όπως έγινε φανερό και από την εισαγωγή το ευφυές σύστημα προσεγγίστηκε από ένα σύστημα κανόνων. Είναι κοινά αποδεκτό ότι για να δουλέψει σωστά και αποδοτικά το οποιοδήποτε σύστημα κανόνων πρέπει να περιέχει κανόνες που να διέπονται από μια

πλήρη και σωστή δομή. Έτσι η ομάδα ανάπτυξης φρόντισε από την αρχή, πριν ακόμη αρχίσει να μοντελοποιεί τον οποιοδήποτε κανόνα, να καθορίσει σαφώς τα αρχικά γεγονότα που θα περιέγραφαν το τμήμα του διυλιστηρίου με το οποίο ασχολείται η εφαρμογή Arethusa. Τα γεγονότα αυτά είναι:

- ✓ `tanks(Id, Characterization, DV, SaabIsON)` όπου η μεταβλητή:
 - *Id* περιγράφει το αναγνωριστικό της δεξαμενής (π.χ. TK1).
 - *Characterization* περιγράφει τον χαρακτηρισμό της δεξαμενής. Μπορεί να πάρει τις τιμές { Προς διύλιση, Προς εξωτερική πηγή }.
 - *DV* είναι η διαφορά όγκου από την προηγούμενη μέτρηση (θετική αν γεμίζει, αρνητική αν αδειάζει).
 - *SaabIsON* περιγράφει αν ο αισθητήρας του συστήματος που μετρά τα χαρακτηριστικά της δεξαμενής είναι σε λειτουργία. Παίρνει την τιμή 1 αν αυτό αληθεύει και την τιμή 0 σε αντίθετη περίπτωση.
- ✓ `previous_state(TankId, HowOld, State)` όπου η μεταβλητή:
 - *TankId* περιγράφει το αναγνωριστικό της δεξαμενής (π.χ. TK1)
 - *HowOld* περιγράφει το πόσο παλαιά είναι η κατάσταση. Παίρνει την τιμή 1 αν η κατάσταση είναι η προηγούμενη (δηλ. η κατάσταση στο προηγούμενο πεντάλεπτο), 2 αν είναι η προ-προηγούμενη κ.ο.κ.
 - *State* περιγράφει την κατάσταση της δεξαμενής.
- ✓ `current_state(TankId, State)` όπου η μεταβλητή:
 - *TankId* περιγράφει το αναγνωριστικό της δεξαμενής (π.χ. TK1)
 - *State* περιγράφει την κατάσταση της δεξαμενής.
- ✓ `refinery(CubicMetresFedded, SLOPReturn)` όπου η μεταβλητή:
 1. *CubicMetresFedded* είναι να κυβικά μέτρα τροφοδοσίας της αποστακτικής στήλης.

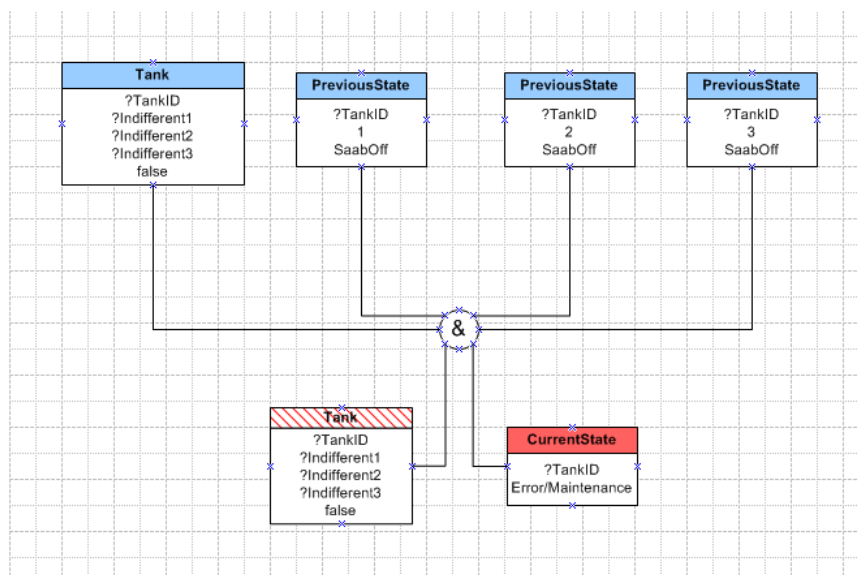
Σημείωση: Τα γεγονότα εκφράζονται ως Κεφαλή(Σώμα).

4.5.2. Κανόνες

Οι παρακάτω κανόνες ενεργοποιούνται με τη σειρά και έτσι χαρακτηρίζεται η τρέχουσα κατάσταση κάθε δεξαμενής (για κάθε κανόνα που περιγράφουμε παραθέτουμε και την αντίστοιχη απεικόνισή του σε RuleML, οπτικοποιημένη με χρήση του Microsoft Visio):

✓ Σφάλμα – Συντήρηση (Error / Maintenance):

Μια δεξαμενή βρίσκεται σε αυτήν την κατάσταση αν το Saab είναι Off την τρέχουσα και όλες τις προηγούμενες στιγμές (PreviousState 1, PreviousState 2, PreviousState 3).

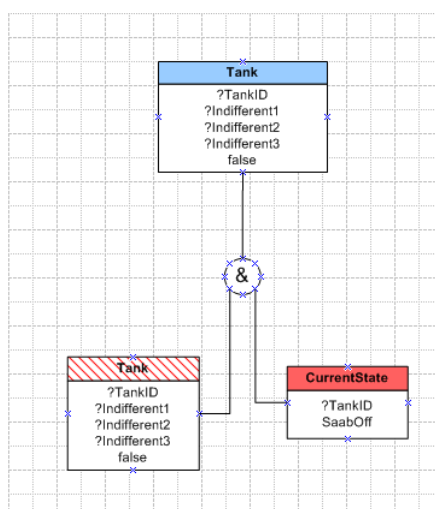


Εικόνα 4-4. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Error / Maintenance»

✓ Ο αισθητήρας είναι εκτός λειτουργίας (Saab Is Off):

Μια δεξαμενή βρίσκεται σε αυτήν την κατάσταση αν το Saab είναι Off την τρέχουσα χρονική στιγμή ενώ κάποια από τις προηγούμενες δεν ήταν Off

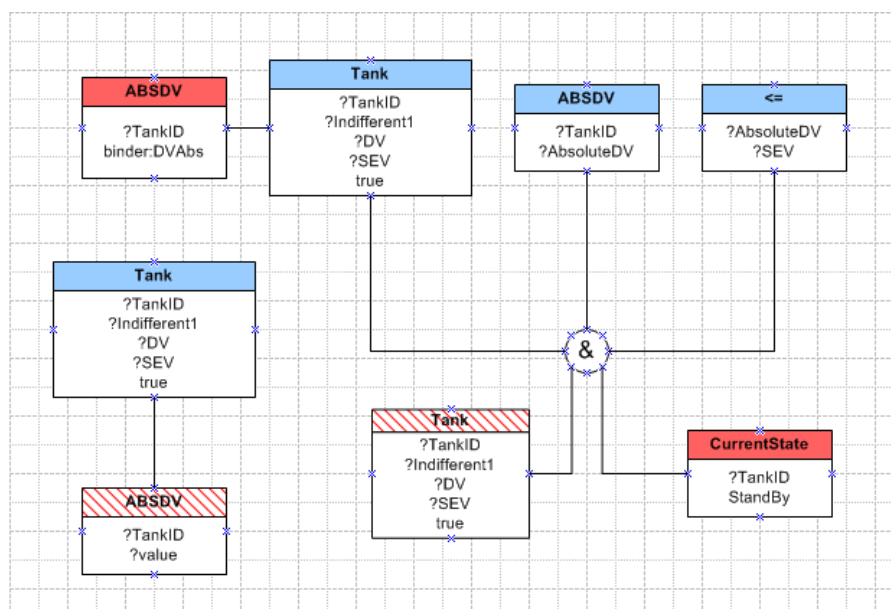
Σημείωση: σε αυτήν την περίπτωση η δεξαμενή θα χαρακτηριστεί SaabOff αλλά θα μεταχειρίζεται ως είχε.



Εικόνα 4-5. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Saab Is Off»

✓ Σε αναμονή (Stand By):

Μια δεξαμενή βρίσκεται σε αυτήν την κατάσταση αν το Saab της είναι On και η μεταβολή όγκου (DV) είναι μικρότερη κατ' απόλυτη τιμή από το επιτρεπόμενο διάστημα μεταβολής όγκου (SEV).

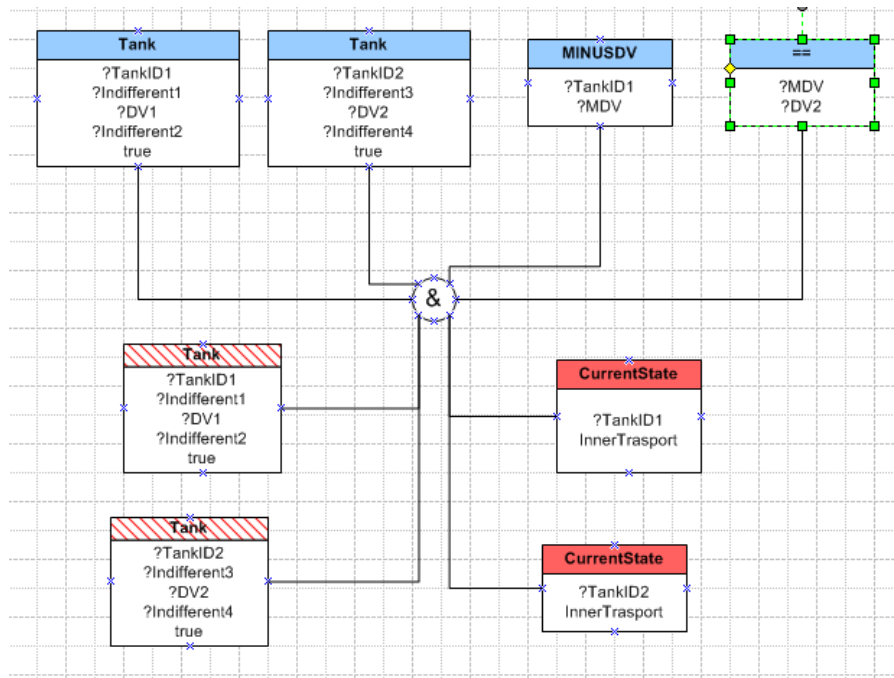


Εικόνα 4-6. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Stand By»

✓ Ενδοδιακίνηση (Inner Transport):

Αν το Saab είναι On σε 2 δεξαμενές και αυτές έχουν μεταβολή όγκου ίση και ετερόσημη.

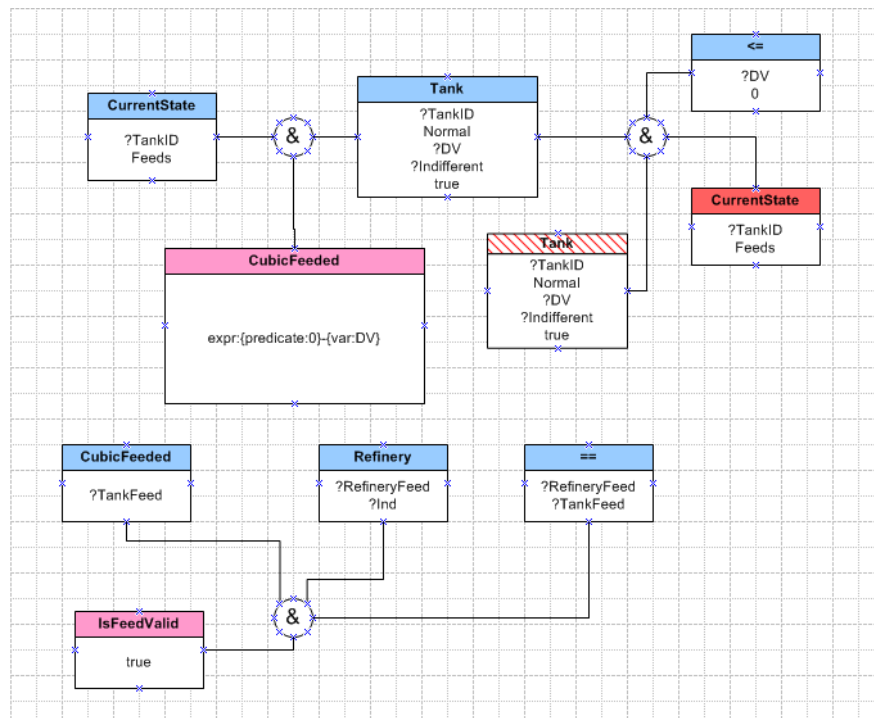
Σημείωση: Ακολουθείται η παραδοχή ότι μόνον μία δεξαμενή μπορεί να ενδοδιακινεί ως προς μία κάθε στιγμή.



Εικόνα 4-7. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Inner Transport»

✓ Τροφοδοσία διυλιστηρίου (Refinery Feed):

Μια δεξαμενή βρίσκεται σε αυτήν την κατάσταση αν το Saab είναι On και έχει χαρακτηριστεί ως normal .



Εικόνα 4-8. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Refinery Feed»

✓ Παραλαβή από αγκυροβόλιο (Receiving):

Μια δεξαμενή βρίσκεται σε αυτήν την κατάσταση αν το Saab είναι On και παραλαμβάνει με ρυθμό ανάλογο με αυτόν παραλαβής από καράβι. Σε αυτήν την περίπτωση ψάχνουμε ενδεχομένως για μια ακόμη δεξαμενή που παραλαμβάνει με ρυθμό διαφορετικό από τον ρυθμό παραλαβής slop. Με αυτόν τον τρόπο καλύπτουμε την πιθανότητα ύπαρξης δεξαμενής που λειτουργεί ως ενδιάμεση (buffer). Αν υπάρχει και άλλη δεξαμενή που παραλαμβάνει με ρυθμό ανάλογο με αυτόν παραλαβής slop τότε τη χαρακτηρίζουμε ως δεξαμενή παραλαβής slop.

Προσοχή: Κάθε φορά που εκτελείται ένας κανόνας, όσα από τα γεγονότα τον ικανοποιούν, όσες δηλαδή δεξαμενές χαρακτηρίζονται από αυτόν, διαγράφονται από τη μνήμη. Έτσι μετά το πέρας όλων των κανόνων θα έχουν χαρακτηριστεί όλες οι δεξαμενές.

4.5.3. RuleML

Η Rule Markup Language (RuleML) αποτελεί μια γλώσσα περιγραφής κανόνων για χρήση από ευφυή συστήματα και συστήματα κανόνων. Αποτελεί προϊόν της σύμπραξης Rule Markup Initiative και ορίστηκε για να δώσει λύση στη μη ύπαρξη μιας κοινά αποδεκτής γλώσσας κανόνων.

Τα πλεονεκτήματα της RuleML είναι πολλά με σημαντικότερα τα παρακάτω:

- ✓ Η χρήση της XML είναι ευρέως διαδεδομένης. Επομένως η RuleML που ακολουθεί πολύ παρόμοια δομή χαίρει της ίδιας αποδοχής.
- ✓ Είναι αποδεκτή τόσο από επιχειρήσεις όσο και από ακαδημαϊκά ιδρύματα (ενδεικτικά αναφέρονται οι IBM, SUN, MIT).
- ✓ Παρέχει κατηγοριοποίηση κανόνων.
- ✓ Αναμένεται να προταθεί σύντομα η προσάρτησή της ως standard στο W3C (World Wide Web Consortium).

4.5.4. NxBRE και Microsoft Visio

Η NxBRE είναι μία ελαφριά Business Rules Engine μηχανή κανόνων. Με τον όρο Business Rules αναφερόμαστε στις περιγραφές των ορισμών, των περιορισμών και των ενεργειών που πρέπει να γίνουν ώστε να επιτευχθούν κάποιοι προκαθορισμένοι στόχοι. Η NxBRE είναι

η πρώτη μηχανή κανόνων ανοικτού κώδικα και διανέμεται υπό την LGPL άδεια χρήσης. Το όνομα της προέρχεται από το ".NET Business Rules Engine".

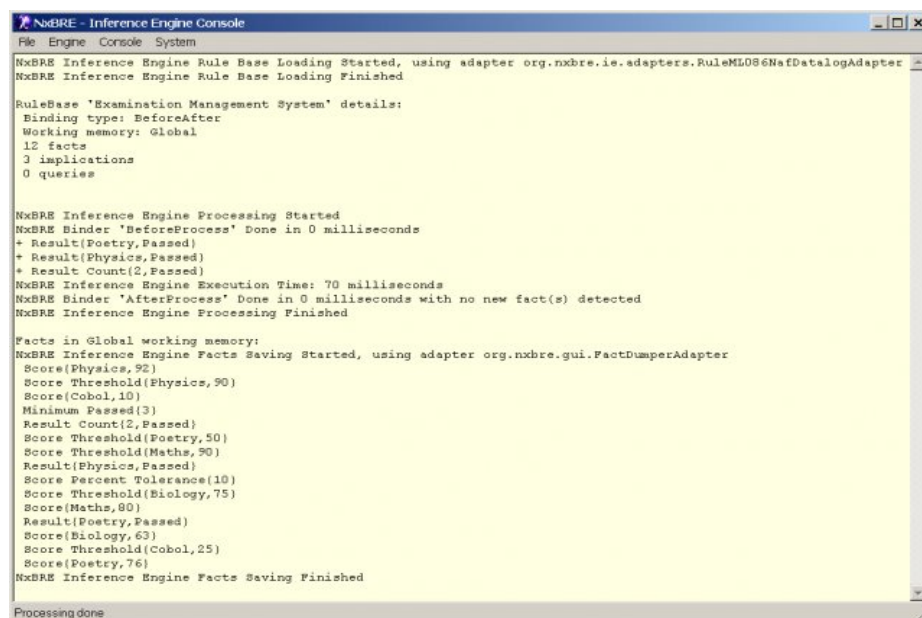
Από άποψη τεχνικών χαρακτηριστικών ενδιαφέρον παρουσιάζουν τα παρακάτω:

- ✓ Είναι υλοποιημένη σε C# και προσφέρεται σε μορφή *dll*.
- ✓ Αποτελεί μετανάστευση της Java Business Rule Engine.
- ✓ Υποστηρίζει τη γλώσσα κανόνων RuleML.
- ✓ Είναι ανοιχτό λογισμικό (άδεια LGPL).

Τα παραπάνω τεχνικά χαρακτηριστικά κάνουν την NxBRE να πλεονεκτεί γιατί:

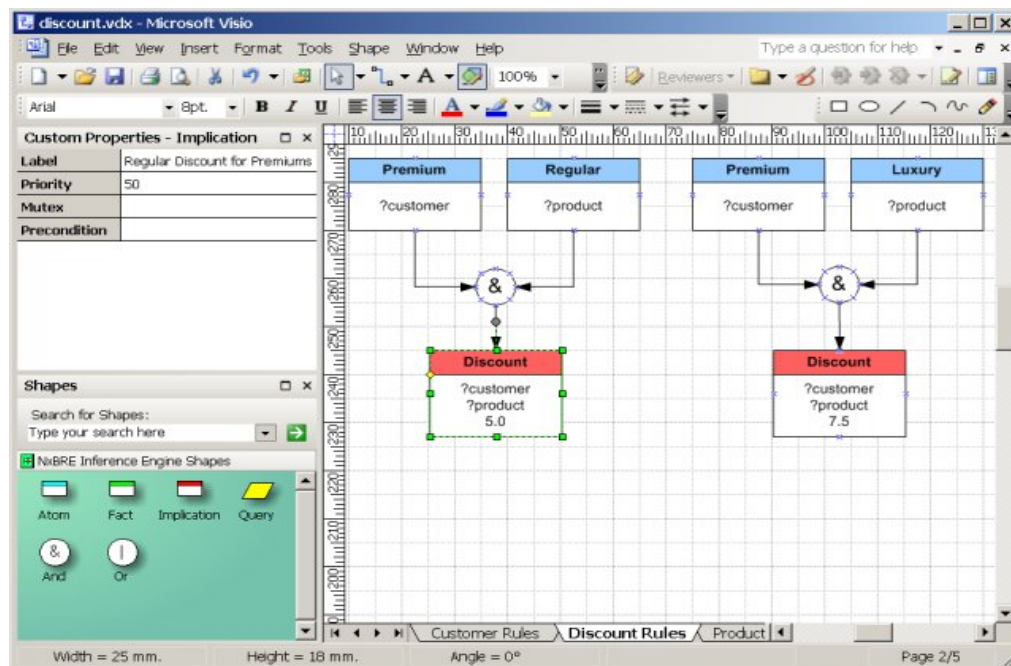
- ✓ Είναι αρκετά "ώριμη" ηλικιακά ώστε και άρα μπορεί πλέον να θεωρηθεί ασφαλής.
- ✓ Είναι μετανάστευση ήδη υπάρχουσας αξιόπιστης μηχανής επομένως είναι απαλλαγμένη από τους κινδύνους του καινούριου.
- ✓ Διανέμεται με LGPL άδεια χρήσης πράγμα που επιτρέπει τη συνεχή ανάπτυξη και διόρθωσή της.

Σημείωση: Η NxBRE συνοδεύεται από ένα *interface* μέσω του οποίου μπορεί να γίνει διαχείριση σε πρωτογενές επίπεδο.



Εικόνα 4-9. Παρακολούθηση των γεγονότων και εκτέλεση ερωτημάτων-κανόνων μέσω του Interface

Η μηχανή NxBRE είναι σε θέση να αναγνωρίσει κανόνες και ερωτήματα τα οποία έχουν σχεδιαστεί μέσω του εργαλείου Microsoft Visio. Η χρήση αυτού του σχεδιαστικού προγράμματος βοηθάει ιδιαίτερα στο σωστό σχεδιασμό και την αποφυγή σχεδιαστικών και άλλων λαθών αφού προσφέρει εποπτική αναπαράσταση όλων όσων χρειάζονται. Μαζί με την NxBRE παρέχεται ένα stencil (NxBRE-IE-3.vsx) το οποίο περιέχει τα εξής που εμπλουτίζουν τις δυνατότητες του MS Visio.



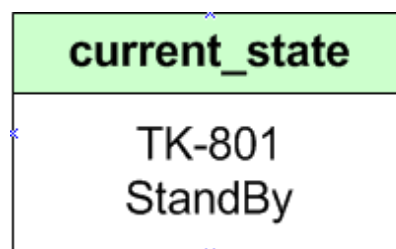
Εικόνα 4-10. Ένα παράδειγμα κανόνα σχεδιασμένου σε Microsoft Visio

- ✓ Οι δομικές οντότητες που περιλαμβάνει το NxBRE-IE-3.vsx και εμπλουτίζουν το περιβάλλον του Microsoft Visio είναι οι εξής:
 - Atom
 - Negative Atom
 - Fact
 - Implication
 - Negative Implication
 - Modifying Implication
 - Counting Implication
- ✓ Αντίστοιχα παρέχονται οι ακόλουθοι τελεστές:
 - And
 - Or
 - Equivalent

Για όσους είναι εξοικειωμένοι με τις έννοιες των κανόνων, συστημάτων κανόνων και συστημάτων γνώσης τα παραπάνω είναι σχετικά προφανή. Για λόγους όμως πληρότητας θα αναφερθούμε αναλυτικότερα.

Fact

Εκφράζει ένα γεγονός ή μία αλήθεια για το σύστημα μας. Παραδείγματος χάριν `current_state(TK-801, StandBy)` εκφράζει ότι η τωρινή κατάσταση της δεξαμενής TK-801 είναι StandBy.

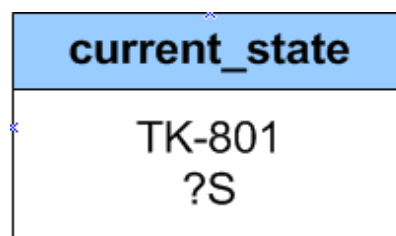


Εικόνα 4-11. Δήλωση ενός fact σε περιβάλλον MS Visio.

Προσοχή: Ένα γεγονός δεν μπορεί να έχει μεταβλητά μέρη.

Atom

Εκφράζει μια ελεύθερη έννοια η οποία προσπαθεί να ταυτοποιηθεί με ένα υπάρχον γεγονός. Χρησιμοποιείται μόνο μέσα σε Queries ή κάποιου είδους Implication και ποτέ από μόνο του. Παραδείγματος χάριν αν θέλαμε να βρούμε μέσα από ένα ερώτημα την τωρινή κατάσταση της TK-801 θα γράφαμε: `current_state(TK-801 , ?S)` όπου S είναι η μεταβλητή στην οποία θα αποθηκευτεί η κατάσταση της δεξαμενής .



Εικόνα 4-12. Δήλωση ενός Atom σε περιβάλλον MS Visio.

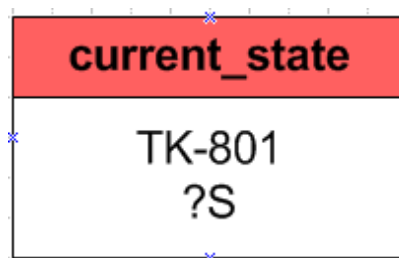
Σημείωση: Αν θέλουμε να κάνουμε αριθμητικούς ελέγχους βάζουμε στην κεφαλή τον ανάλογο τελεστή (`>=` , `>` , `<=` , `<` , `=` , `<>`).

Negative Atom

Είναι σαν το Atom μόνο που χρησιμοποιείται όταν θέλουμε να ρωτήσουμε αν κάτι δεν ισχύει.

Implication

Όταν θέλουμε να εισάγουμε κάτι σαν γεγονός τότε χρησιμοποιούμε το implication. Παραδείγματος χάριν αν θέλουμε να εισάγουμε το γεγονός ότι η κατάσταση της δεξαμενής TK-802 είναι ότι περιέχει η μεταβλητή S γράφουμε:



Εικόνα 4-13. Δήλωση ενός implication σε περιβάλλον MS Visio.

Negative Implication

Το Negative Implication χρησιμοποιείται με τον ίδιο τρόπο με την μόνη διαφορά ότι αφαιρεί γεγονότα.

Modifying Implication

Το Modifying Implication χρησιμοποιείται με τον ίδιο τρόπο με την μόνη διαφορά ότι αλλάζει υπάρχοντα γεγονότα.

Counting Implication

Το Counting Implication προσθέτει ένα γεγονός που ως σώμα του έχει τον αριθμό των γεγονότων που ταυτοποιήθηκαν με τα Atoms του Implication αυτού.

Query

Εκτελεί ένα ερώτημα με βάση τα Atoms με τα οποία συνδέεται και έπειτα επιστρέφει όλα τα γεγονότα τα οποία ταυτοποιήθηκαν.

4.5.5. Υλοποίηση και κώδικας

Στην φάση της υλοποίησης του ευφυούς συστήματος ο περισσότερος χρόνος δόθηκε στη μοντελοποίηση των κανόνων λειτουργίας του διυλιστηρίου βάση των οποίων επρόκειτο στη συνέχεια το ευφύες σύστημα να χαρακτηρίζει τις δεξαμενές. Για τη μοντελοποίηση χρησιμοποιήθηκε σαν σχεδιαστικό εργαλείο το Microsoft Visio ενώ οι έλεγχοι των κανόνων γίνονταν στο interface που παρέχεται δωρεάν μαζί με την NxBRE και αποτελεί μια κονσόλα η οποία διαχειρίζεται ένα σύστημα κανόνων μέσω διάδρασης με τον χρήστη – προγραμματιστή.

Από πλευράς κώδικα δημιουργήθηκε ένα ακόμη project κάτω από τη solution του έργου Arethusa το οποίο ονομάστηκε Sophocles και διαχειρίζονταν ουσιαστικά την NxBRE κάνοντας χρήση τριών (3) απλών μεθόδων.

```
connectAndRise()
```

Αυτή η μέθοδος αναλαμβάνει να συνδεθεί στην NxBRE, να δημιουργήσει το σύνολο των αρχικών κανόνων και να τους φορτώσει στην NxBRE (ουσιαστικά αρχικοποιεί την μηχανή κανόνων).

```
String [] findStates()
```

Αναλαμβάνει να διασχίσει τους κανόνες με τη σειρά που δόθηκαν παραπάνω (Κανόνες – §4.5.2) χαρακτηρίζοντας με αυτόν τον τρόπο την κάθε δεξαμενή ξεχωριστά. Τελικά επιστρέφει έναν πίνακα που περιέχει τους χαρακτηρισμούς όλων των δεξαμενών.

```
disconnect()
```

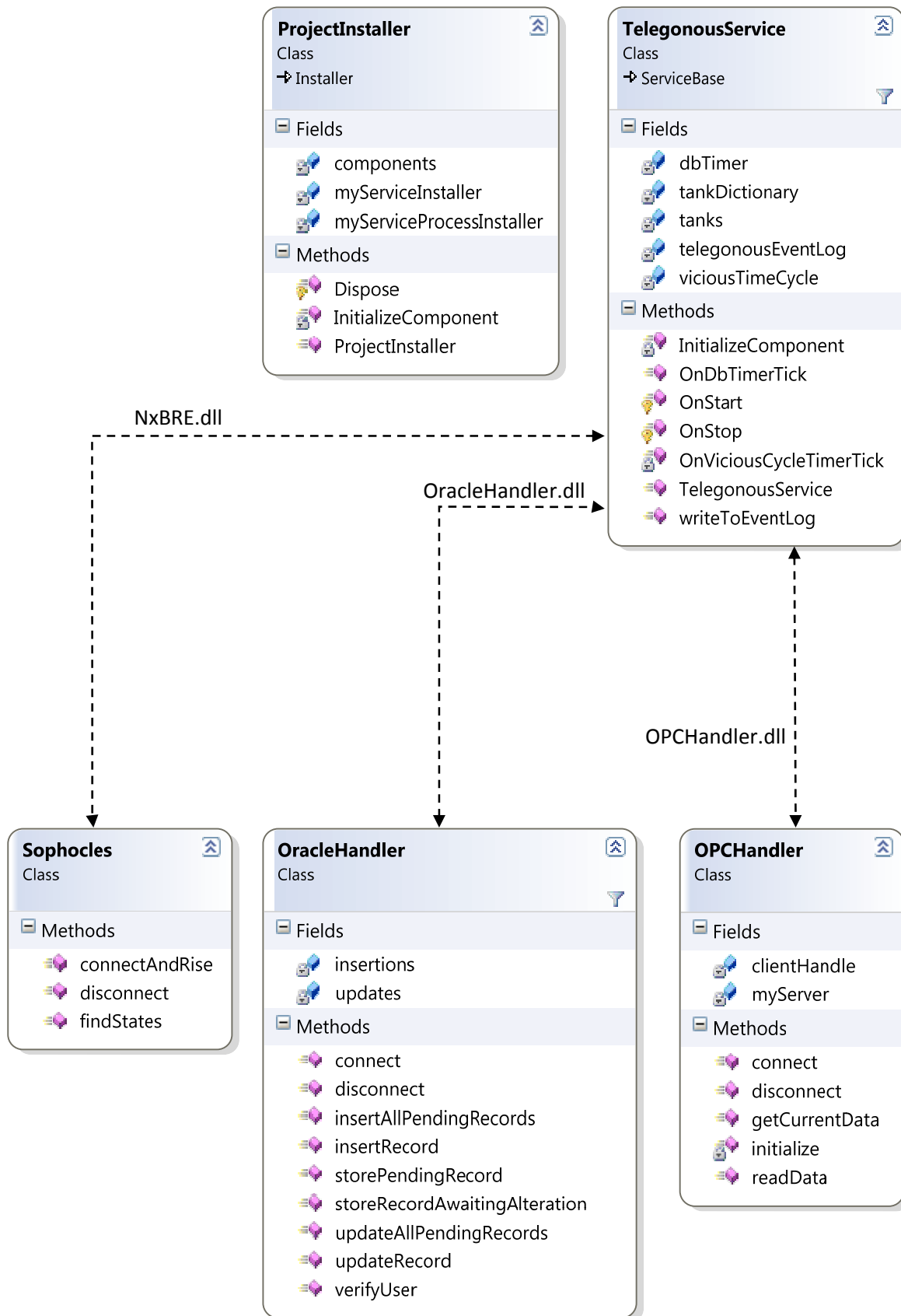
Τερματίζει τη σύνδεση με την NxBRE.

4.5.6. Αξιολόγηση υλοποίησης της πέμπτης φάσης

Η πέμπτη φάση αποτέλεσε μία από τις δυσκολότερες που κλήθηκε να αντιμετωπίσει η ομάδα ανάπτυξης και χαρακτηρίζεται από προβλήματα σε όλα τα επίπεδα. Αρχικά χάθηκε πολύς χρόνος στην μοντελοποίηση των αφηρημένων κανόνων που καταγράφηκαν μετά από πλήθος συναντήσεων και συζήτησης με την ομάδα των ειδικών. Η έλλειψη τριβής τους με την έννοια του κανόνα και συστήματος κανόνων αντίστοιχα είχε ως αποτέλεσμα την ατυχή περιγραφή των κανόνων που διέπουν τη λειτουργία του διυλιστηρίου ή την παράλειψη

σημείων και πληροφοριών που ήταν απαραίτητα για τη μοντελοποίηση της γνώσης που προσπαθούσαν να μεταδώσουν στην ομάδα ανάπτυξης. Έτσι οι κανόνες άργησαν σημαντικά να φθάσουν στην τελική τους μορφή ενώ πολλά προβλήματα εμφανίστηκαν κατά την προσπάθεια μοντελοποίησής τους λόγω των πληροφοριών που είχαν παραληφθεί νωρίτερα. Ακόμη υπήρξε μεγάλη κωλυσιεργία στην περιγραφή των κανόνων για χρήση από την NxBRE. Αυτή η κωλυσιεργία οφείλεται κυρίως στην φτωχή τεκμηρίωση της NxBRE που είχε σαν αποτέλεσμα να μην γίνει αντιληπτό εξ αρχής από την ομάδα ότι οι συμβάσεις του interface που παρέχονται από την ομάδα ανάπτυξης της NxBRE και του stencil που καθιστά εφικτή την σύνδεσή της με το Microsoft Visio δεν συμπίπτουν. Έτσι όταν χρειάστηκε να γίνει cross checking η ομάδα έχασε πολύτιμο χρόνο σε «κυνήγι φαντασμάτων» αφού δεν σκέφθηκε ότι μπορεί τα δύο εργαλεία να απαιτούν διαφορετικούς κανόνες σύνταξης. Όταν βέβαια το τελευταίο έγινε αντιληπτό είχε ήδη χαθεί πολύς και πολύτιμος χρόνος.

Ακόμη χειρότερα έγιναν τα πράγματα κατά την προσπάθεια σύνδεσης της NxBRE με το Visual Studio .NET. Και εκεί ο τρόπος αντιμετώπισης άργησε να βρεθεί αφού η λογική της χρήσης των κανόνων μέσα στο περιβάλλον ανάπτυξης του Visual Studio ήταν τελείως διαφορετική από τη χρήση μέσω των προαναφερθέντων περιβαλλόντων. Αξίζει να σημειωθεί ότι παρά τις καθυστερήσεις που υπήρξαν το έργο δεν εκτροχιάστηκε και η ομάδα δεν βρέθηκε σε πανικό. Σε αυτό βοήθησε η χημεία που αναπτύχθηκε μεταξύ των δύο συνεργατών με το πέρασμα του χρόνου και την συχνή τριβή με προβλήματα τέτοιου τύπου. Όλα αυτά είχαν σαν αποτέλεσμα τα μέλη της ομάδας να μπορούν να ανταπεξέρχονται ευκολότερα στα διάφορα προβλήματα και να δίνουν ταχύτερα αποτελεσματικότερες λύσεις. Η υλοποίηση αυτού του τμήματος του έργου ολοκληρώθηκε, τελικά, σε πέντε (5) συνεδρίες και συνδέθηκε απευθείας με την διαδικτυακή υπηρεσία. Παρουσιάστηκε στις 04/11/2007.



Σχήμα 4-2. Διάγραμμα κλάσεων του έργου Arethusa

5. Έλεγχος

Όπως γίνεται εύκολα αντιληπτό, το κομμάτι του ελέγχου είναι ύψιστης σημασίας για μια βιομηχανική εφαρμογή. Επίσης είναι πολύ σημαντικός ο προέλεγχος των υλοποιημένων τμημάτων αφού αν ένα ή περισσότερα λάθη διαπιστωθούν κατά την εγκατάσταση του τελικού προϊόντος μπορεί να προκαλέσουν σημαντικά προβλήματα. Λόγω λοιπόν της σημαντικότητας του ελέγχου στην υλοποίηση του έργου τον εξετάζουμε αναλυτικά στο παρόν κεφάλαιο.

Για να είναι σε θέση να θεμελιώσει τον προσεκτικό έλεγχο του προς υλοποίηση πληροφοριακού συστήματος, η ομάδα ανάπτυξης αποφάσισε, ατότου ήρθε σε συμφωνία με τον πελάτη, την εφαρμογή δύο βασικών μεθόδων σχεδιασμού ελέγχων και λογισμικού.

5.1.1. Ανάθεση Λειτουργιών στο Μοντέλο.

Αυτή η μεθοδολογία εφαρμόζεται συνήθως στα συστήματα που περιλαμβάνουν κάποια διεπαφή Χρήστη και είτε επιτελούν πολύπλοκες λειτουργίες είτε έχουν υπερβολικά μεγάλο αριθμό οπτικών αντικειμένων (visual components) ώστε να καθιστούν πρακτικά αδύνατο τον έλεγχο βασισμένο στην γνωστή μέθοδο "δοκιμασία και αποτυχία" (test and failure), σύμφωνα με την οποία η διεπαφή δοκιμάζεται από κάποιον, ακόμη και αν αυτός δεν συμμετέχει στην ομάδα ανάπτυξης, μέχρις ότου να βρει κάποιο λάθος. Στη συνέχεια διορθώνεται το λάθος ή τα λάθη που εντοπίστηκαν και η διαδικασία ξεκινά πάλι από την αρχή.

Κατά την *Ανάθεση Λειτουργιών στο Μοντέλο*, η διεπαφή χωρίζεται σε δύο τμήματα: στο *μοντέλο* (model), όπου επιτελούνται όλες οι λειτουργίες όπως μετατροπές δεδομένων, επικοινωνία με τη βάση δεδομένων, ανάγνωση αρχείου κλπ., και στην *όψη* (view), όπου απλά απεικονίζονται δεδομένα του μοντέλου και γίνεται διαχείριση των παραθύρων.

Με τον διαχωρισμό αυτόν είναι πλέον εύκολο να γραφούν έλεγχοι για το μοντέλο ώστε να ελεγχθεί το αν επιτελεί τις λειτουργίες του σωστά. Αν οποιαδήποτε στιγμή υπάρχει οποιοδήποτε είδους λάθος (συνήθως λογικό) τότε θα εντοπιστεί άμεσα και φυσικά θα διορθωθεί. Με τον τρόπο αυτό είμαστε βέβαιοι ότι το μοντέλο δεν παράγει λανθασμένα δεδομένα οπότε αν παρατηρείται οποιοδήποτε είδος ασυμφωνίας, είναι σε θέση κανείς να συμπεράνει ότι ευθύνεται η όψη. Λόγω της απλότητας της, η όψη σπανίως εμφανίζει λανθασμένη συμπεριφορά, δυστυχώς όμως όταν συμβεί κάτι τέτοιο η ομάδα είναι

υποχρεωμένη να εντοπίσει το λάθος με τον πατροπαράδοτο τρόπο αναζήτησης στον κώδικα της όψης.

Στο έργο αυτό η παραπάνω λογική χρησιμοποιήθηκε σε δύο επίπεδα. Η διεπαφή χρήστη έπρεπε να παρουσιάζει τα δεδομένα ανηγμένα σε ποσοστιαίες μονάδες, να τα οπτικοποιεί καθώς και να επικοινωνεί με την δικτυακή υπηρεσία ώστε να λαμβάνει τα δεδομένα αυτά. Το σκέλος της οπτικοποίησης των δεδομένων καθώς και η διαχείριση των παραθύρων ήταν οι προφανείς βασικές λειτουργίες της όψης. Η επικοινωνία μέσω του δικτύου με την υπηρεσία ήταν μια λειτουργία η οποία ήταν καθαρά ανεξάρτητη από το πώς έπρεπε να εμφανίζονται τα δεδομένα, άρα ήταν μια λειτουργία η οποία έπρεπε να ανατεθεί στο μοντέλο.

Η αναγωγή όμως των δεδομένων σε ποσοστά καθώς και κάποιοι άλλοι υπολογισμοί δεν ήταν σαφές σε ποιο τμήμα του έργου έπρεπε να ανατεθούν. Συμφωνά με αυτήν την πρακτική ήταν βέβαιο ότι έπρεπε να αποφευχθεί η ανάθεση τους στην όψη της διεπαφής. Ήταν, λοιπόν, ευθύνη της ομάδας να αποφασίσει αν θα ανατεθούν οι υπολογισμοί στο μοντέλο της διεπαφής ή στην δικτυακή υπηρεσία. Λαμβάνοντας υπ' όψην παραμέτρους όπως η επιβάρυνση του συστήματος που θα φιλοξενήσει την υπηρεσία καθώς και τη μονομερή διαθεσιμότητα όλων των δεδομένων αποφάσισε να προσθέσει την λειτουργικότητα αυτή στα αντικείμενα που ανταλλάσσουν η δικτυακή υπηρεσία με την διεπαφή χρήστη και να εκτελούνται οι υπολογισμοί όπου θεωρείται απαραίτητο. Στην παρούσα υλοποίηση οι υπολογισμοί αυτοί γίνονται εξ ολοκλήρου στο μοντέλο της διεπαφής.

5.1.2. Έλεγχος με Θεωρητικά και Πραγματικά Δεδομένα.

Από την περιγραφή του συστήματος και ιδίως από τον τόπο εγκατάστασής του, είναι εύκολο να καταλάβει κανείς την σπουδαιότητά του και την ανάγκη για εξονυχιστικό έλεγχο. Όπως έχει ήδη τονισθεί σκοπός του συστήματος είναι να ικανοποιεί απαιτήσεις αναπαράστασης καθώς και να καταγράφει με μεγάλη ακρίβεια την διακίνηση των αργών. Αναλυτικότερα, πρέπει να είναι δυνατή η εποπτεία σε ψευδό-πραγματικό χρόνο ώστε να γίνεται έλεγχος της πραγματικής τροφοδοσίας με την προγραμματισμένη και να υπάρχει η δυνατότητα διορθώσεων. Ακόμη, στην περίπτωση απωλειών, αναλυτική καταγραφή τους για την υποστήριξη κάποιων προτάσεων που θα ελαχιστοποιήσουν τέτοιες απώλειες. Τέλος, δημιουργία μιας υποδομής στην βάση δεδομένων ώστε μελλοντικά να είναι δυνατή η εφαρμογή μεθόδων ανακάλυψης γνώσης για την εύρεση νέων επικίνδυνων συνθέσεων τροφοδοσίας.

Είναι εμφανές το γεγονός ότι πρόκειται για ένα σύστημα στο οποίο ο πελάτης, κατά κύριο λόγο, θα στηριχθεί για οικονομικές αποφάσεις καθώς και για αποφυγή λαθών την στιγμή που αυτά πραγματοποιούνται με σκοπό την μεγιστοποίηση του κέρδους. Για να αποδεχθεί, αλλά κυρίως να εμπιστευτεί ο πελάτης το σύστημα αυτό, πρέπει να είναι βέβαιος ότι δεν θα οδηγηθεί σε λανθασμένες αποφάσεις, που θα του στοιχήσουν οικονομικά, λόγω εσφαλμένων μετρήσεων του συστήματος.

Με βάση τα παραπάνω η ομάδα για να ικανοποιήσει τον πελάτη σχεδίασε τους ελέγχους με τέτοιο τρόπο ώστε να είναι δυνατός ο έλεγχος από τον ίδιο. Για την ακρίβεια, οι έλεγχοι οι οποίοι συντάχθηκαν, πέρα από τους βασικούς, εξαρτώνται από εξωτερικές πηγές από τις οποίες λαμβάνουν τα δεδομένα έλεγχου. Αποτέλεσμα αυτού είναι η δυνατότητα τροφοδότησης του συστήματος ελέγχων με οποιαδήποτε δεδομένα.

Η ομάδα όμως, δεν μπορούσε να έχει στην διάθεση της εξ' αρχής δεδομένα τα οποία να είναι σε θέση να τα χρησιμοποιήσει για να εκτελέσει τους ελέγχους που επιθυμούσε. Για τον λόγο αυτό, αποφάσισε να χρησιμοποιήσει κάποιους έτοιμους προσομοιωτές για τα προϋπάρχοντα συστήματα καθώς και να ετοιμάσει κάποια θεωρητικά δεδομένα όπου δεν υπήρχε η δυνατότητα της προσομοίωσης.

Πιο συγκεκριμένα, το σύστημα της βάσης δεδομένων (Oracle Server) προσομοιώθηκε από μια δωρεάν διανομή του ίδιου, αλλά με περιορισμένες δυνατότητες, συστήματος. Εκεί αναπτύχθηκε η βάση δεδομένων του προγράμματος, όπου και θα αποθηκεύονταν οι πληροφορίες που έπρεπε να καταγραφούν σχετικά με την διακίνηση των αργών. Επίσης με βάση αυτό το σύστημα, αναπτύχθηκε και ελέγχθηκε η επικοινωνία της δικτυακής υπηρεσίας με την βάση δεδομένων.

Έπειτα, προσομοιώθηκε το σύστημα ογκομέτρησης (OPC Server) με το αντίστοιχο πρόγραμμα προσομοίωσης το οποίο παρείχε η κατασκευάστρια εταιρία. Με βάση τον προσομοιωτή αυτό, προσομοιώθηκαν οι τιμές των αισθητήρων των δεξαμενών καθώς και η επικοινωνία της δικτυακής υπηρεσίας με το σύστημα μέτρησης του όγκου των δεξαμενών.

Έχοντας επιβεβαιώσει την σωστή επικοινωνία των δύο προσομοιωτών με την δικτυακή υπηρεσία, η ομάδα έλεγξε την αρμονική λειτουργία των δύο συστημάτων αυτών μεταξύ τους, λαμβάνοντας δεδομένα από το ένα σύστημα, αποστέλλοντας τα δεδομένα αυτά για εγγραφή στο άλλο και έπειτα ανάγνωση των ίδιων δεδομένων με σκοπό την σύγκριση μεταξύ των ληφθέντων και των αναγνωσμένων.

Το πρόβλημα στην όλη αυτή διαδικασία προέκυψε όταν ολοκληρώθηκε η υλοποίηση του ευφυούς συστήματος. Αυτό συνέβη γιατί ο προσομοιωτής μέτρησης του όγκου έδινε μόνο την δυνατότητα στην ομάδα να προσομοιώσει τις δεξαμενές ανεξάρτητα τη μία από την

άλλη. Για παράδειγμα, αν και ήταν δυνατό να προσομοιωθούν δύο δεξαμενές που ο όγκος των περιεχόμενων αυξομειωνόταν, δεν ήταν δυνατό να προσομοιωθεί η ενδοδιακίνηση όπου έχουμε αύξηση του όγκου περιεχομένων μίας δεξαμενής ίση με την μείωση μίας άλλης. Συνεπώς, έπρεπε είτε να δημιουργηθεί ένας νέος προσομοιωτής, είτε να βρεθεί μία άλλη μέθοδος ελέγχου του ευφυούς συστήματος.

Η ομάδα ακολουθώντας τα γνωμικά της *Απλής Σχεδίασης* επέλεξε να σχεδιάσει και να συντάξει ξεχωριστούς ελέγχους για το ευφύες σύστημα. Η επιλογή δημιουργίας ενός νέου προσομοιωτή καθώς και η συγγραφή ελέγχων με βάση τον προσομοιωτή αυτό θεωρήθηκε υπερβολικά δαπανηρή, από πλευράς χρόνου. Για το λόγο αυτό και αφού είχε λάβει υπ' όψιν όσα είχαν υλοποιηθεί, αποφάσισε ότι θα ήταν πιο εύκολο να συντάξει αυτοματοποιημένους ελέγχους χρησιμοποιώντας την ήδη έτοιμη βάση δεδομένων.

Λαμβάνοντας δεδομένα από την προϋπάρχουσα βάση δεδομένων και εμπλουτίζοντας την με τα αποτελέσματα τα οποία θα έπρεπε να παράγει το ευφύες σύστημα συντάχθηκε μία ρουτίνα αυτοματοποιημένου ελέγχου η οποία τροφοδοτούσε το ευφύες σύστημα με τα απαραίτητα δεδομένα και έπειτα σύγκρινε τα αποτελέσματα του ευφυούς συστήματος με αυτά που είχαν τοποθετηθεί στη βάση δεδομένων.

Με αυτόν τρόπο η ομάδα κατάφερε να δημιουργήσει μια υποδομή για τον έλεγχο αποδοχής χρησιμοποιώντας πραγματικά δεδομένα τα οποία θα λάμβανε το σύστημα κατά την δοκιμή του κοντά στο περιβάλλον λειτουργίας, καθώς και την δυνατότητα συγγραφής ελέγχων με θεωρητικά δεδομένα για τον εκ των προτέρων έλεγχο της μονάδας του ευφυούς συστήματος. Ήταν δυνατό, λοιπόν, έπειτα από τις προσθήκες που έκανε στην βάση δεδομένων να εκτελέσει τους ελέγχους ανεξάρτητα αν αυτοί προέρχονταν από πραγματικά ή θεωρητικά δεδομένα.

Ένα ακόμη πλεονέκτημα ήταν η δυνατότητα επαλήθευσης των θεωρητικών δεδομένων με αυτά των πραγματικών, δηλαδή αν αυτά που είχε καταλάβει η ομάδα από την περιγραφή του πελάτη ανταποκρίνονταν στην πραγματικότητα ή υπήρξε κάποιο επικοινωνιακό χάσμα ανάμεσα τους το οποίο οδήγησε στην ασυμφωνία αυτή των δεδομένων.

6. Συμπεράσματα

Παρότι ο αναγνώστης μπορεί να βρει πλήθος αναφορών σε συμπεράσματα που προέκυψαν από την εφαρμογή των ευέλικτων μεθοδολογιών κατά την υλοποίηση του πληροφοριακού συστήματος Arethusa, θεωρήσαμε σκόπιμο να τα παρουσιάσουμε, οργανωμένα και συγκεντρωμένα, σε ένα ξεχωριστό κεφάλαιο. Στόχος αυτού του κεφαλαίου είναι να δώσει στον αναγνώστη την προσωπική μας άποψη για τα πλεονεκτήματα και τα μειονεκτήματα που εμφανίζει η χρήση των ευέλικτων μεθοδολογιών σε ένα μεγάλο έργο με πολλές απαιτήσεις, πίεση χρόνου και ανάγκη για περιορισμό των λαθών στο ελάχιστο. Το συγκεκριμένο κομμάτι της εργασίας καλείται, με άλλα λόγια, να επιβεβαιώσει ή να διαψεύσει όσα ευαγγελίζονται ότι δύνανται να παρέχουν, οι ευέλικτες μεθοδολογίες, σε αυτούς που σκοπεύουν να τις δοκιμάσουν.

Δεδομένου ότι η παρούσα εργασία διαπραγματεύθηκε κατά κύριο λόγο την περιγραφή της σχεδίασης και υλοποίησης του έργου, η ανάλυση των υπέρ και των κατά αφορά κυρίως τις δύο αυτές φάσεις. Αντίστοιχα συμπεράσματα που συνδέονται με την θεωρητική θεμελίωση των ευέλικτων μεθόδων και πιο αναλυτικά με τον Ακραίο Προγραμματισμό και τις πρακτικές του μπορούν να βρεθούν στο αντίστοιχο κεφάλαιο της πτυχιακής εργασίας του Σκαλιστή Στέφανου με τίτλο *«Εφαρμογή Ακραίου Προγραμματισμού στο Πληροφορικό Σύστημα Arethusa»*.

Αξίζει ακόμη να τονισθεί ότι όσα αναφέρονται σε αυτό το κεφάλαιο αποτελούν καθαρά προσωπικές απόψεις και δεν προέρχονται από κανένα βιβλίο, σύγγραμμα, άρθρο, δημοσίευση κλπ.

6.1.1. Πλεονεκτήματα

Ένα βασικό πλεονέκτημα που παρέχει η εφαρμογή του προγραμματισμού ανά ζεύγη στους συμβαλλόμενους (ανά ζευγάρι) είναι η ψυχολογική υποστήριξη του ενός ως προς τον άλλο. Το γεγονός ότι τα μέλη που καλούνται να δουλέψουν ως ομάδα επιλέγονται με βάση κάποια κριτήρια (τα οποία συνήθως εξετάζονται μέσω κάποιων tests) επιβάλλει οι προσωπικότητες των μελών να είναι αλληλοσυμπληρούμενες. Έτσι όταν κατά τη διάρκεια ανάπτυξης ενός έργου παρουσιασθεί κάποιο πρόβλημα, εάν ο ένας από τους συνεργάτες

πανικοβληθεί, ο άλλος καλείται να φέρει την ισορροπία και να οργανώσει τις κινήσεις κατά τέτοιο τρόπο ώστε να βγει η ομάδα από το τέλμα. Με αυτόν τον τρόπο, ακόμη και αν το έργο βαλτώσει, θα μπορέσει να ανακάμψει. Μια μεγαλύτερη ομάδα ανάπτυξης η πολυφωνία κάνει την διαδικασία λήψης αποφάσεων από την ομάδα αργή και δύσκαμπτη. Ακόμη όμως και στην περίπτωση του ενός ατόμου (ένας προγραμματιστής έχει αναλάβει την υλοποίηση κάποιου έργου) σε μια περίπτωση όπως αυτή που περιγράφεται παραπάνω δεν θα καταφέρει να ανταπεξέλθει αν βρίσκεται σε πανικό με αποτέλεσμα να χάνει πολύτιμο χρόνο. Γενικά μπορούμε να πούμε ότι η αίσθηση μόνο του ότι υπάρχει ακόμη ένα άτομο ικανό να δώσει λύση σε ενδεχόμενη εμφάνιση κάποιου σοβαρού προβλήματος κάνει το εκάστοτε μέλος να νιώθει ασφάλεια

Ένα δεύτερο πλεονέκτημα που αποτελεί απόρροια του προηγούμενου είναι η ευκολία παρουσίασης προτάσεων και ιδεών μεταξύ των μελών των ζευγαριών κατά την διαδικασία ανάπτυξης του έργου. Πράγματι για το εκάστοτε μέλος είναι ευκολότερο να παρουσιάζει τις σκέψεις του (σπουδαίες, μέτριες ή κακές) γνωρίζοντας ότι προτού αποφασιστεί κάτι θα εξετασθούν από ένα τουλάχιστον επιπλέον άτομο το οποίο είναι σχετικό με το προς υλοποίηση έργο επομένως θα είναι σε θέση να αντιληφθεί πιθανές αστοχίες που ξέφυγαν κατά την αρχική σύλληψη. Γίνεται λοιπόν φανερό ότι αυτή η λογική οδηγεί αφενός στην ταχύτερη λήψη αποφάσεων και καθιστά ικανή την εκ των προτέρων αξιολόγηση κάποιας πρότασης και των πιθανών αποτελεσμάτων αυτής χωρίς να πρέπει να ξεκινήσει η υλοποίησή της.

Ένα ακόμη αξιοσημείωτο πλεονέκτημα είναι η δυνατότητα μεταφοράς, όποτε κρίνεται απαραίτητο, μικρού ή μεγάλου μέρους του φόρτου εργασίας (workload) μεταξύ των συμβαλλόμενων μελών. Αυτή η δυνατότητα επιτρέπει την διεξαγωγή μιας συνεδρίας ακόμη και σε συνθήκες ελαφριάς ασθενείας ενός από τα δύο μέλη ή, ένας αστάθμητος παράγοντας που μπορεί να εμφανισθεί ακόμη συχνότερα, όταν ένα από τα δύο μέλη αντιμετωπίζει κάποιο πρόβλημα ψυχολογικής φύσης (υπερβολική πίεση από τις υποχρεώσεις τις οικογένειας, προσωπικά ή οικογενειακά προβλήματα, οικονομικά προβλήματα, κακή διάθεση). Σε όλες αυτές τις περιπτώσεις οι συνεδρίες διεξάγονται κανονικά με το προβληματικό μέλος να υπολειτουργεί ή σε ακραίες περιπτώσεις το μέλος αυτό δεν εμφανίζεται καθόλου και «πατοίζει» όταν παρουσιαστεί παρόμοια ανάγκη.

Η πρακτική των μικρών παραδόσεων βοηθά την ομάδα στο να μένει ανεπηρέαστη από ενδεχόμενες αλλαγές απαιτήσεων σε βάθος χρόνου. Η λογική υλοποίησης ενός μικρού τμήματος του έργου κάθε φορά μεταφέρει την προσοχή της ομάδας στο κομμάτι του έργου που υλοποιούν την συγκεκριμένη περίοδο. Έτσι αν υπάρξει ανάγκη για αλλαγή των

απαιτήσεων που αφορούν ένα μη υλοποιημένο τμήμα του έργου, εύκολα μπορούν να ληφθούν υπόψη και να ενσωματωθούν.

6.1.2. Μειονεκτήματα

Το βασικότερο μειονέκτημα του προγραμματισμού σε ζεύγη είναι ότι η παρουσία και των δύο συμβαλλόμενων είναι απαραίτητη (είδαμε πως καλύπτεται η αναγκαστική απουσία τους ενός αλλά, όπως είναι εμφανές, χρήση αυτήν την πολιτικής γίνεται μόνον σε περιπτώσεις ανάγκης). Μπορεί εν γένει η ύπαρξη του συνεργάτη να λύνει πολλά προβλήματα αλλά δεν θα πρέπει να ξεχνούμε ότι η καθημερινή, πολύωρη συσχέτιση με ένα συγκεκριμένο άτομο οδηγεί πολλές φορές σε προστριβές.

Τα οφέλη των μικρών παραδόσεων δεν εμφανίζονται σε έργα των οποίων οι λειτουργικές απαιτήσεις είναι ασαφείς. Αν ο πελάτης δεν δώσει σαφή εικόνα στην ομάδα ανάπτυξης σχετικά με το τελικό αποτέλεσμα όπως το έχει στο μυαλό του η ομάδα θα αδυνατεί να σχεδιάσει ένα πλάνο που να προσεγγίζει ορθά το πρόβλημα και να προσφέρει μια αποτελεσματική λύση με αποτέλεσμα το έργο να βαλτώσει.

7. Παραρτήματα

A. Ονοματολογία

Στα κεφάλαια της σχεδίασης και υλοποίησης ο αναγνώστης συναντά πολλές φορές αναφορές σε ονόματα υποσυστημάτων του πληροφοριακού συστήματος. Η επιλογή αυτή των ονομάτων δεν είναι τυχαία. Σε αυτό το παράρτημα θα παραθέσουμε τα ονόματα των υποσυστημάτων που απαρτίζουν το πληροφοριακό σύστημα στο σύνολό του και θα εξηγήσουμε πως έγινε η επιλογή τους.

Arethusa: Προέρχεται από το ελληνικό *Αρέθουσα*, το οποίο δηλώνει αυτόν που τείνει προς την τελειότητα. Εμφανίζεται ως νύμφη του δάσους στην ελληνική μυθολογία. Το όνομα αυτό χρησιμοποιήθηκε για να χαρακτηρίσει το πληροφοριακό σύστημα στο σύνολό του.

Telegonous: Προέρχεται από το αρχαιοελληνικό *Τηλέγονος*, το οποίο δηλώνει αυτόν που γεννιέται από απόσταση. Το όνομα χρησιμοποιήθηκε αρχικά για να περιγράψει την εφαρμογή εξυπηρέτη (δηλαδή τον πυρήνα του έργου) και επιλέχθηκε για να δηλώσει ότι η υλοποίησή του γίνονταν μακριά από εκεί όπου θα χρησιμοποιούνταν (δηλαδή τον χώρο των ΕΛ.ΠΕ., Β.Ε.Θ.). Αργότερα σε Telegonous μετονομάστηκε η υπηρεσία των Windows.

Sophocles: Προέρχεται από το ελληνικό *Σοφοκλής* και δηλώνει αυτόν που συνδυάζει ανδρεία και σοφία. Το όνομα χρησιμοποιήθηκε για να περιγράψει το ευφύες σύστημα.

Dali + El Greco: Και τα δύο ονόματα αντιστοιχούν σε ζωγράφους παγκόσμιας φήμης και επιλέχθηκαν με τη σειρά που αναφέρονται για να περιγράψουν τη διεπαφή χρήστη. Η μετάβαση από την πρώτη επιλογή στην δεύτερη υπήρξε καθαρά σοβινιστική και έγινε με γνώμονα την ελληνική καταγωγή του El Greco (Δομίνικου Θεοτοκόπουλου).

B. Φόρμες Εργασίας

A = Απόστολος	Ημερομηνία: 20/12/2006
Σ = Στέφανος	Περιγραφή: GUI Template

Ωρα έναρξης pp:	11:00	Ωρα λήξης pp:	13:00	Σύνολο ωρών:	2:00
-----------------	-------	---------------	-------	--------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1		✓						Tanks
2		✓						
3				✓				
4				✓				
5						✓		
6			✓					
7					✓			Tank Images and Colour
8		✓						
9		✓						
10		✓						
11				✓				
12				✓				
13				✓				
14			✓					
15						✓		

Άλλα - Σχόλια
<ul style="list-style-type: none"> - Εκκίνηση με Πάθος - Μικρή απογοήτευση γιατί τα πράγματα

A = Απόστολος	Ημερομηνία: 23/12/2006
Σ = Στέφανος	Περιγραφή: GUI Template

Ωρα έναρξης pp:	11:00	Ωρα λήξης pp:	13:00	Σύνολο ωρών:	2:00
-----------------	-------	---------------	-------	--------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1		✓						Flows
2		✓						
3				✓				
4				✓				
5						✓		
6		✓						Tooltip
7					✓			
8				✓				
9		✓						
10		✓						
11			✓					
12						✓		

Άλλα - Σχόλια
<ul style="list-style-type: none"> - Αναπαράσταση ροών - Διάλειμμα 15' - Πολύ καλή Δουλεία

A = Απόστολος	Ημερομηνία: 5/1/2007
Σ = Στέφανος	Περιγραφή: GUI Template

Ωρα έναρξης pp:	16:30	Ωρα λήξης pp:	19:30	Σύνολο ωρών:	3:00
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος UNIT	Άλλο Περιγραφή
1		✓						Tank Characteristics Panel
2		✓						
3				✓				
4				✓				
5				✓				
6		✓						Crude Characteristics Panel
7						✓		
8				✓				
9			✓					
10						✓		
11		✓						State Characteristics Panel
12					✓			
13				✓				
14		✓						
15		✓						
16		✓						
17				✓				
18				✓				
19			✓					
20						✓		

Άλλα - Σχόλια								
<ul style="list-style-type: none"> - Δυναμική εκκίνηση με την νέα χρονιά - Πρόβλημα με τα χαρακτηριστικά αργού - Διάλειμμα 15' για να ξελαμπικάσουμε - Πολύ καλό τελείωμα! 								

A = Απόστολος	Ημερομηνία: 5/2/2007
Σ = Στέφανος	Περιγραφή: Oracle Handler

Ωρα έναρξης pp:	18:00	Ωρα λήξης pp:	20:00	Σύνολο ωρών:	2:00
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος UNIT	Άλλο Περιγραφή
1	A	✓						
2	Σ			✓				
3	Σ	✓						Database Building
4	Σ			✓				
5	Σ	✓						
6	Σ			✓				
7	A		✓					
8	A					✓		
9	Σ	✓						Connecting
10	Σ						✓	
11	A	✓						
12	A			✓				
13	Σ		✓					
14	Σ					✓		
15	A				✓			

Άλλα - Σχόλια								
<ul style="list-style-type: none"> - Μικρή απογοήτευση λόγω του αποτελέσματος του GUI - Πονοκέφαλος και νεύρα 								

A = Απόστολος	Ημερομηνία: 13/2/2007
Σ = Στέφανος	Περιγραφή: Oracle Handler

Ωρα έναρξης pp:	14:30	Ωρα λήξης pp:	18:50	Σύνολο ωρών:	4:20
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						Single Insert
2	Σ			✓				
3	Σ			✓				
4	Σ						✓	
5	Σ					✓		Delete
6	Σ	✓						
7	A			✓				
8	A						✓	
9	A	✓						Update
10	A	✓						
11	Σ						✓	
12	A		✓					
13	A				✓			Mass Insert
14	A			✓				
15	Σ	✓						
16	A	✓						
17	A	✓						Mass Insert
18	A						✓	
19	Σ			✓				
20	Σ		✓					

Άλλα - Σχόλια								
<ul style="list-style-type: none"> - Δυναμικό ξεκίνημα - Ενθαρυντικά αποτελέσματα - Διάλειμμα 15' - Καλό αποτέλεσμα αλλά επήλθε η κόπωση 								

A = Απόστολος	Ημερομηνία: 8/3/2007
Σ = Στέφανος	Περιγραφή: Oracle Handler

Ωρα έναρξης pp:	15:30	Ωρα λήξης pp:	16:30	Σύνολο ωρών:	1:00
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						Mass Update
2	Σ			✓				
3	Σ			✓				
4	A	✓						
5	A			✓				
6	A		✓					
7	A		✓					

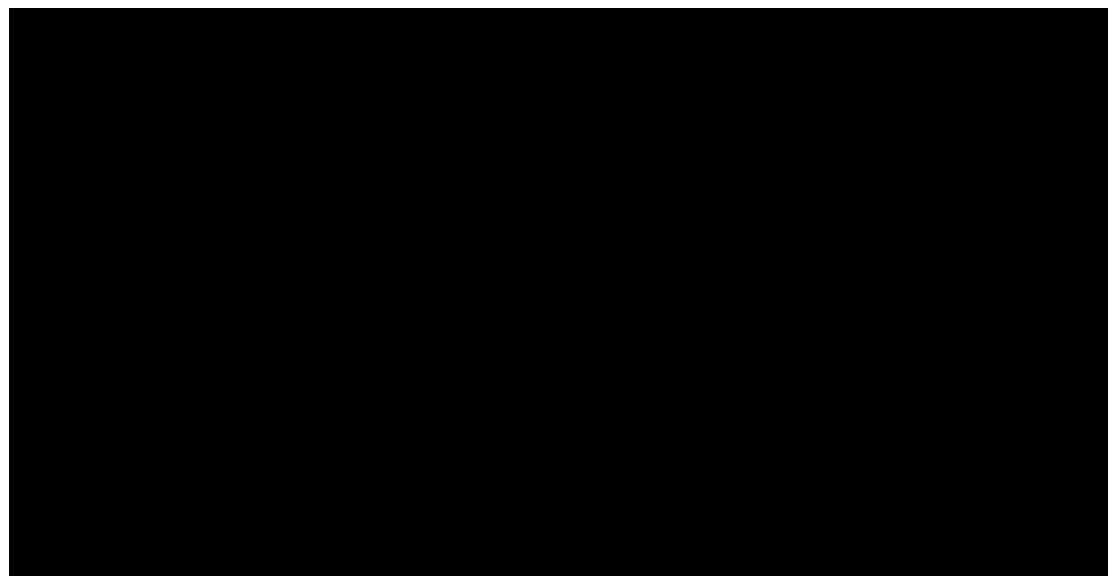
Άλλα - Σχόλια								
<ul style="list-style-type: none"> - Μουδιασμένοι - Απόφαση να το διαλύσουμε αφού δεν παράγουμε 								

A = Απόστολος	Ημερομηνία: 20/3/2007
Σ = Στέφανος	Περιγραφή: Oracle Handler

Ωρα έναρξης pp:	10:30	Ωρα λήξης pp:	16:45	Σύνολο ωρών:	6:15
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						Mass Update
2	Σ	✓						
3	A						✓	
4	A		✓				✓	
5	A				✓			
6	Σ			✓				
7	A					✓		
8	A							
9	Σ	✓						
10	Σ	✓						
11	Σ			✓				
12	A		✓					DBWindow creation
13	Σ					✓		
14	A						✓	
15	A						✓	
16	A			✓				
17	A	✓						Temp GUI
18	Σ	✓						
19	A			✓				
20	Σ		✓					

Άλλα - Σχόλια
<ul style="list-style-type: none"> - Λιγό αργό ξεκίνημα...Επαφή με τα προηγούμενα - Παρά ότι ο Σ δεν ήταν καλά...λειτουργεί η ομάδα - Διάλειμμα 30' - Ο Α τα κατάφερε μια χαρά...



A = Απόστολος	Ημερομηνία: 6/5/2007
Σ = Στέφανος	Περιγραφή: OPC Handler

Ωρα έναρξης pp:	11:45	Ωρα λήξης pp:	13:15	Σύνολο ωρών:	1:30
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						OPC Base
2	Σ			✓				
3	Α						✓	
4	Α		✓					
5	Σ					✓		
6	Σ				✓			
7	Σ						✓	
8	Α		✓					
9	Σ					✓		
10	Σ		✓					
11	Σ					✓		OPC Simulation Install & Configure
12	Σ					✓		
13	Α		✓					
14	Α					✓		
15	Α						✓	
16	Σ					✓		
17	Α			✓				
18	Α			✓				
19	Α			✓				
20	Α			✓				

Άλλα - Σχόλια
- Διάλειμμα 5'

A = Απόστολος	Ημερομηνία: 7/5/2007
Σ = Στέφανος	Περιγραφή: OPC Connection & GUI

Ωρα έναρξης pp:	18:14	Ωρα λήξης pp:	22:30	Σύνολο ωρών:	4:15
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						
2	Α			✓				
3	Σ	✓						
4	Α				✓			
5	Α						✓	Connect
6	Σ	✓						
7	Σ	✓						
8	Α			✓				
9	Α						✓	Retrieve data
10	Α				✓			
11	Σ						✓	(Fail)
12	Α		✓					
13	Α				✓			
14	Α	✓						
15	Α	✓						
16	Σ			✓				GUI connection
17	Α	✓						
18	Σ			✓				
19	Σ			✓				

Άλλα - Σχόλια
- Διάλειμμα 25'
- Nice job

A = Απόστολος	Ημερομηνία: 15/5/2007
Σ = Στέφανος	Περιγραφή: OPC GUI

Ωρα έναρξης pp:	10:45	Ωρα λήξης pp:	14:30	Σύνολο ωρών:	2:45
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ		✓					
2	Σ					✓		
3	Σ					✓		
4	Λ		✓					
5	Λ					✓		
6	Σ					✓		
7	Λ				✓			GUI Παρουσίαση OPC Server
8	Λ	✓						
9	Λ			✓				
10	Σ	✓						
11	Σ			✓				
12	Σ			✓				
13	Σ			✓				
14	Σ			✓				
15	Σ						✓	
16	Σ						✓	

Άλλα - Σχόλια
- OPC Presentation GUI Running? - Εκκίνηση συγγραφή Rules!

A = Απόστολος	Ημερομηνία: 8/7/2007
Σ = Στέφανος	Περιγραφή: Rules

Ωρα έναρξης pp:	11:30	Ωρα λήξης pp:	14:15	Σύνολο ωρών:	4:45
------------------------	--------------	----------------------	--------------	---------------------	-------------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						
2	Σ	✓						
3	Α	✓						
4	Α		✓					
5	Σ	✓						
6	Α	✓						
7	Α			✓				
8	Σ			✓				
9	Σ			✓				Tank facts
10	Σ			✓				
11	Α		✓					
12	Α			✓				
13	Α			✓				
14	Α			✓				
15	Α			✓				
16	Α			✓				
17	Α			✓				
18	Σ	✓						
19	Σ			✓				
20	Σ		✓					
21	Α		✓					
22	Σ					✓		
23	Σ			✓				
24	Α		✓					
25	Α		✓					
26	Σ					✓		
27	Σ					✓		
28	Σ	✓						
29	Σ			✓				
30	Α		✓					
31	Σ					✓		
32	Σ		✓					
33	Σ					✓		
34	Σ		✓					
35	Σ					✓		
36	Α		✓					
37	Σ					✓		

Άλλα - Σχόλια
- Διάλειμμα 10' (Δεν την παλέψαμε) - It was goood!

Σ = Στέφανος	Περιγραφή: NxBRE Rules
---------------------	-------------------------------

Ωρα έναρξης pp:	17:30	Ωρα λήξης pp:	19:30	Σύνολο ωρών:	2:00
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ		✓					
2	Σ					✓		
3	Σ					✓		
4	Α	✓						
5	Α					✓		
6	Σ	✓						
7	Σ			✓				
8	Α	✓						
9	Σ			✓				
10	Σ		✓					
11	Σ					✓		StandBy Rule
12	Σ	✓						
13	Σ			✓				
14	Α		✓					

Άλλα - Σχόλια
- Μ... στην προηγούμενη συνεδρία
- Α:Πονοκέφαλος και ψύξη

Α = Απόστολος	Ημερομηνία:21/7/2007
Σ = Στέφανος	Περιγραφή: NxBRE Rules

Ωρα έναρξης pp:	18:15	Ωρα λήξης pp:	21:00	Σύνολο ωρών:	2:45
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						Αλλαγές με expr(χαμαλίκι)
2	Α					✓		
3	Α					✓		
4	Α					✓		
5	Α		✓					Standby
6	Σ					✓		
7	Α	✓						Πρόβλημα με αριθμητικές εκφράσεις. Προσπάθεια για διόρθωση και πάλλες
8	Σ					✓		
9	Α	✓						
10	Α	✓						
11	Σ					✓		Πρόβλημα με abs()
12	Σ					✓		
13	Σ	✓						Visual Studio: Σύνδεση με NxBRE και VS2005
14	Σ			✓				
15	Σ			✓				
16	Σ			✓				
17	Σ			✓				
18	Α	✓						
19	Α			✓				

Άλλα - Σχόλια
- Ο Απόστολος αισθάνεται ότι έχει χάσει την μπάλα...
Θα χρειαστεί στην επόμενη συνεδρία επεξηγήσεις για ότι έγινε!
- Διακόπτη συνεδριών λόγω καλοκαιριού

A = Απόστολος	Ημερομηνία: 24/10/2007
Σ = Στέφανος	Περιγραφή: NxBRE Rules

Ωρα έναρξης pp:	14:30	Ωρα λήξης pp:	18:30	Σύνολο ωρών:	4:00
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ		✓					NXBRE Doc SUCKS BIG TIME!
2	Σ			✓				
3	Σ						✓	
4	Α		✓					
5	Α					✓		
6	Α			✓				
7	Α				✓			Τελείωσε το γ!@# το Stand By rule
8	Α	✓						
9	Α		✓					
10	Σ			✓				
11	Α				✓			
12	Α				✓			
13	Σ	✓						
14	Σ			✓				
15	Α					✓		Αλλαγές σε όλους τους fact/rules λόγω major λάθους λόγω λανθασμένου doc στην NXBRE
16	Α					✓		
17	Α					✓		
18	Α					✓		
19	Α					✓		
20	Α					✓		
21	Α					✓		
22	Α					✓		
23	Α					✓		
24	Σ						✓	Major ηλιθιότητα
25	Σ			✓				
26	Α	✓						
27	Α	✓						
28	Α			✓				
29	Σ	✓						
30	Σ			✓				

Άλλα - Σχόλια
<ul style="list-style-type: none"> - Διάλειμμα 15' - Εκενευρισμός & Κακή Ψυχολογία - Το Documentation της NxBRE μας οδήγησε σε λάθος υλοποίηση - Πολλές Αλλαγές

A = Απόστολος	Ημερομηνία: 2/11/2007
Σ = Στέφανος	Περιγραφή: All Together

Ωρα έναρξης pp:	10:30	Ωρα λήξης pp:	16:00	Σύνολο ωρών:	5:30
------------------------	--------------	----------------------	--------------	---------------------	-------------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						
2	Σ	✓						
3	Α	✓						
4	Σ			✓				
5	Σ			✓				
6	Σ			✓				
7	Α						✓	
8	Σ					✓		
9	Α	✓						
10	Σ			✓				
11	Σ						✓	Functionality...Does alltogether work properly??
12	Α						✓	
13	Σ						✓	
14	Α						✓	
15	Α		✓					
16	Σ					✓		
17	Α					✓		
18	Σ	✓						
19	Σ			✓				
20	Α					✓		
21	Σ						✓	
22	Σ					✓		
23	Σ	✓						
24	Σ			✓				
25	Σ			✓				
26	Σ			✓				
27	Α						✓	
28	Α					✓		
29	Σ		✓					

Άλλα - Σχόλια	- Ωρα για σοβαρό Testing όλων μαζί
- Διάρκεια 5' - Διάρκεια 30'	

A = Απόστολος	Ημερομηνία: 16/11/2007
Σ = Στέφανος	Περιγραφή: OPC Handler Remake

Ωρα έναρξης pp:	15:30	Ωρα λήξης pp:	18:00	Σύνολο ωρών:	2:30
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A					✓		OPC Handler
2	A	✓						
3	A			✓				
4	Σ			✓				
5	Σ	✓						
6	A						✓	Error Handling
7	A			✓				
8	A					✓		
9	Σ		✓					
10	A				✓			
11	Σ					✓		
12	A		✓					
13	Σ			✓				
14	A		✓					
15	A				✓			

Άλλα - Σχόλια	
+ Com #HRESULT + Error Codes differ in Gr/En Windows - InvaliidMethod removal	

A = Απόστολος	Ημερομηνία: 18/11/2007
Σ = Στέφανος	Περιγραφή: Finishing with NxBRE

Ωρα έναρξης pp:	10:30	Ωρα λήξης pp:	15:00	Σύνολο ωρών:	4:30
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						NxBRE Test
2	Σ						✓	
3	Σ			✓				
4	A	✓						
5	Σ				✓			
6	Σ			✓				
7	Σ					✓		
8	A		✓					NxBRE Port 2 Service
9	Σ				✓			
10	A							
11	Σ		✓					
12	A					✓		
13	Σ		✓					
14	Σ					✓		
15	A			✓				Rules
16	Σ	✓						
17	Σ			✓				
18	Σ	✓						
19	Σ			✓				

Άλλα - Σχόλια	
882→error 806→saab off 801,802→stand by 803,804→innertransport 805→feed (doesn't work) - Κούραση... Σπασμοδικές Ενέργειες - Διακόπη... Μιδαμινή απόδοση!	

Ωρα έναρξης pp:	10:30	Ωρα λήξης pp:	13:30	Σύνολο ωρών:	3:00
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	Σ	✓						Timers
2	Σ	✓						
3	Σ			✓				dbtimer
4	Α	✓						
5	Α			✓				datatimer
6	Α			✓				
7	Α						✓	
8	Α		✓					
9	Σ					✓		datatimer
10	Σ		✓					
11	Σ					✓		
12	Σ	✓						Service's Log
13	Σ			✓				
14	Α						✓	dbtimer
15	Σ		✓					
16	Σ					✓		
17	Α				✓			
18	Α			✓				
19	Σ	✓						
20	Σ			✓				
21	Α		✓					

Άλλα - Σχόλια
<ul style="list-style-type: none"> - Μεταφορά Εφαρμογής σε Υπηρεσία. - Διάλλειμα 5' - Διάλλειμα 10'

A = Απόστολος	Ημερομηνία: 27/11/2007
Σ = Στέφανος	Περιγραφή: Port Application 2 Service

Ωρα έναρξης pp:	10:15	Ωρα λήξης pp:	13:00	Σύνολο ωρών:	2:45
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						Timers
2	Σ	✓						
3	Σ		✓					
4	A					✓		
5	Σ			✓				
6	A			✓				
7	A						✓	
8	Σ		✓					Log Problem
9	A			✓				
10	A			✓				
11	A				✓			
12	A		✓					OPC↔Telegonous connection
13	A				✓			
14	A			✓				
15	A			✓				
16	A						✓	
17	A			✓				
18	A			✓				
19	Σ		✓					Oracle↔Telegonous connection
20	A					✓		
21	Σ	✓						
22	Σ			✓				
23	A		✓					wrong password
24	Σ					✓		
25	Σ					✓		
26	A		✓					
27	Σ						✓	
28	Σ		✓					
29	Σ					✓		

Άλλα - Σχόλια
- Συνέχεια από προηγούμενη Συνεδρία - SUCCESS!!

A = Απόστολος	Ημερομηνία: 3/12/2007
Σ = Στέφανος	Περιγραφή: GUI

Ωρα έναρξης pp:	11:15	Ωρα λήξης pp:	14:30	Σύνολο ωρών:	3:15
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						Handling Excel Graphics via .NET
2	A			✓				
3	A			✓				
4	A					✓		
5	A			✓				
6	Σ	✓						Telegonous Service Installer
7	Σ			✓				
8	Σ			✓				
9	Σ	✓						
10	Σ			✓				

Άλλα - Σχόλια								
<ul style="list-style-type: none"> - Δυσκολίες στην παρουσίαση των γραφικών αναπαραστάσεων έστω και μέσω Excel - Δημιουργία πρότασης για WPF - Εγκατάλειψη του Excel - Δημιουργία Installer της Service 								

A = Απόστολος	Ημερομηνία: 12/12/2007
Σ = Στέφανος	Περιγραφή: GUI

Ωρα έναρξης pp:	10:30	Ωρα λήξης pp:	13:00	Σύνολο ωρών:	2:30
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						
2	Σ			✓				
3	Σ			✓				
4	Σ	✓						
5	Σ			✓				
6	Σ			✓				
7	A	✓						Gui Installer
8	Σ			✓				

Άλλα - Σχόλια								
<ul style="list-style-type: none"> - Προβλήματα με το Resize - Πολύ ώρα για απλά πράγματα 								

A = Απόστολος	Ημερομηνία: 14/12/2007
Σ = Στέφανος	Περιγραφή: Port Application 2 Service

Ωρα έναρξης pp:	10:15	Ωρα λήξης pp:	16:00	Σύνολο ωρών:	5:45
------------------------	-------	----------------------	-------	---------------------	------

A/A	Αριθμός Devel	Εύρεση Στοιχείου	Εύρεση Λάθους	Προσθήκη Στοιχείου	Αφαίρεση Στοιχείου	Αλλαγή Στοιχείου	Έλεγχος JUNIT	Άλλο Περιγραφή
1	A	✓						Window Handling
2	Σ	✓						
3	Σ		✓					
4	A					✓		
5	Σ			✓				
6	A			✓				
7	A						✓	Model Checking
8	Σ		✓					
9	A			✓				
10	A			✓				
11	A						✓	Connection Problem
12	A		✓					
13	A				✓			
14	A			✓				GUI↔Telegonous connection
15	A			✓				
16	A						✓	
17	A			✓				
18	A			✓				
19	Σ		✓					
20	A					✓		Presentation Error
21	Σ	✓						
22	Σ			✓				
23	A		✓					Yet another test
24	Σ					✓		
25	Σ					✓		
26	A		✓					
27	Σ						✓	
28	Σ		✓					
29	Σ					✓		

Άλλα - Σχόλια
- Ανυπομονησία... - Διάλειμμα 25' - Τελειώσαμε????

8. Ευρετήριο

8.1. Εικόνες

Εικόνα 4-1. Διεπαφή χρήστη. Μια πρώτη προσέγγιση υπο μορφήν προτύπου	29
Εικόνα 4-2. Οι δεξαμενές και τα χαρακτηριστικά τους όπως απεικονίζονται στον Matricon OPC Server.	36
Εικόνα 4-3. Οι δεξαμενές και τα χαρακτηριστικά τους όπως απεικονίζονται στον Matricon OPC Explorer.	37
Εικόνα 4-4. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Error / Maintenance»	46
Εικόνα 4-5. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Saab Is Off»	46
Εικόνα 4-6. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Stand By»	47
Εικόνα 4-7. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Inner Transport»	48
Εικόνα 4-8. Κανόνας εύρεσης δεξαμενών σε κατάσταση «Refinery Feed»	48
Εικόνα 4-9. Παρακολούθηση των γεγονότων και εκτέλεση ερωτημάτων-κανόνων μέσω του Interface	50
Εικόνα 4-10. Ένα παράδειγμα κανόνα σχεδιασμένου σε Microsoft Visio	51
Εικόνα 4-11. Δήλωση ενός fact σε περιβάλλον MS Visio.	52
Εικόνα 4-12. Δήλωση ενός Atom σε περιβάλλον MS Visio.	52
Εικόνα 4-13. Δήλωση ενός implication σε περιβάλλον MS Visio.....	53
Σχήμα 4-2. Διάγραμμα κλάσεων του έργου Arethusa	56

8.2. Σχήματα

Σχήμα 3-1: Η Arethusa στην αρχική της μορφή.....	21
Σχήμα 3-2: Ο Telegonous τροφοδοτεί τρία στιγμιότυπα του Dali	22
Σχήμα 3-3: Η Arethusa μετά την εισαγωγή του ευφυούς συστήματος Sophocles.....	23
Σχήμα 4-1. ER διαγράμματα για τους πίνακες tanks και refinery.	31
Σχήμα 4-2. Προσέγγιση του περιεχομένου του myServer με δένδρική αναπαράσταση.	39

9. Βιβλιογραφία

1. **Janis Greenberg**. Oracle Data Provider for .NET Developer's Guide 10g (release 2). 2002.
2. **Randy Kondor**. OPC Tutorial. [Ηλεκτρονικό].
<http://www.matrikon.com/tutorial>
3. **Matrikon Inc.** Matrikon Solutions for Industrial Agility. Introduction to OPC. 2002.
4. **Matrikon Inc.** Matrikon Solutions for Industrial Agility. OPC Server for DDE. 2004.
5. **Matrikon Inc.** MatrikonOPCData Manager, User Manual . 2006.
6. **Nancy Greenberg, Priya Nathan**. Introduction to Oracle: SQL, Student Guide (Volume I). 2002.
7. **Nancy Greenberg, Priya Nathan**. Introduction to Oracle: SQL, Student Guide (Volume II). 2002.
8. **Kent Beck, Martin Fowler**. Planning Extreme Programming. Addison Wesley. 2000.
9. **Laurie Williams, Robert Kessler**. Pair Programming Illuminated. Addison Wesley. 2002.
10. **Neil Roodyn**. eXtreme .NET: Introducing eXtreme Programming Techniques to .NET Developers. 2004.
11. **P.G. Sarang, Rahim Adatia, Bruno Joughier**. J#.
12. **John Longshaw, Andy Sharp**. Microsoft Visual J# .NET (Core Reference). 2003.
13. **Alistair Cockburn Laurie Williams**. The Costs and Benefits of Pair Programming. [Δημοσίευση].
14. **Richard Hightower, Warner Onstine et al**. Professional Java Tools for Extreme Programming. Wrox Press. 2004.
15. **Dave Grundgeiger** . Programming Visual Basic .NET. O'Reilly. 2002
16. **Juval Louy**. COM and .Net component services. O'Reilly.
17. **Bill Pribyl**. Learning Oracle PL/SQL. O'Reilly. 2001.
18. **Ian Griffiths and Matthew Adams**. .NET windows forms in a nutshell. O'Reilly.
19. **John Alexander, Billy Hollis**. Developing Web Applications with Visual Basic .NET and ASP.NET. Willey & Sons. 2002
20. **Alistair Cockburn**. Agile Software Development. 2000.
21. **Lisa Crispin, Tip House**. Testing Extreme Programming. Addison Wesley.
22. **William C. Wake**. Extreme Programming Explored. 2000.
23. **David Dossot**. .NET Business Rules Engine. [Ηλεκτρονικό]
<http://nxbre.org>
24. **Kent Beck**. Extreme Programming Explained, Embrace change. 1999