

Αρχές Σπονδυλωτού Προγραμματισμού σε Κατανεμημένα Συστήματα



Κεφάλαιο Ένα
-
Συστατικά και Αγορές

1.1 Εισαγωγή

1.1.1 Σύνθεση με Συστατικά

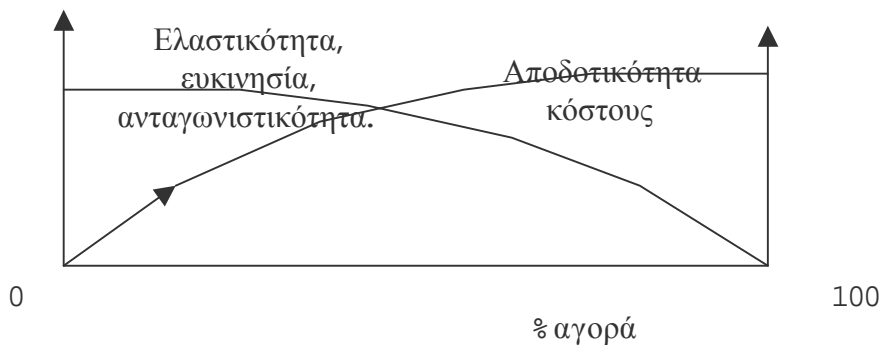
- Κύριο χαρακτηριστικό των συστημάτων που συναρμολογούνται από συστατικά είναι η σύνθεση ή επανασύνθεση με τα ίδια κοινά συστατικά. Το λογισμικό τους καλείται λογισμικό συστατικών.
- Επομένως και οι ιδιότητες των συστατικών πρέπει να είναι στον κατάλληλο βαθμό γενικές ή ειδικές ώστε να επιτρέπουν την χρησιμοποίησή τους σε διαφορετικά πλαίσια. Οι κύριες ιδιότητες είναι η *ανεξαρτησία* και η *δυναμική μορφή*.

1.1.2 Επί Παραγγελία και Καθιερωμένο Λογισμικό

- Στο επί παραγγελία λογισμικό τα συστήματα αναπτύσσονται με βάση εργαλεία προγραμματισμού και βιβλιοθήκες επεξεργασιών. Οι λύσεις που προσφέρει πέρα από την ανάπτυξη τοπικών εφαρμογών είναι λιγότερο από ιδανικές. Τα κυριότερα μειονεκτήματα είναι η συντήρηση, η εγγραφή στο διαδύκτιο και η διαλειτουργικότητα. Η πιθανότητα το όλο σύστημα να γίνει παραγωγικό πριν γίνει ξεπερασμένο είναι μικρή.
- Στο άλλο άκρο καθιερωμένο λογισμικό αγοράζεται και παραμετροποιείται για κατά προσέγγιση λύση. Η ευθύνη της συντήρησης και διαλειτουργικότητας αφήνεται στον πωλητή. Η αναλογία χρόνου και χρήματος είναι πολύ καλύτερη.
 - Τα προβλήματα αυτού του είδους προγραμματισμού βρίσκονται στην αναδιοργάνωση των διεργασιών που επιρρεάζονται.
 - Το καθιερωμένο λογισμικό δεν είναι ανταγωνιστικό αφού είναι διαθέσιμο προς όλους.
 - Ο κεντρικός έλεγχός του, δυσχεραίνει την προσαρμοστικότητα σε τοπικό επίπεδο.
- Ο σπονδυλωτός προγραμματισμός βρίσκεται στο μέσο των δύο λύσεων. Παρόλο που κάθε συστατικό στην αγορά είναι καθιερωμένο προϊόν, με όλα τα πλεονεκτήματά του, η διαδικασία της συναρμολόγησης επιτρέπει και την τυποποίηση. Φαίνεται ότι θα υπάρχουν διαφορετικής ποιότητας συστατικά (επίπεδο εκτέλεσης, αποδοτικότητα πόρων, αυτοδυναμία) σε διαφορετικές τιμές (βλ. σχ. 1.1) Τα συστατικά επίσης θέτουν τέλος στο πρόβλημα της μαζικής περιοδικής αναβάθμισης που υπάρχει στο παραδοσιακό λογισμικό. Και αυτό γιατί η ξεχωριστή αναβάθμιση σε επίπεδο συστατικών γίνεται 'εκτός φάσης' και περισσότερο ομαλά.

1.1.3 Το Αναπόφευκτο των Συστατικών

- Η ανάπτυξη συστατικών υψηλής τεχνολογίας δεν είναι αρκετή για να καθιερωθούν αυτά στην αγορά. Εδώ εισάγεται ο όρος *κρίσιμη μάζα* που πρέπει να έχει το συστατικό για την εμπορική του επιτυχία και συνίσταται από:
 - Ικανοποιητική ποικιλία και ποιότητα.
 - Αν υπάρχει όφελος από την χρήση του συστατικού.
 - Αν υποστηρίζεται από δυνατούς πόρους ή αρκετούς δευτερεύοντες πόρους.
- Αφού αποκλειθεί αυτή η κρίσιμη μάζα σε ένα τομέα της αγοράς, τότε η χρήση των συστατικών σ' αυτό το τομέα θα είναι αναπόφευκτη. Τότε μία 'δείνη' θα σχηματισθεί που θα τραβήξει παραδοσιακές λύσεις στην κατανάλωση.
- Η κατασκευή δικών μας λύσεων δικαιολογείται μόνο όταν είναι πολύ ανώτερες από τα υπάρχοντα συστατικά. Επίσης τα συστατικά στην αγορά βελτιώνονται ταχύτερα από τις 'χειροποίητες' λύσεις. Καθώς τα συστατικά ενεργούν σαν πολλαπλασιαστές στην αγορά, η αύξηση μπορεί να γίνει εκθετική.
- Η εισαγωγή στην αγορά συστατικών απαιτεί προετοιμασία δηλ. επιλογή προσέγγισης μηχανικής λογισμικού φιλικής προς την δημιουργία συστατικών (υποστήριξη διαμόρφωσης απαιτήσεων, αρχιτεκτονικής, σχεδιασμού και εφαρμογής). Επομένως υποχρεωτικά χρησιμοποιείται καλύτερη προσέγγιση μηχανικής λογισμικού.
- Ο πρώτος οργανισμός που δημιουργεί ένα αξιόλογο σύνολο συστατικών για ένα τομέα της αγοράς μπορεί να καθορίσει κανονισμούς και μορφές ανάπτυξης της αγοράς στο δικό του όφελος.



Σχ. 1.1 Φάσμα μεταξύ παραγωγής και αγοράς.

1.1.4 Η Φύση του Λογισμικού και οι Αναπτυσσόμενες Οντότητες

- Η φύση του λογισμικού διαφέρει από άλλα προϊόντα του τομέα της μηχανικής στο ότι κάθε παραγωγή του είναι αντίγραφο προϊόντος. Οι υπολογιστές θεωρούνται σαν μηχανές που δέχονται και ενεργοποιούν αυτά τα αντίγραφα όσες φορές χρειαστεί. Έτσι το λογισμικό πρέπει να θεωρείται σαν μεταπροϊόν. Το ίδιο ισχύει για τα συστατικά λογισμικού.
- Έχει υπάρξει σύγχυση σε σχέση με τις αφηρημένες έννοιες και τις περιπτώσεις (instances), αφότου δημιουργήθηκε το πρότυπο συσχέτισης οντοτήτων (entity-relationship modeling). Κανονικά θα έπρεπε εξ' αρχής να εχρησιμοποιούντο οι εκφράσεις 'συμβάν οντότητας' (entity occurrence) και 'ορισμός οντότητας' (entity definition). Στην τεχνολογία των αντικειμένων (objects) η διάκριση μεταξύ τάξης και αντικειμένου συχνά παραλείπεται. Αν και υπάρχει διαύγεια σχετική με όρους όπως 'περίπτωση αντικειμένου' ή 'τάξη αντικειμένου', στην καθιερωμένη πρακτική υπερτερεί η μη διάκριση μεταξύ τάξης και αντικειμένου.
- Η τεχνική προτύπου αντικειμένου (object modeling technique, OMT) περιγράφει τις στατικές σχέσεις των τάξεων, αλλά όταν γίνεται αναφορά στις επισημειώσεις για αριθμούς συνεταίρων σε μία σχέση, τότε εννοούνται περιπτώσεις τάξεων (αντικείμενα). Η ενεργοποίηση αυτών των αντικειμένων δημιουργεί περιπτώσεις αντικειμένων. Δηλαδή οι τάξεις από μόνες τους θεωρούνται αντικείμενα που οι εφαρμογές τους (αφού οι τάξεις έχουν κάποιες παραμέτρους) είναι και αυτές αντικείμενα, έτοιμα να δημιουργήσουν περιπτώσεις με κάθε ενεργοποίησή τους.
- Συμπερασματικά, τα συστατικά δεν είναι αναντικατάστατα και η υπεροχή τους βρίσκεται σε θέματα επαναχρησιμοποίησης, προθεσμίας στην αγορά, ποιότητας και βιωσιμότητας.

1.1.5 Τα Συστατικά Είναι Μονάδες Ανάπτυξης

- Σαν μονάδα ανάπτυξης το συστατικό συνίσταται σε υποδομή μίας η περισσότερων τάξεων. Η δημιουργία αντικειμένων βασίζεται σ' αυτές τις τάξεις. Έτσι ένα συστατικό σπάνια ενεργοποιείται ολόκληρο. Όταν αποτελείται από συλλογή τάξεων, καλείται *διαμόρφωμα* (module). Ωστόσο ένα συστατικό μπορεί να περιέχει και άλλα είδη τεχνολογίας όπως συναρτήσεις ή γλώσσα assembly.
- Οι λόγοι που υπάρχουν τάξεις στα συστατικά και όχι απ' ευθείας αντικείμενα:
 - Ο ορισμός των αντικειμένων είναι περισσότερο τεχνικός, δηλαδή τα αντικείμενα περιέχουν κατάσταση και συμπεριφορά, πολυμορφισμό και διαδοχή.
 - Η τεχνολογία των αντικειμένων αγνοεί τις προοπτικές οικονομίας και αγοράς και τις τεχνικές τους επιπτώσεις.
- Η αποτυχία των αγορών αντικειμένων έγκειται και στην αδιαφορία των τεχνολόγων που ασχολούνται περισσότερο με τεχνολογικά προβλήματα.
- Ένα συστατικό τάξεων έχει περισσότερες χρήσεις και επομένως χρήστες και έτσι είναι βιωσιμότερο. Αυτή είναι η βασική ιδέα πίσω από την έννοια της επαναχρησιμοποίησης. Πλεονεκτήματά του μπορεί να είναι η τεχνολογική ανωτερότητα, η πρώτη λύση σε γνωστό πρόβλημα, ή η πλατειά υποστήριξη.

1.1.6 Διδάγματα

- Παραδείγματα επιτυχημένων συστατικών.
 - Microsoft Visual Basic.
 - Όλα τα σύγχρονα λειτουργικά συστήματα.
 - Οι εφαρμογές είναι χονδρά-διαμελισμένα συστατικά που εκτελούν στο περιβάλλον κάποιου λειτουργικού. Η διαλειτουργικότητα τέτοιων συστατικών, είναι τόσο παλιά όσο το μοίρασμα του συστήματος αρχείων και το κοινό σχήμα (format) αρχείων, ή την χρήση σωλήνα μεταφοράς στοιχείων (pipe) και η σύνθεση διηθητήρα (filter).
 - Μηχανές συσχέτισης βάσης δεδομένων.
 - Οθόνες επεξεργασίας κινήσεων – μεταβολών.
 - Αρχιτεκτονικές βύσματος (plug – in). Εφαρμόστηκαν με λεπτότερου διαμελισμού συστατικά και διαδόθηκαν με την εισαγωγή των παρατηρητών του Netscape Navigator Web.
 - Apple’s QuickTime για Mac OS.
 - Εφαρμογές TSR (terminate and stay resident) για DOS.
- Πολλά συστατικά από διαφορετικές πηγές μπορούν να συνυπάρχουν στην ίδια εγκατάσταση αν και οι αθαίρετοι συνδυασμοί δεν είναι οι καλύτεροι. Σε όλα τα πετυχημένα συστατικά υπάρχει απ’ ευθείας εμφάνιση της λειτουργικότητας τους στον πελάτη. Αυτό δεν συμβαίνει με τα αντικείμενα, βιβλιοθήκες τάξεων και τις υποδομές, γι’ αυτό και η σύνθεση των συστατικών γίνεται από ειδικούς προγραμματιστές.
- Στις μέρες μας υπάρχει και η τάση για δημιουργία αντικειμένων που μπορούν να συναρμολογήσουν και οι μη ειδικοί. Η ρύθμιση και ολοκλήρωση ενός αντικειμένου σε ένα σύστημα δεν είναι εύκολη υπόθεση και έτσι δεν πωλούνται ανεξάρτητα. Όταν πρέπει να συνδυασθούν υποδομές τα πράγματα είναι ακόμα πιο δύσκολα. Ένα τέτοιο αποτυχημένο σύστημα είναι το CommonPoint. Εκτός της πολυπλοκότητας ένας άλλος λόγος αποτυχίας του είναι η χρησιμοποίηση της C++ για στήριξη ενός πλαισίου συστατικών με πολλαπλές ιεραρχίες διαδοχής.
- Ο διαμελισμός των συστατικών και οι αμοιβαίες εξαρτήσεις πρέπει να ελέγχονται από εξωτερικό επίπεδο. Σημαντικό ρόλο στην σύνθεση συστατικών παίζει και η τεχνολογία των αντικειμένων.

1.2 Αγορά και Τεχνολογία

1.2.1 Δημιουργία Αγοράς

- Ένα νέο προϊόν μπορεί να δημιουργήσει αγορά αν η άφιξή του αναμένεται.
- Ο διακριτικός τρόπος για αποφυγή δημιουργίας νέας αγοράς είναι η επέκταση υφιστάμενης. Αυτό χρειάζεται δύο στάδια:
 - Σε περιβάλλον ανταγωνισμού πρέπει να βελτιώνονται οι προσφορές για διατήρηση της υπάρχουσας αγοράς.
 - Η υπάρχουσα βάση πελατών βοηθά στην έλξη νέων πελατών για το εξεληγμένο προϊόν.
 - Οι χειριστές της Visual Basic Extensions (VBX) γενικεύτηκαν σε χειριστές Object Linking and Embedding (OLE) και ακολούθως μετασηματίστηκαν σε χειριστές ActiveX που επίσης δουλεύει με εφαρμογές του διαδικτύου (Internet), επεκτείνοντας έτσι την αγορά.
- Η παραγωγή συστατικών πρέπει να στοιχίζει λιγότερο από την παραγωγή ολοκληρωμένων λύσεων.

1.2.2 Θεμελιώδεις Ιδιότητες Τεχνολογίας Συστατικών

- Η ύπαρξη αγοράς συστατικών βασίζεται στο εφικτόν της τεχνολογίας. Όταν συστατικά από ανεξάρτητες πηγές λογισμικού ολοκληρώνονται από τρίτους, τότε προκαλούνται τα περισσότερα τεχνικά προβλήματα. Έτσι ένα σύστημα συστατικών είναι τόσο δυνατό όσο το πιο αδύνατό του συστατικό. Η λύση είναι η απομόνωση λαθών σε κάθε συστατικό ξεχωριστά.
- Τον ατομικό έλεγχο πρέπει να ακολουθεί ο έλεγχος ολόκληρου του συστήματος. Τα υπόλοιπα των λαθών ωφείλονται στο συνδυασμό των συστατικών. Η ολοκλήρωση από τρίτους είναι ακόμη πιο δύσκολη.
- Η καθυστερημένη ολοκλήρωση, όπως συμβαίνει με τα Java applets έχει αρνητική δοκιμή ολοκλήρωσης, αφού μέχρι να αποκτηθούν τα applets τα συστατικά έχουν αποκτηθεί και αναπτυχθεί. Ο έλεγχος έκδοσης είναι χρήσιμος σε τέτοιες περιπτώσεις. Έτσι το θέμα της καθυστερημένης (και συνεχούς) εισαγωγής και ανάπτυξης συστατικών σε ένα σύστημα εμποδίζει τον κανονικό του έλεγχο.
- Το καλύτερο που μπορεί να γίνει είναι οι *τμηματικές δοκιμές* (modular checking) αντί της τελικής ολοκληρωμένης. Έτσι γίνεται ανάλυση των ιδιοτήτων ενός συστατικού και των διασυνδέσεων πάνω στις οποίες κτίζει. Οι ιδιότητες των συστατικών πρέπει να είναι τέτοιες ώστε ακόμα και αν το συστατικό αποτύχει δεν παραβιάζει κανόνες του συστήματος. Μία από τις κρίσιμες αποφάσεις είναι η κατάλληλη εκλογή γλώσσας προγραμματισμού και εργαλείων, έτσι ώστε μέρος από το βάρος της ασφάλειας των ιδιοτήτων να φεύγει από τους προγραμματιστές των συστατικών. Διεργασίες ανάπτυξης λογισμικού, που δεν εξαρτώνται από δοκιμές όπως το Cleanroom φυσικά αποκτούν σημασία.
- Μετά τη διαπίστωση των ιδιοτήτων ασφάλειας ακολουθούν οι δοκιμές λειτουργικότητας και απόδοσης. Αυτό μπορεί να γίνει με το ταίριασμα διαφόρων υπηρεσιών αντικειμένων για διαπίστωση της διαλειτουργικότητάς τους.
- Η απόδοση του συστήματος επιρρεάζεται κυρίως από τη σύνθεσή του.

1.3. Κριτήρια

1.3.1 Η Ύψιστη Σημασία των Κριτηρίων

- Οι ανάγκες ενός πετυχημένου συστατικού πρέπει να υποστηρίζονται από ευρύ φάσμα συστημάτων, οι δε υπηρεσίες του να είναι αναγκαίες σε πολλούς. Εν συντομία το κομμάτι της αγοράς που θα πάρει το συστατικό θα πρέπει να του παρέχει οικονομική βιωσιμότητα.
- Αν το τμήμα αυτό της αγοράς περιέχει λίγους πελάτες τότε ο πωλητής εύκολα κατανοεί τις ανάγκες το καθενός και τα περιβάλλοντα ανάπτυξης. Μπορεί να υπάρξει και η ακραία περίπτωση της διαμόρφωσης ενός συστατικού για ένα μόνο πελάτη. Αυτό καλείται κατασκευή διαμορφωτή.
- Τελικά με την αύξηση των χρήσεων και των πελατών, ο συνδυασμός περιβάλλον ανάπτυξης – ανάγκες πελάτη γίνεται ολοένα και πιο δύσκολο να ικανοποιηθεί. Έτσι συχνά τα κριτήρια ανάπτυξης είναι προσεγγίσεις καινοτόμων, δηλαδή των πρώτων πωλητών σε κάθε τμήμα της αγοράς. Οι πελάτες ωξάνονται παράλληλα με τους πωλητές και οι αρχικά ιδιωτικές προσεγγίσεις γίνονται βιομηχανικά κριτήρια. Τα κριτήρια πρέπει να δίνουν ικανοποιητικές πληροφορίες για τις διασυνδέσεις ενός συστατικού, ώστε να μπορούν αρκετοί πελάτες και πωλητές να συνεργάζονται.
- Υπάρχουν δύο προσεγγίσεις δημιουργίας των συστατικών λογισμικού και των κριτηρίων αγοράς.
 - Η μία βασίζεται πρώτα στη δημιουργία αγοράς και ακολούθως στη δημιουργία κριτηρίων. Αυτή είναι η προσέγγιση της Microsoft και της Sun Microsystems. Εδώ προηγείται η δημιουργία λειτουργικών προϊόντων πριν από τα κριτήρια και έτσι μπορεί να έχουμε τεχνολογική ανεπάρκεια. Η μετεξέλιξη των προϊόντων πρέπει να γίνει προσεκτικά ώστε να μην χαθεί η δημιουργηθείσα βάση των πελατών.
 - Η δεύτερη δημιουργεί πρώτα τα κριτήρια και έπειτα τις αγορές. Τέτοιο παράδειγμα είναι η βιομηχανική κοινοπραξία Object Management Group (OMG) που καθορίζει τα κριτήρια της στο Object Management Architecture (OMA).

1.3.2 Κριτήρια σύνδεσης συστατικών

- Συμβατότητα εισόδων και εξόδων ή και χρησιμοποίηση τεχνολογίας γεφύρωσης των διαφορών.
- Στο πρότυπο όπου πρώτα δημιουργούνται τα κριτήρια (όπως OMG), αυτά πρώτα θα ασχοληθούν με το επίπεδο των διασυνδέσεων των συστατικών. Έτσι η OMG πρώτα ασχολήθηκε με την αρχιτεκτονική μεσάζοντα CORBA, που επιτρέπει την επικοινωνία μεταξύ αντικειμένων που γράφτηκαν με διαφορετικές γλώσσες και υποστηρίζονται από διαφορετικά λειτουργικά συστήματα.
- Αντιθέτως η Microsoft είχε πρώτα τα προϊόντα OLE και Visual Basic, που αποτελούν επιτυχημένες τεχνολογίες υψηλού επιπέδου και αργότερα διεύρυνε τις επιλογές της με COM, generic OLE, ActiveX, Visual Basic for Applications και άλλα. Μόνο τότε άρχισε να εκδίδει κάποια κριτήρια ενθαρρύνοντας και άλλους πωλητές να ακολουθήσουν. Αυτά συντηρούνται από την Active Group που είναι μέλος της Open Group.
- Στις περιοχές εφαρμογών των συστατικών (όπως σύνθετου κειμένου) τα κριτήρια υλοποιούνται με συστατικά κώδικα και δεν έχουν αναπτυχθεί πολύ.