Β.Βεσκούκης

2000 προβλήματα και παραλείψεις στην πληροφορική

Ορισμός του "προβλήματος του 2000" ερήμην των ...δημοσιογράφων

Ενα φάντασμα πλανιέται πάνω από τον κόσμο της πληροφορικής: το φάντασμα του 2000. Αναφορές το αποκαλούν "ιό του 2000", άλλες θεωρούν ότι το πρόβλημα θα εμφανιστεί ακριβώς το 2000 και κάποιοι θέλουν να μας κάνουν να πιστέψουμε ότι πρόκειται για κάτι τρομακτικό και ότι όλοι έχουμε πρόβλημα. Αλλοι αποκαλούν την κατάσταση αυτή "πρόβλημα", άλλοι "πρόκληση", άλλοι "κρίση" ενώ κυκλοφορούν και διάφορα άλλα ονόματα ανάλογα με την προέλευση, την ιδιοσυγκρασία και το σκοπό του εκάστοτε "ειδικού αναλυτή". Ενα είναι σίγουρο: η δομή του λογισμικού εφαρμογών και συστήματος αλλά και του υλικού, σε μερικές και μόνο περιπτώσεις δεν είναι επαρκής για την διεξαγωγή υπολογισμών με ημερομηνίες μεγαλύτερες του 1999 και τα λάθη που προκύπτουν μπορούν να έχουν σημαντικές επιπτώσεις σε μεγάλης κλίμακας οικονομικής φύσεως εφαρμογές. Το θέμα έχει τεχνικές και μη-τεχνικές αιτίες, αποτελέσματα και αντιμετώπιση. Σε καθένα από τα τρία αυτά επίπεδα υπάρχουν μέτρα που πρέπει να ληφθούν ώστε να αντιμετωπιστεί το άμεσο πρόβλημα, αλλά και αξιολόγηση που πρέπει να γίνει για να καταγραφούν τα πραγματικά του αίτια και να αποφευχθούν παρόμοιες καταστάσεις στο μέλλον. Τα δε εμπλεκόμενα μέρη είναι οι εταιρίες υλικού και λογισμικού, οι managers και ασφαλώς οι προγραμματιστές, οι τεχνικοί και οι επιστήμονες πληροφορικής γενικότερα.

Ορισμός

Το πρόβλημα από τεχνικής πλευράς είναι ιδιαίτερα απλό: όταν η παράσταση ημερομηνιών σε κάποιο υπολογιστικό σύστημα γίνεται με χρήση διψήφιων αριθμών (95 αντί 1995), τότε υπάρχει πρόβλημα στους υπολογισμούς που αφορούν ημερομηνίες. Χαρακτηριστικότερη πλευρά του προβλήματος είναι το αποτέλεσμα μιας αφαίρεσης ημερομηνιών το οποίο μπορεί να είναι αρνητικός αριθμός μιας και το 2000 σε διψήφια παράσταση είναι "00", το 2001 είναι "01", κ.ο.κ. Αν τα χρόνια εργασίας προκειμένου να συνταξιοδοτηθεί κάποιος εργαζόμενος οφείλουν να είναι τουλάχιστον (λ.χ.) 35, τότε ο έλεγχος αν κάποιος ο οποίος ξεκίνησε να εργάζεται το 1966 μπορεί να συνταξιοδοτηθεί το 2001 εμπεριέχει τον υπολογισμό 2001-1966 = 35. Αν όμως για την παράσταση των ημερομηνιών χρησιμοποιούνται διψήφιοι αριθμοί τότε η παραπάνω αφαίρεση είναι 01-66 = -65 οπότε υπάρχει σοβαρό σφάλμα. Παρόμοιο πρόβλημα εμφανίζεται και κατά την ταξινόμηση ημερομηνιών όπου οι μεγαλύτερες ή ίσες του 2000 ημερομηνίες ταξινομούνται πριν από όλες (καθότι λ.χ. το 02 είναι αριθμητικά μικρότερο του λ.χ. 70, ενώ στην πραγματικότητα το 2002 είναι μεγαλύτερο του 1970).

Αυτό που πρέπει να διευκρινιστεί από την αρχή, είναι ότι η εμβέλεια του προβλήματος δεν περιορίζεται μόνο στο λογισμικό, αλλά φτάνει σε οποιοδήποτε επίπεδο από τον μικροκώδικα που ενσωματώνεται στους μικροεπεξεργαστές μέχρι και στα προγράμματα εφαρμογών, περνώντας από το BIOS, το firmware, το λειτουργικό σύστημα, τις γλώσσες προγραμματισμού, τα συστήματα αρχείων, τα συστήματα διαχείρισης Βάσεων Δεδομένων και άλλα κατά περίπτωση επίπεδα. Η εμφάνιση του προβλήματος δεν τοποθετείται υποχρεωτικά στο 2000 και σε πολλές περιπτώσεις αναμένεται να γίνει νωρίτερα κατά τη διεξαγωγή πάσης φύσεως υπολογισμών που περιλαμβάνουν μελλοντικές ημερομηνίες, όπως για παράδειγμα

αποδόσεις επενδύσεων, αποπληρωμή δανείων, συνταξιοδοτήσεις, προβλέψεις οικονομικών μεγεθών, κ.ά.

Από το παραπάνω προκύπτει ότι τα επίπεδα που πρέπει να ελεγχθούν για να διαπιστωθεί εάν το μηχανογραφικό σύστημα κάποιου οργανισμού/εταιρίας έχει πρόβλημα, είναι πολλά και ανομοιογενή. Οι μεγάλες μηχανές mainframes που επί δεκαετίες λειτουργούν σε μηχανογραφικά κέντρα μεγάλων εταιριών και οργανισμών ενδεχομένως να έχουν πρόβλημα ακόμα και στο υλικό τους. Στο επίπεδο του Λογισμικού, η επί μακράν χρήση γλωσσών προγραμματισμού που αποτελούσαν αποκλειστική ιδιοκτησία της κατασκευάστριας του υλικού εταιρίας επαναφέρει τους μεγάλους κατασκευαστές του παρελθόντος (ή το προσωπικό που κάποτε εργάζονταν σε αυτούς) στο προσκήνιο. Αν στα παραπάνω προστεθεί και η νοοτροπία που ως επί το πλείστον επικρατούσε (και σε ένα βαθμό εξακολουθεί να επικρατεί, αλλά εδώ θα επανέλθουμε αργότερα) στην ανάπτυξη των εφαρμογών Λογισμικού, τότε οι ενέργειες που πρέπει να γίνουν δέον είναι να δρομολογηθούν άμεσα.

Τεχνολογικά και όχι μόνο αίτια

Να αναφέρουμε στο σημείο αυτό ότι παρόμοια προβλήματα έχουν εμφανιστεί και στο παρελθόν (όχι πριν από …1000 χρόνια) κυρίως στον τραπεζικό τομέα με την εφαρμογή νέων τρόπων υπολογισμού της αποπληρωμής δανείων, της αξίας ασφαλιστικών συμβολαίων, υποθηκών κλπ. Το νέο στην προκείμενη περίπτωση είναι ότι ενδεχομένως η αντιμετώπιση πρέπει να περιλαμβάνει εκτός των εφαρμογών Λογισμικού και το υλικό, καθώς και ότι επηρεάζεται εν δυνάμει πολύ μεγαλύτερο εύρος εφαρμογών Λογισμικού από τις τραπεζικές. Τεχνολογικά πάντως, τα πράγματα είναι αρκετά απλά: η παράσταση του έτους στις ημερομηνίες με δύο ψηφία αρκεί να μεταβληθεί ώστε να γίνεται με τέσσερα ώστε να περιλαμβάνεται και ο αιώνας και το πρόβλημα λύνεται. Η απόσταση της απλής αυτής σκέψης από την υλοποίηση μεγαλώνει εκθετικά αν λάβουμε υπόψη μας τα παρακάτω:

Το πρώτο είναι η παλαιότητα του υλικού των ίδιων των υπολογιστικών συστημάτων η οποία αγγίζει και σε αρκετές περιπτώσεις ξεπερνά την εικοσαετία. Τα συστήματα αυτά, ενίοτε υδρόψυκτα, έχουν αποκλειστικά «δικά τους» (δηλαδή της κατασκευάστριας εταιρίας) χαρακτηριστικά, όπως αρχιτεκτονική, επεξεργαστές, κλπ υλικό, αλλά και λειτουργικό σύστημα και λογισμικό εφαρμογών. Η παράσταση ενός μεγέθους με τέσσερα αντί των 2 ψηφίων σήμερα ακούγεται θέμα ανάξιο συζήτησης, τότε όμως, που «κεντρικά συστήματα» λειτουργούσαν με 1 MB κεντρικής μνήμης και που χωρητικότητα δίσκου 2 GB ήταν μόνο για τις «πραγματικά μεγάλες υπολογιστικές εγκαταστάσεις», τα πράγματα ήταν διαφορετικά και όλα έπρεπε να μετρηθούν σχολαστικά. Η παράσταση, λοιπόν, της ημερομηνίας στο υλικό και στο λειτουργικό σύστημα με τη μεγαλύτερη δυνατή οικονομία χώρου, ήταν λιγότερο αβλεψία και περισσότερο δείγμα του ότι, παρά το μεγάλο κόστος τους, τα συστήματα εκείνα δεν μπορούσαν να περάσουν «μια ζωή» στο μηχανογραφικό τμήμα. Στην πράξη όμως, και για μια σειρά αιτιών που θα προσπαθήσουμε να εξετάσουμε στη συνέχεια, τράπεζες, ασφαλιστικοί και χρηματοπιστωτικοί οργανισμοί, κυβερνήσεις, στρατιωτικά επιτελεία και πλήθος οργανισμών και εταιριών διατήρησαν και διατηρούν τέτοια συστήματα σε λειτουργία.

Το δεύτερο έχει να κάνει με το λογισμικό εφαρμογών, όπου τα πράγματα είναι κατά πολύ χειρότερα διότι η εντελώς απαραίτητη για την κατασκευή του υλικού τεκμηρίωση πάντων των σχετιζομένων με αυτό είναι από δαπανηρή πολυτέλεια μέχρι άγνωστη λέξη. Ακόμα και εκεί που υπάρχει κάποια τεκμηρίωση, αυτή είναι γραμμένη σύμφωνα με τον τρόπο που οι προγραμματιστές αντιλαμβάνονται τις κατευθύνσεις που δίνει ο κατασκευαστής της γλώσσας, μέσα στα εκάστοτε όρια χρονικών και οικονομικών περιορισμών και πάντως χωρίς η συμμόρφωση με τα όποια πρότυπα να αποτελεί προτεραιότητα. Τα ίδια τα πρότυπα εξάλλου

ήρθαν τουλάχιστον μια δεκαετία αργότερα και άρχισαν να γίνονται διεθνώς αποδεκτά και γνωστά στη δεκαετία του 80.

Οι προγραμματιστές, λοιπόν, γράφουν σε μια γλώσσα που είναι ιδιοκτησία του κατασκευαστή της μηχανής στην οποία αυτή τρέχει, με τα εργαλεία που μόνο αυτός διαθέτει. Η πρώτη προσπάθεια δημιουργίας μιας γλώσσας προγραμματισμού για χρήση σε εμπορικές εφαρμογές η οποία να τρέχει σε όσο το δυνατόν περισσότερα συστήματα, είναι η Cobol. Σήμερα μπορεί να την κατηγορούμε, λίγοι όμως αμφισβητούν ότι η εντυπωσιακή διάδοσή της οφείλεται ακριβώς στο ότι ήταν η πρώτη γλώσσα που το πέτυχε, η δε βιωσιμότητά της οφείλεται επίσης στο ότι τα συστήματα που την δέχτηκαν παρέμειναν σε χρήση για τόσο μεγάλο χρονικό διάστημα – κανείς δεν τολμούσε να πειράξει κάτι που δούλευε.

Πώς όμως παριστάνονται οι ημερομηνίες; Καμία σχεδόν γλώσσα της εποχής δεν έχει τύπο δεδομένων DATE και ακόμα και αν έχει, δεν μπορεί να υποχρεώσει τους προγραμματιστές να τον χρησιμοποιούν. Μια ημερομηνία μπορεί να ορίζεται και να χρησιμοποιείται όπως βολεύει τον προγραμματιστή. Τίποτε δεν τον περιορίζει, η χρήση παραστατικών ονομάτων μεταβλητών και σχολίων, η καλή δόμηση των μονάδων προγράμματος που πραγματοποιούν υπολογισμούς, καθώς και η με οποιονδήποτε τρόπο τεκμηρίωση σε χαρτί είναι όλα προαιρετικά και επαφίενται στον …πατριωτισμό του κάθε προγραμματιστή ο οποίος καλείται κάτω από συνθήκες πίεσης να γράψει το πολυπόθητο πρόγραμμα που «κάνει την δουλειά», αδιάφορο (τελικά) το πώς. Κάποιος μπορεί να χρησιμοποιεί κλήσεις του λειτουργικού για τον χειρισμό ημερομηνιών, κάποιος όχι, ενώ υπάρχουν ασφαλώς και όλες οι ενδιάμεσες καταστάσεις ανάλογα με τα κέφια και την περίοδο.

Τα παραπάνω πολλαπλασιάζονται με την ενσωμάτωση μεταβολών επί μεταβολών στα προγράμματα κάθε φορά που ανακαλύπτεται κάποιο λάθος, που απαιτείται «μια εκτύπωση ακόμη», που η σχετική νομοθεσία αλλάζει, κλπ. Μεταβολών που δεν τεκμηριώνονται ή που η κατανόηση της τεκμηρίωσης τους από τρίτους προγραμματιστές αποτελεί άθλο. Τα πράγματα έφτασαν για πρώτη φορά στο «απροχώρητο» με την «κρίση του Λογισμικού» (software crisis) την δεκαετία του 80 η οποία κατά βάθος ποτέ δεν ξεπεράστηκε, απλά τονώθηκε από τις ενέσεις της τεχνολογικής προόδου στον τομέα.

Σήμερα όμως ο κόμπος έφτασε στο χτένι, και η νέμεσις ήρθε για να επιβάλλει τιμωρίες και μάλιστα έντοκες με υψηλό επιτόκιο. Η δυσκολία που εμπεριέχει το «πρόβλημα του 2000» δεν βρίσκεται στη διόρθωση των αριθμητικών υπολογισμών που έχουν σφάλμα, αλλά στον εντοπισμό τους μέσα σε εκατομμύρια γραμμές κακογραμμένου κώδικα που επί δεκαετίες αφέθηκε να «κάνει τη δουλειά του». Και μάλιστα η διόρθωση αυτή θα πρέπει να γίνει μέσα σε ένα τεράστιο και πολύχρωμο μωσαϊκό προγραμμάτων, χωρίς παρενέργειες, χωρίς τη βοήθεια τεκμηρίωσης που να μπορεί να εμπιστεύεται κανείς, χωρίς παραδοχές που να ισχύουν στο βαθμό που απαιτείται και κυρίως, χωρίς την πολυτέλεια του παραμικρού σφάλματος. Η δεύτερη όψη της δυσκολίας του προβλήματος είναι η διόρθωση όλων των αρχείων στις εγγραφές των οποίων το έτος της ημερομηνίας παριστάνεται με δύο ψηφία. Ακούγεται «εύκολο» αλλά δεν είναι αν αναλογιστεί κανείς ότι μεγάλο μέρος των αρχείων βρίσκεται off-line σε ταινίες η φόρτωση των οποίων απαιτεί δέσμευση της μηχανής η οποία όμως δεν γίνεται λόγω της κρισιμότητας των εφαρμογών που τρέχει.

Το τρίτο από τα αίτια του προβλήματος βρίσκεται, κατά την γνώμη μας, εκτός των computer rooms, δεν έχει να κάνει με εργαλεία, τεκμηρίωση, μεθοδολογίες κλπ. Αφορά αποκλειστικά με τον ανθρώπινο παράγοντα και συγκεκριμένα την αντίληψή του για την αλλαγή και το νέο ως απειλή τη θέσης του, αλλά και την αδράνεια να αντιμετωπίσει τα νέα «μέσα παραγωγής» του αιώνα με τον τρόπο που αρμόζει σε αυτά και όχι στις ατμομηχανές και τα εργοστάσια του

1800 – αντίληψη που πάντα στην ιστορία ήταν ορατή κατά την εξέλιξη των παραγωγικών μέσων και δυνάμεων.

Πριν την εποχή των υπολογιστών όλοι οι υπολογισμοί γίνονταν από τους Λογιστές και τους Αποθηκάριους υπό την επίβλεψη των Διευθυντών – managers της εποχής. Ολοι αυτοί είχαν κάθε λόγο να διατηρήσουν υπό τον έλεγχό τους την έλευση των νέων μέσων τα οποία απειλούσαν τη θέση τους. Επρεπε λοιπόν είτε η θέση αυτή να εξελιχθεί στις νέες συνθήκες σε κάτι διαφορετικό, είτε η νέα τεχνολογία και οι πρεσβευτές της να υποταγούν στις επιταγές τους. Για ένα μεγάλο διάστημα επελέγη και εφαρμόστηκε το πρώτο. Οι τεχνικοί ήταν για «να κάνουν αυτό που τους λέμε εμείς που ξέρουμε». Οι μεθοδολογίες για την ανάπτυξη του Λογισμικού, δηλαδή ενός κατασκευάσματος το οποίο στα μάτια τους έμοιαζε με δακτυλογραφημένο κείμενο, ήταν ακατανόητα «ψιλά γράμματα» τα οποία προσπαθούσαν να εγκαθιδρύσουν την απειλή απέναντι στους ρόλους τους. «Απόσβεση» και «αντικατάσταση» είναι έννοιες που ορίζονται μόνο για ότι είναι χειροπιαστό, εν προκειμένω για το hardware. Χαρακτηριστική η φράση που άκουσε ο υπογράφων εν έτει 1987 στο μηχανογραφικό τμήμα κάποιας τεχνικής εταιρίας ότι «ο υπολογιστής ονομάστηκε έτσι διότι είναι μια μηχανή που λειτουργεί υπό-τον-λογιστή»! Με τη βοήθεια της κεκτημένης από το παρελθόν «εμπιστοσύνης της διοίκησης της εταιρίας», οι φορείς της τεχνολογίας-απειλής περιορίζονταν «στα τεχνικά τους», οι αμοιβές τους συμπιέζονταν και σπάνια είχαν την αναγνώριση και το ρόλο που τους άξιζε. Αν σε όλα αυτά προσθέσουμε και την αδράνεια των εκπαιδευτικών συστημάτων να προσαρμοστούν στις νέες συνθήκες (OPEN Newsletter 12ος/97), φτάνουμε αβίαστα στη σημερινή κατάσταση.

Τεχνικές παρανοήσεις

Για να επανέλθουμε όμως στα τεχνικά, σκόπιμο είναι να αναφερθούμε σε κάποιες συνηθισμένες παρανοήσεις που αιωρούνται εξ' αιτίας εκούσιας είτε ακούσιας παραπληροφόρησης.

Παρανόηση 1: Το πρόβλημα θα προκύψει το 2000. Λάθος!. Το πρόβλημα, αν είναι να εμφανιστεί, θα προκύψει την πρώτη φορά που θα γίνει υπολογισμός με ημερομηνία μεγαλύτερη από 31.12.1999, ακόμα και αν αυτό γίνει αύριο το πρωί. Εχουν μάλιστα αναφερθεί περιπτώσεις όπου το πρόβλημα προκύπτει κατά τον χειρισμό ημερομηνιών από την 1.1.1999.

Παρανόηση 2: Το πρόβλημα αφορά τις μεγάλες εφαρμογές που τρέχουν σε mainframes. Λάθος! Στατιστικά το πρόβλημα παρουσιάζεται περισσότερο σε τέτοιες εφαρμογές. Αυτό δε σημαίνει ούτε ότι όλες οι εφαρμογές αυτές έχουν υποχρεωτικά πρόβλημα, ούτε ότι οι υπόλοιπες δεν έχουν καθόλου.

Παρανόηση 3: Αν ο τρόπος παράστασης των ημερομηνιών στο λειτουργικό μου σύστημα έχει/δεν έχει το πρόβλημα, τότε και οι εφαρμογές μου το έχουν/δεν το έχουν αντίστοιχα. Λάθος! Αυτό ισχύει μόνο στις περιπτώσεις που είμαστε σίγουροι ότι ο χειρισμός των ημερομηνιών στις εφαρμογές γίνεται αποκλειστικά μέσω των κλήσεων του λειτουργικού συστήματος. Αν υπάρχει οποιαδήποτε αμφιβολία περί αυτού, πρέπει να γίνει έλεγχος.

Παρανόηση 4: Το υπολογιστικό μου σύστημα είναι τελευταία τεχνολογίας, οπότε δεν έχω πρόβλημα. Σωστό μόνο αν και οι εφαρμογές λογισμικού που χρησιμοποιώ είναι και αυτές σύγχρονες και ο κατασκευαστής εγγυάται ότι συμπεριφέρονται σωστά στην αλλαγή της χιλιετίας.

Παρανόηση 5: Σε αυτή τη φάση είμαστε όμηροι των κατά δήλωσιν ικανών να διορθώσουν την κατάσταση. Με μία έννοια ναι. Ομως η κατάσταση δεν γίνεται να αγνοηθεί οπότε το βάρος πέφτει στη σωστή επιλογή των αναδόχων.

Παρανόηση 6: Αν τα κατασκευάσουμε όλα από την αρχή αντικαθιστώντας με την ευκαιρία τις πλατφόρμες υλικού και λογισμικού μας, τότε, ακόμη και με μεγαλύτερο κόστος θα λύσουμε σίγουρα το πρόβλημα. Πράγματι, το «πρόβλημα του 2000» θα λυθεί. Μόνο που στην κατασκευή του λογισμικού υπάρχει πάντα χώρος για σφάλματα, οπότε καλό είναι να σχεδιάζουμε πολύ προσεκτικά τις κινήσεις μας.

Παρανόηση 7: Το επόμενο αντίστοιχο πρόβλημα θα προκύψει μετά από …1000 χρόνια. Ουδέν σχόλιο.

Συμπερασματικά

Το πρόβλημα του 2000 ή όπως αλλιώς και αν ονομάσει κανείς την κατάσταση που περιγράφηκε και για την οποία ακούγονται πολλά, έχει τις ρίζες του στις τεχνικές αλλά και στις διοικητικές επιλογές για την ανάπτυξη κρίσιμων εφαρμογών λογισμικού σε μια εποχή που όλα τα σχετικά με την πληροφορική βρίσκονταν εν τη γεννέσει τους. Η αντιμετώπιση του υλικού αλλά κυρίως του λογισμικού των μεγάλων συστημάτων σαν δεύτερης κατηγορίας εργαλείο παραγωγής του οποίου η κατασκευή και η συντήρηση πρέπει να γίνεται με το ελάχιστο κόστος, η δε αντικατάσταση «ποτέ», συσσώρευσε επί δεκαετίες προβλήματα των οποίων η σοβαρότητα υποτιμείτο ως η κορυφή ενός παγόβουνου του οποίου το πραγματικό μέγεθος αναγκαστήκαμε να συνειδητοποιήσουμε όταν τα πράγματα «δεν πήγαιναν άλλο».

Η τεχνική πλευρά του θέματος είναι απλή, δύσκολος είναι ο χειρισμός της αντιμετώπισης από διοικητικής πλευράς, θέμα στο οποίο αναφέρεται άλλο άρθρο του φύλλου που κρατάτε. Αυτή τη φορά πάντως η γνώμη των καταρτισμένων επιστημόνων τεχνικών συμβούλων και γενικά ο ρόλος τους οφείλει να τύχει καλύτερης αντιμετώπισης από τους managers η νοοτροπία των προγόνων των οποίων δημιούργησε το πρόβλημα. Καθότι, όπως θα έλεγαν και οι αρχαίοι ημών πρόγονοι, «το δις εξαμαρτείν, ουκ ανδρός σοφού».

Δημοσιεύτηκε στο OPEN Newsletter

τον Φεβρουάριο του 1998