The Cathedral and the Bazaar

by Eric S. Raymond

$Date: 1998/08/11 20:27:29 $

Στο παρόν αναλύω ένα επιτυχημένο project ανοιχτού κώδικα (open source), το fechmail. Αυτό ήταν που έπαιξε το ρόλο ενός εσκεμμένου τεστ μερικών εκπληκτικών θεωριών γύρω από την τεχνολογία λογισμικού (software engineering) όπως υποδείχτηκαν από την ιστορία του Linux. Πραγματεύομαι αυτές τις θεωρίες με τους όρους δυο θεμελιακά διαφορετικών στυλ ανάπτυξης, το Καθεδρικό μοντέλο (cathedral) υιοθετημένο από το μεγαλύτερο κομμάτι του εμπορικού κόσμου εναντίον του Παζαριώτικου μοντέλου (bazaar) του κόσμου του Linux. Δείχνω ότι αυτά τα μοντέλα κατευθύνονται από αντίθετες υποθέσεις για την φύση της διαδικασίας αποσφαλμάτωσης λογισμικού. Στην συνέχεια δημιουργώ ένα υποστηρικτικό επιχείρημα από την εμπειρία του Linux για την υπόθεση "έχοντας αρκετά μάτια, όλα τα bugs είναι ρηχά", προτείνω παραγωγικές αναλογίες με άλλα αυτό-διορθώσιμα συστήματα από εγωιστικούς συντελεστές και κλείνω με εξερεύνηση των συνεπειών αυτής της διόρασης για το μέλλον του λογισμικού.

Contents

1 Ο Καθεδρικός και το Παζάρι.

Το Linux είναι ανατρεπτικό. Ποιος θα σκεφτόταν, ακόμη και πριν από πέντε χρόνια , ότι ένα λειτουργικό σύστημα παγκόσμιας εμβέλειας θα φτιαχνόταν, σαν από μαγεία, από τμηματικά hacking που έκαναν χιλιάδες προγραμματιστές σκορπισμένοι ς' όλο τον πλανήτη, ενωμένοι μόνο με τα αδύναμα νήματα του Internet;

Κανείς, βέβαια. Την ώρα που μάθαινα για το Linux στις αρχές του 1993, ασχολούμουνα ήδη με το Unix και τον προγραμματισμό ανοιχτού κώδικα [open source]επί δέκα χρόνια. Ήμουν από τους πρώτους διανεμητές της GNU στα μέσα της δεκαετίας του '80. Είχα κατασκευάσει και διαθέσει έναν σεβαστό αριθμό λογισμικού ανοιχτού κώδικα στο δίκτυο, αναπτύσσοντας ή συναναπτύσσοντας πολλά προγράμματα (nethack, Emacs VC και GUD modes, xlife και άλλα) που παραμένουν σε ευρεία χρήση έως σήμερα. Νόμιζα πως ήξερα πώς είχαν γίνει όλ' αυτά.

Το Linux ανέτρεψε πολλά απ' αυτά που νόμισα ότι ήξερα. Υποστήριζα το "Ευαγγέλιο" των μικρών εργαλείων του Unix, την γρήγορη ανάπτυξη πρωτοτύπων και τον εξελικτικό προγραμματισμό [evolutionary programming] για χρόνια. Πίστευα, όμως, ότι υπάρχει μια συγκεκριμένη κρίσιμη πολυπλοκότητα πάνω από την οποία απαιτούνταν μια περισσότερο κεντρική, απριόρι προσέγγιση. Πίστευα ότι τα πιο σημαντικό λογισμικό (λειτουργικά συστήματα και πραγματικά μεγάλα εργαλεία όπως το Emacs) έπρεπε να χτιστούν σαν καθεδρικοί ναοί, προσεχτικά φτιαγμένοι από μεμονωμένους ειδικούς [wizards] ή μικρές ομάδες από "μάγους" [mages] που να δουλεύουν σε απόλυτη απομόνωση, χωρίς να δημοσιεύονται οι beta πριν την ώρα τους.

Το στυλ προγραμματισμού του Linus Torvalds- πρώιμες και ανά μικρά διαστήματα εκδόσεις λογισμικού, μεταβιβάσεις του κάθε τι, ανοχή στο ζήτημα της ετερογενούς μεγάλης ανάμειξης - ήταν έκπληξη. Δεν έμοιαζε με θρησκευτικό καθεδρικό ναό - η κοινότητα του Linux έμοιαζε περισσότερο με ένα μεγάλο φλύαρο παζάρι διαφορετικών πρακτικών και προσεγγίσεων (που συμβολίζονται με αρχειοθήκες [sites] λογισμικού για Linux, στα οποία μπορεί να συνεισφέρει ο *καθένας*) από το οποίο ένα συνεπές και σταθερό σύστημα μπορούσε να φτιαχτεί μόνο μετά από μια ακολουθία θαυμάτων.

Το γεγονός ότι αυτό το παζάρι έδειχνε να δουλεύει και να δουλεύει καλά, ήταν ένα ξεκάθαρο σοκ. Καθώς ο καιρός περνούσε κι εγώ μάθαινα, δούλευα σκληρά όχι μόνο σε συγκεκριμένα projects, αλλά και προσπαθώντας να καταλάβω γιατί ο κόσμος του Linux όχι μόνο δεν έπεφτε σε σύγχυση αλλά έδειχνε να δυναμώνει συνεχώς, με μια ταχύτητα που δεν την φαντάζονταν οι αρχιτέκτονες καθεδρικών ναών.

Στα μέσα του 1996 άρχισα να καταλαβαίνω. Μου δόθηκε η ευκαιρία να δοκιμάσω την θεωρία μου με την μορφή ενός project ανοιχτού κώδικα που θα προσπαθούσα να θέσω σε λειτουργία με το στυλ του παζαριού. Έτσι κι έκανα - και η επιτυχία ήταν μεγάλη.

Στο υπόλοιπο μέρος αυτού του άρθρου θα αφηγηθώ την ιστορία αυτού του project και θα το χρησιμοποιήσω για να προτείνω κάποιες θέσεις για την αποτελεσματική ανάπτυξη προγραμμάτων ανοιχτού κώδικα. Πολλά από αυτά δεν τα πρωτοέμαθα μέσα από το κόσμο του Linux, αλλά θα δούμε πώς αυτός τους δίνει ένα ιδιαίτερο νόημα. Αν είμαι σωστός στις απόψεις μου, θα σας βοηθήσουν να καταλάβετε τι ακριβώς είναι αυτό που κάνει την κοινότητα του Linux μια πηγή τόσο καλού λογισμικού-και θα σας βοηθήσει να γίνεται και εσείς οι ίδιοι πιο παραγωγικοί._

2 Το Μήνυμα Πρέπει να Περάσει.

Από το 1993 έχω την ευθύνη της τεχνικής πλευράς ενός μικρού Παροχέα ελεύθερης πρόσβασης, που ονομάζεται Chester County InterLink (CCIL) στο Δυτικό Τσέστερ της Πενσυλβάνια (είμαι συνιδρυτής του CCIL και κατασκεύασα το δικό μας μοναδικό πολυχρηστικό bulletin-board software –μπορείτε να το δείτε κάνοντας telnet στο *locke.ccil.org* `<telnet://locke.ccil.org>`. Σήμερα υποστηρίζει σχεδόν τρεις χιλιάδες χρήστες σε τριάντα γραμμές). Αυτή η δουλειά μου επέτρεπε ελεύθερη πρόσβαση στο δίκτυο επί 24ώρου βάσεως μέσω της 56K γραμμής του CCIL. in fact, it practically demanded it!

Επομένως, ήμουν μαθημένος στην χρήση του άμεσου ηλεκτρονικού ταχυδρομείου. Για κάποιους περίπλοκους λόγους ήταν δύσκολο να συνδέσω με SLIP τον υπολογιστή που έχω στο σπίτι μου (snark.thyrsus.com) και τον CCIL. Όταν, τελικά ,τα κατάφερα ανακάλυψα ότι πρέπει να συνδέομαι (telnet) περιοδικά στον "locke" για να παραλαμβάνω την αλληλογραφία μου. Αυτό που ήθελα για την αλληλογραφία μου ήταν να παραδίδεται στον snark έτσι ώστε να μπορώ να ειδοποιούμε όταν φτάνει και να μπορώ να την χειριστώ χρησιμοποιώντας όλα τα τοπικά εργαλεία μου.

Η απλή προώθηση μηνυμάτων με το sendmail δεν δούλευε, επειδή ο προσωπικός υπολογιστής μου δεν είναι συνεχώς στο δίκτυο και δεν έχει στατική διεύθυνση IP. Αυτο που ήθελα ήταν ένα πρόγραμμα που θα έπαιρνε τον έλεγχο πάνω από την SLIP σύνδεση μου και θα μετέφερε την αλληλογραφία μου να την παραλάβω τοπικά. Ήξερα ότι τέτοια προγράμματα υπήρχαν και ότι τα περισσότερα χρησιμοποιούσαν ένα απλό πρωτόκολλο που λέγεται POP(Post Office Protocol). Και ήμουν αρκετά σίγουρος ότι υπήρχε ένας POP 3 σέρβερ που συμπεριλαμβάνεται στο BSD/OS λειτουργικό που βρισκόταν στον "locke".

Χρειαζόμουν έναν POP 3 client. Έτσι, συνδέθηκα στο δίκτυο και βρήκα έναν. Στην πραγματικότητα Βρήκα τρεις τέσσερις. Χρησιμοποίησα pop-perl για λίγο, αλλά έλλειπε μια σημαντική δυνατότητα, Η δυνατότητα να μετατρέπονται οι διευθύνσεις των παραλαμβανόμενων μηνυμάτων ώστε η δυνατότητα απάντησης [reply] να δουλεύει σωστά.

Ιδού το πρόβλημα: υποθέστε ότι κάποιος που λέγεται "joe" στον "locke" μου έστειλε ένα μήνυμα [mail] Αν παραλάβω το μήνυμα στον snark και μετά επιχειρήσω να απαντήσω, το πρόγραμμα ηλεκτρονικού ταχυδρομείου θα προσπαθήσει να το στείλει ς' έναν "joe" που δεν υπάρχει στον snark. Η τακτική μετατροπής με το σερί των διευθύνσεων προς απάντηση [Reply addresses] στον "@ccil.org" γρήγορα αποδείχθηκε πολύ δύσκολη.

Ήταν κάτι που ο υπολογιστής έπρεπε οπωσδήποτε να κάνει για μένα. Κανένας, όμως, από τους υπάρχοντες POP clients δεν ήξερε πώς! Κι ερχόμαστε στο πρώτο μας μάθημα:

1) Κάθε καλή δουλειά στον χώρο του λογισμικό αρχίζει με την προσωπική φαγούρα του προγραμματιστή.

Ίσως αυτό θα έπρεπε να είναι ολοφάνερο (από παλιά έχει ειπωθεί, "Η ανάγκη είναι η μητέρα της εφεύρεσης") αλλά, πολύ συχνά, οι προγραμματιστές ξοδεύουν τις μέρες τους πασχίζοντας για ένα μεροκάματο φτιάχνοντας εφαρμογές που ούτε χρειάζονται ούτε τους ενδιαφέρουν. Όχι, όμως, και στον κόσμο του Linux–πράγμα που εξηγεί γιατί η μέση ποιότητα του λογισμικού για Linux είναι τόσο υψηλή.

Τι νομίζετε, λοιπόν; Ότι βούτηξα στην δίνη της κωδικοποίησης ενός ολοκαίνουργιου POP 3 client για να ανταγωνιστεί τους υπάρχοντες; Με τίποτα στον κόσμο! Έριξα μια προσεκτική ματιά στα POP utilities που είχα στα χέρια μου, και αναρωτήθηκα "ποιό απ' όλα είναι πλησιέστερο σε αυτό που θέλω;".

2) Οι καλοί προγραμματιστές ξέρουν τι να γράψουν. Οι σπουδαίοι ξέρουν τι να ξαναγράψουν (και να ξαναχρησιμοποιήσουν).

Αν και δεν ισχυρίζομαι ότι είμαι σπουδαίος προγραμματιστής, προσπάθησα να μοιάσω τέτοιος. Ένα σημαντικό γνώρισμα των σπουδαίων προγραμματιστών είναι η εγγενής τους τεμπελιά. Ξέρουν ότι χρειάζεται ένα πρόγραμμα όχι για να περνάνε οι χρήστες την ώρα τους αλλά για να έχουν κάποια αποτελέσματα και ότι είναι σχεδόν πάντα ευκολότερο να ξεκινάς από μια μερική λύση από το να ξεκινάς από το μηδέν.

Ο *Linus Torvalds* `<http://www.earthspace.net/~esr/faqs/linus>`, για παράδειγμα, δεν έγραψε το Linux απ' την αρχή. Αντίθετα ,ξεκίνησε να ξαναχρησιμοποιεί κώδικα και ιδέες απ' το Minix, ένα μικρό Unix-οϊδές λειτουργικό για μηχανές 386. Τελικά, ο κώδικας Minix απομακρύνθηκε ή ξαναγράφτηκε εντελώς–όσο, όμως, ήταν εκεί λειτουργούσε σαν η σκάλα για το βρέφος που, τελικά, έγινε το Linux.

Στο ίδιο πνεύμα, άρχισα να ψάχνω για ένα POP utility που θα είχε καλό κώδικα, για να μου χρησιμεύσει σαν βάση προγραμματισμού.

Η παράδοση ελεύθερου κώδικα του Unix ήταν πάντα φιλική προς την επαναχρησιμοποίηση κώδικα (γι' αυτό τον λόγο το GNU project επέλεξε το Unix σαν βασικό λειτουργικό, παρά τις σοβαρές νομικές επιφυλάξεις για το ίδιο το λειτουργικό). Η κοινότητα του Linux ώθησε αυτή την παράδοση σχεδόν στα τεχνολογικά της όρια. Προσφέρει σε όλους terabytes ελεύθερου κώδικα. Έτσι, το να αφιερώνεις λίγο χρόνο για να βρεις την κατάλληλη εφαρμογή κάποιου άλλου είναι περισσότερο πιθανό να δώσει θετικά αποτελέσματα στον κόσμο του Linux, παρά οπουδήποτε αλλού.

Έτσι έγινε και στην δική μου περίπτωση. Μαζί με αυτά που βρήκα νωρίτερα, η δεύτερη έρευνά μου κατέληξε μ' ένα σύνολο εννέα υποψήφιων–το fetchpop, το PopTart, το get-mail, το gwpop, το pop-perl, το pimp, το popc, το popmail και το upop. Αυτό που εγκατέστησα πρώτο ήταν το 'fetchpop' του Seung-Hong Oh. Έβαλα το χαρακτηριστικό header rewrite ς' αυτό κι έκανα και κάποιες άλλες βελτιώσεις τις οποίες ο δημιουργός του συμπεριέλαβε στην έκδοση 1.9.

Λίγες βδομάδες αργότερα, όμως, σκόνταψα στον κώδικα του 'popclient' του Carl Harris και ανακάλυψα ότι είχα πρόβλημα. Αν και το 'fetchpop' είχε μερικές καλές ιδέες στον κώδικά του (όπως την λειτουργία σε κατάσταση daemon), μπορούσε να διαχειριστεί μόνο POP 3 και ήταν ερασιτεχνικά κωδικοποιημένο (ο Seung-Hong Oh ήταν ένας έξυπνος αλλά άπειρος προγραμματιστής και τα δύο αυτά χαρακτηριστικά του εκδηλώθηκαν στο πρόγραμμα αυτό). Ο κώδικας του Carl ήταν καλύτερος, αληθινά επαγγελματικός και σταθερός, αλλά από το πρόγραμμά του έλειπαν πολλά σημαντικά και μάλλον δύσκολα στην υλοποίησή τους χαρακτηριστικά που ήδη είχε το fetchpop (συμπεριλαμβανομένων κι αυτών που κωδικοποίησα εγώ).

Να τα παρατήσω ή να επιμείνω; Αν τα παρατούσα, θα έπρεπε να πετάξω τον κώδικα που εί χα ήδη φτιάξει σε αντάλλαγμα μιας καλύτερης προγραμματιστικής βάσης.

Ένα πρακτικό κίνητρο για να τα παρατήσω ήταν η παρουσία υποστήριξης πολλαπλών πρωτοκόλλων. Από τα πρωτόκολλα που χρησιμοποιούν οι διαμετακομιστές ταχυδρομείου το POP 3 ήταν το πιο κοινό, αλλά όχι το μοναδικό. Το fetchpop και οι άλλοι ανταγωνιστές του δεν υποστήριζαν POP 2, RPOP ή APOP και σκεφτόμουν, ήδη, να προσθέσω IMAP (I nternet Message Access Protocol, το πιο καινούριο και πιο ισχυρό πρωτόκολλο ταχυδρομείου) *IMAP* `<http://www.imap.org>`, μόνο και μόνο για την ευχαρίστηση μου.

Όμως, είχα έναν πιο θεωρητικό λόγο για να πιστεύω ότι, το να τα παρατήσω θα ήταν καλή ιδέα, κάτι που έμαθα πολύ πριν το Linux.

3)"Σχεδιάζεις να απορρίψεις κάποιο πρόγραμμα; Θα το κάνεις, ούτως ή άλλως". (Fred Books, "The Mythical Man-Month", chapter 11)

Ή, για να το πω αλλιώς, συνήθως δεν καταλαβαίνεις το πρόβλημα μέχρι τη στιγμή που υλοποιείς μια λύση. Την δεύτερη φορά ίσως ξέρεις περισσότερα για να πράξεις σωστότερα. Έτσι, αν θέλεις να είσαι σωστός, ετοιμάσου να αρχίσεις απ' την αρχή *τουλάχιστον* μια φορά.

Λοιπόν (είπα στον εαυτό μου) οι αλλαγές που έκανα στο fetchpop ήταν η πρώτη προσπάθεια. Έτσι, τα παράτησα.

Μετά την παράδοση του πρώτου συνόλου διορθωτικών πακέτων κώδικα για το popclient [popclient paches] που έστειλα στον Carl Harris στις 25 Ιουνίου 1996, ανακάλυψα ότι εί χε χάσει το ενδιαφέρον του για το popclient λίγο καιρό πριν. Ο κώδικας ήταν λίγο σκόρπιος, με μικρά bugs εδώ κι εκεί. Είχα πολλές αλλαγές να κάνω και κατέληξα γρήγορα στο ότι το πιο λογικό πράγμα που έπρεπε να κάνω ήταν να αναλάβω το πρόγραμμα.

Σχεδόν χωρίς να το προσέξω, το project κλιμακώθηκε. Δεν θα καταπιανόμουν άλλο με ασήμαντα διορθωτικά πακέτα [patches] για τους υπάρχοντες POP clients. Ξεκίνησα να δουλεύω πάνω ς' έναν απ' αυτούς και οι ιδέες στριμώχνονταν μέσα στο μυαλό μου και ήξερα ότι, πιθανόν, να οδηγήσουν σε μεγάλες αλλαγές.

Σε μια κουλτούρα λογισμικού που ενθαρρύνει την από κοινού κωδικοποίηση [code-sharing], αυτός είναι ένας φυσικός τρόπος ανάπτυξης ενός σχεδίου. Ενεργούσα ως εξής:

4) Αν η συμπεριφορά σου είναι σωστή, θα συναντήσεις ενδιαφέροντα προβλήματα.

Η συμπεριφορά του Carl Harris, όμως, ήταν ακόμη πιο σημαντική. Αντιλήφθηκε ότι

5) Όταν ένα πρόγραμμα παύει να ς' ενδιαφέρει, το τελευταίο σου καθήκον είναι να το παραδώσεις ς' έναν ικανό διάδοχο.

Χωρίς ποτέ να χρειαστεί να το συζητήσουμε, ο Carl κι εγώ ξέραμε ότι είχαμε τον κοινό στόχο να δώσουμε τις καλύτερες λύσεις. Το μόνο ερώτημα και των δύο, ήταν αν ήμουν κατάλληλος γι' αυτή την δουλειά. Μια φορά που του έδειξα ότι ήμουν, αντέδρασε γενναιόδωρα. Ήλπιζα να αντιδράσω κι εγώ έτσι, όταν θα ερχόταν η σειρά μου.

3 Η Σπουδαιότητα του να Έχεις Χρήστες.

Έτσι, κληρονόμησα το popclient. Το ίδιο σημαντικό ήταν ότι κληρονόμησα και τους χρήστες του. Είναι πολύ καλό να έχεις χρήστες κι όχι μόνο επειδή δηλώνουν πως ικανοποιείς μια ανάγκη τους, ή ότι κάνεις κάτι σωστό. Αν αντιμετωπισθούν κατάλληλα, μπορούν να γίνουν συνεργάτες στην ανάπτυξη της εφαρμογής.

Μια άλλη δύναμη της παράδοσης του Unix, που το Linux ωθεί στα όριά της, είναι ότι πολλοί χρήστες είναι επίσης και hackers. Επειδή ο πηγαίος κώδικας είναι ελεύθερος, μπορούν να γίνουν *αποτελεσματικοί* hackers. Κάτι τέτοιο μπορεί να αποδειχθεί εξαιρετικά χρήσιμο για την μείωση του χρόνου αποσφαλμάτωσης [debugging]. Με λίγη ενθάρρυνση, οι χρήστες σας θα διαγνώσουν προβλήματα, θα προτείνουν διορθώσεις και θα βοηθήσουν στην βελτίωση του κώδικα, πολύ πιο γρήγορα από το να το κάνατε μόνος σας χωρίς βοήθεια.

6) Η αντιμετώπιση των χρηστών σαν συνεργάτες προγραμματιστές είναι ο λιγότερο επικίνδυνος δρόμος προς την γρήγορη βελτίωση του κώδικα και την αποτελεσματική αποσφαλμάτωση.

Η δύναμη αυτής της άποψης εύκολα μπορεί να υποεκτιμηθεί. Στην πραγματικότητα, πολλοί από εμάς στον κόσμο του ανοιχτού κώδικα υποεκτιμούμε σε δραματικό βαθμό το πόσο καλύτερα θα αυξανόμασταν σε αριθμό χρηστών κι ενάντια στην πολυπλοκότητα των συστημάτων, μέχρι ότου ο Linus Torvalds μας έδειξε ότι τα πράγματα μπορούν να είναι και διαφορετικά.

Νομίζω ότι το πιο έξυπνο και αποτελεσματικό κατόρθωμα [hack] του Linus, δεν ήταν η κατασκευή του ίδιου του πυρήνα του Linux αλλά, μάλλον, η εφεύρεση του μοντέλου ανάπτυξης Linux. Όταν, παρουσία του, εξέφρασα αυτή την άποψη εκείνος χαμογέλασε και σιγανά επανέλαβε κάτι που συχνά έλεγε: "Βασικά, είμαι πολύ τεμπέλης άνθρωπος που του αρέσει να του αναγνωρίζουν πράγματα που άλλοι άνθρωποι έκαναν". Τεμπέλης σαν γάτος. Ή, όπως θα έλεγε ο Robert Heinlein, πολύ τεμπέλης για να αποτύχει.

Εκ των υστέρων, ένα προηγούμενο της μεθόδου κι επιτυχίας του Linux μπορεί να θεωρηθεί η ανάπτυξη της βιβλιοθήκης GNU Emacs Lisp και των αρχείων κώδικα Lisp [Lisp code archives]. Σε αντίθεση με το καθεδρικό-μεγαλειώδες στυλ του C πυρήνα του Emacs και των περισσότερων υπόλοιπων εργαλείων FSF, η εξέλιξη του κώδικα της Lisp ήταν ρευστή και εξαρτημένη από τους χρήστες. Διάφορες ιδέες και λειτουργίες πρωτοτύπων συχνά ξαναγράφονταν τρεις ή τέσσερις φορές πριν καταλήξουν σε μια σταθερή μορφή. Και οι χαλαρές συνεργασίες, μέσω του Internet, αλα Linux, ήσαν συχνές.

Πραγματικά, το πιο επιτυχές προσωπικό κατόρθωμα μού, πριν από το fetchmail, ήταν ίσως το Emacs VC mode, μια Linux-οϊδής συνεργασία μέσω ηλεκτρονικού ταχυδρομείου τριών ανθρώπων, εκ των οποίων μόνο έναν (τον Richard Stallman, τον κατασκευαστή του Emacs και ιδρυτή του *FSF* `<http://www.fsf.org>`) γνώριζα ως εκείνη τη μέρα. Ήταν μια λειτουργία front-end για SCCS, RCS και, αργότερα, CVS μέσα από το Emacs, που επέτρεπε χειρισμούς ελέγχου έκδοσης "με ένα άγγιγμα" ["one-touch"]. Αναπτύχθηκε από μια μικρή, άχαρη λειτουργία sccs.el, που είχε γράψει κάποιος άλλος. Και η ανάπτυξη του VC πέτυχε, επειδή, αντίθετα με το ίδιο το Emacs, ο κώδικας του Emacs Lisp μπορούσε να περάσει γρήγορα μέσα από διαδικασίες έκδοσης / δοκιμής / βελτίωσης.

Ένα αναπάντεχο, δευτερεύων αποτέλεσμα της πολιτικής του FSF για νόμιμη σύνδεση κώδικα μέσα στο GPL ήταν ότι, γινόταν διαδικαστικά όλο και πιο δύσκολο για την FSF να χρησιμοποιήσει την χαλαρή / παζαριώτικη λειτουργία, αφού οι άνθρωποι της FSF πίστευαν ότι πρέπει να αποκτήσουν copyright εφαρμογής για κάθε ξεχωριστή διανομή περισσότερων από είκοσι γραμμών κώδικα, ώστε να προστατέψουν τον κώδικα GPL από τους νόμους για το copyright. Όσοι άσκησαν πνευματικά δικαιώματα χρησιμοποιώντας τις άδειες των κονσόρτσια BSD και MIT X δεν έχουν τέτοια προβλήματα. Δεν προσπαθούν να εξασφαλίσουν δικαιώματα, που ο καθένας θα μπορούσε να έχει κίνητρα να προκαλέσει.

4 Εκδόσεις Νωρίς, Εκδόσεις Συχνά.

Οι πρώιμες και ανά μικρά χρονικά διαστήματα εκδόσεις είναι κρίσιμο μέρος του μοντέλου ανάπτυξης του Linux. Οι περισσότεροι προγραμματιστές (συμπεριλαμβανόμενου και εμένα) πίστευαν ότι αυτό είναι μια κακή πολιτική για μεγαλύτερα από τα τετριμμένα projects, επειδή οι πρώιμες εκδόσεις είναι σχεδόν εξ ορισμού γεμάτες bugs και το μόνο που δεν επιθυμείς είναι να εξαντλείς την υπομονή των χρηστών σου.

Αυτή η άποψη ενίσχυε την γενική αποδοχή ενός "καθεδρικού" είδους ανάπτυξης. Αν ο κύριος στόχος είναι να συναντούν οι χρήστες όσο το δυνατόν λιγότερα bugs, τότε γιατί να εκδίδεις μια φορά κάθε έξη μήνες (ή και λιγότερο συχνά) και μετά να δουλεύεις σκληρά (like a dog) για να απαλλαχθείς απ' τα bugs; Ο πυρήνας του Emacs C φτιάχτηκε μ' αυτό τον τρόπο. Ενώ αντίθετα, η βιβλιοθήκη Lisp όχι, αυτό συνέβη επειδή υπήρχαν ενεργά αρχεία Lisp έξω απ' τον έλεγχο του FSF, στα οποία μπορούσε κανείς να ανατρέξει για να βρει νέες εκδόσεις του κώδικα, ανεξάρτητα απ' τους κύκλους εκδόσεων του Emacs.

Η πιο σημαντική απ' αυτές τις αρχειοθήκες, η elisp του Οχάιο, προέβλεπε το πνεύμα και πολλά απ' τα χαρακτηριστικά των σημερινών μεγάλων αρχειοθηκών του Linux. Αλλά λίγοι από μας σκέφτονταν πραγματικά πολύ για το τι κάναμε, ή για το τι σήμαινε η ύπαρξη αυτών των αρχειοθηκών για το πρόβλημα του "καθεδρικού" μοντέλου ανάπτυξης του FSF. Έκανα μια σοβαρή προσπάθεια γύρω στο 1992 να προσαρτήσω μεγάλο μέρος του κώδικα του Οχάιο στην επίσημη βιβλιοθήκη του Emacs Lisp. Μπήκα σε πολιτικούς μπελάδες και η αποτυχία μου ήταν μεγάλη.

Όμως, ένα χρόνο μετά, καθώς το Linux γινόταν ευρύτερα γνωστό, έγινε ξεκάθαρο ότι κάτι διαφορετικό και υγιέστερο συμβαίνει. Το ανοιχτό μοντέλου ανάπτυξης του Linus ήταν το άκρως αντίθετο του "καθεδρικού" μοντέλου. Έκαναν την εμφάνισή τους οι αρχειοθήκες sunsite και tsx-11 καθώς και πολλαπλές διανομές. Κι όλ' αυτά οδηγούμενα από μια αναπάντεχη συχνότητα εκδόσεων του πυρήνα του συστήματος.

Ο Linus αντιμετώπιζε τους χρήστες του σαν συν-προγραμματιστές με τον καλύτερο δυνατό τρόπο:

7.Εκδόσεις νωρίς, εκδόσεις συχνά και άκου τους χρήστες σου.

Η καινοτομία του Linus δεν ήταν τόσο αυτό (κάτι τέτοιο συνέβαινε για πολύ καιρό στον κόσμο του Unix), αλλά η κλιμάκωσή του σε βαθμό που πλησίαζε την περιπλοκότητα της προς ανάπτυξη εφαρμογής. Εκείνη τη χρονιά (περί το 1992) δεν του ήταν άγνωστη η τακτική να εκδίδει νέο πυρήνα κάθε μέρα! Αυτή η προσπάθεια πέτυχε επειδή ανέπτυσσε την βάση των συν-προγραμματιστών του και χρησιμοποιούσε την δύναμη του Internet για συνεργασία πιο καλά από κάθε άλλον.

Αλλά *πώς* τα κατάφερε; Και ήταν κάτι που μπορούσα να μιμηθώ ή βασιζόταν σε κάποια μοναδική ευφυία του Linus Torvalds;

Ο Linus ήταν ένας πολύ καλός hacker (πόσοι από εμάς θα κατασκεύαζαν έναν παραγωγικά ποιοτικό πυρήνα λειτουργικά συστήματος;). Το Linux δεν ήταν κανένα τρομερό άλμα προς τα εμπρός. Ο Linus δεν είναι (ή όχι ακόμα) καμιά ιδιοφυία στο να σχεδιάζει όπως, ας πούμε, ο Richard Stallman ή ο James Gosling (NeWS και Java). Εμένα μου φαίνεται ότι ο Linus είναι ένας ευφυής προγραμματιστής, με μια έκτη αίσθηση να αποφεύγει τα bugs και προγραμματιστικά αδιέξοδα και εύκολα βρίσκει το πιο σύντομο δρόμο απ' το σημείο A προς το σημείο B. Πραγματικά, ολόκληρος ο σχεδιασμός του Linux αποπνέει αυτή την ποιότητα και αντανακλά την ουσιαστικά συντηρητική και απλοποιητική προσέγγιση του Linus.

Αν, λοιπόν, οι γρήγορες εκδόσεις και η πλήρης χρήση του Internet δεν ήσαν τυχαία γεγονότα αλλά ολοκληρωμένα μέρη της διορατικότητας του Linus, ποια ήταν η πρωτοτυπία του;

Για να το πω απλά, η ερώτηση απαντάται μόνη της. Ο Linus κρατούσε του hackers/χρήστες συνεχώς σε υπερένταση και τους αντάμειβε-σε υπερένταση μέσω του project ή διατηρώντας ένα εγωιστικό κομμάτι της όλης δράσης και τους αντάμειβε με την σταθερά (σχεδόν *κάθε μέρα*) βελτίωση της εργασία τους.

Ο Linus σκόπευε άμεσα στην μεγιστοποίηση του αριθμού ανθρωποωρών στην κατάργηση των bugs και στην ανάπτυξη, ακόμη και με πιθανό κόστος την αστάθεια του κώδικα ή την αχρήστευση της βάσης των χρηστών του, αν ένα σοβαρό bug αποδεικνυόταν ατίθασο. Ο Linus συμπεριφερόταν σαν να πίστευε στο εξής:

8. Δεδομένης μια μεγάλης βάσης δοκιμαστών beta και συν-προγραμματιστών, σχεδόν κάθε πρόβλημα γρήγορα θα εντοπισθεί κι ένα fix θα κάνει την εμφάνισή του.

Ή , λιγότερο τυπικά, "έχοντας αρκετά μάτια, όλα τα bugs είναι ρηχά". Το άλλαξα σε: "Ο Νόμος του Linus".

Η αρχική μου σκέψη ήταν ότι, κάθε πρόβλημα "θα γίνει φανερό σε κάποιον". Ο Linus απάντησε λέγοντας ότι, το πρόσωπο που καταλαβαίνει και διορθώνει το πρόβλημα δεν είναι αναγκαία ή συνήθως το πρόσωπο που πρώτο το χαρακτηρίζει. "Κάποιος βρίσκει ένα πρόβλημα", λέει, "και κάποιος *άλλος* το καταλαβαίνει. Και η ανακάλυψη του προβλήματος είναι η μεγαλύτερη πρόκληση". Το ζήτημα, όμως, ήταν ότι και τα δύο έτειναν να γίνονται ταχύτατα.

Εδώ, πιστεύω, ότι βρίσκεται η κεντρική διαφορά που υπογραμμίζει τα δύο στυλ προγραμματισμού, το καθεδρικό και το στυλ παζαριού. Από την άποψη του καθεδρικού προγραμματιστή, τα bugs και τα άλλα προβλήματα προγραμματισμού είναι απρόβλεπτα, ύπουλα, βαθιά φαινόμενα. Χρειάζονται μήνες λεπτομερούς εξέτασης από μερικούς αφοσιωμένους ανθρώπους για να κατασκευάσουν έμπιστο κώδικα. Γι' αυτό και τα μεγάλα διαλείμματα μεταξύ των εκδόσεων και η αναπόφευκτη απογοήτευση όταν οι εκδόσεις αυτές, που τόσο καιρό περιμένεις, δεν είναι τέλειες.

Απ' την άποψη του σε "στυλ παζαριού" προγραμματισμού, τα bugs είναι γενικά επιπόλαια φαινόμενα-ή, τουλάχιστον, καταλήγουν να γίνουν επιπόλαια όταν εκτίθενται σε χίλιους ανυπόμονους συν-προγραμματιστές που μελετούν κάθε νέα έκδοση. Έτσι, κάνεις συχνές εκδόσεις για να κερδίσεις περισσότερες διορθώσεις και σαν δευτερεύον κέρδος, έχεις λιγότερα να χάσεις αν παρουσιαστεί κάποια περιστασιακή τσαπατσουλιά

Αν ο "Νόμος του Linus" είναι εσφαλμένος, τότε κάθε σύστημα που είναι τόσο περίπλοκο όσο ο πυρήνας του Linux, που "μαστορεύεται " από τόσα χέρια, θα έπρεπε να καταρρέει κάτω από το βάρος των απρόβλεπτων κακών αλληλεπιδράσεων και καλά κρυμμένων bugs. Αν, από την άλλη είναι σωστός, εξηγεί ικανοποιητικά την σχετική απουσία bugs από το Linux.

Ως προς αυτό, δεν θα έπρεπε να αποτελεί έκπληξη. Οι κοινωνιολόγοι πριν από χρόνια ανακάλυψαν ότι η μέση γνώμη ενός συνόλου ισοδύναμα ειδικών (ή ισοδύναμα ανίδεων) παρατηρητών είναι λίγο περισσότερο αξιόπιστη από εκείνη ενός μοναδικού τυχαία επιλεγμένου παρατηρητή. Αυτό αποκαλείται "Φαινόμενο των Δελφών". Φαίνεται ότι, αυτό που απέδειξε ο Linus είναι ότι το φαινόμενο αυτό βρίσκει εφαρμογή στην απομάκρυνση των bugs από ένα λειτουργικό σύστημα - ότι το Φαινόμενο των Δελφών μπορεί να ελέγξει την προγραμματιστική πολυπλοκότητα, ακόμη και στο επίπεδο πολυπλοκότητας ενός πυρήνα λειτουργικού συστήματος.

Είμαι υποχρεωμένος στον Jeff Dutky (dutky@wam.umd.edu) που υπέδειξε ότι ο Νόμος του Linus μπορεί να παραφρασθεί σε "Η Κατάργηση των Bugs Μπορεί να Παραλληλισθεί". Ο Jeff παρατηρεί ότι, αν και η κατάργηση των bugs απαιτεί από αυτούς που το πραγματοποιούν να επικοινωνούν με κάποιον συνεργαζόμενο προγραμματιστή, δεν απαιτεί σημαντικό συντονισμό μεταξύ των πρώτων. Έτσι, η κατάργηση των bugs δεν πέφτει στην παγίδα της ίδιας δευτεροβάθμιας πολυπλοκότητας και εξόδων management που καθιστούν την αύξηση των προγραμματιστών στην εργασία προβληματική.

Στην πράξη, η θεωρητική απώλεια αποτελεσματικότητας εξαιτίας του διπλασιασμού της εργασίας των προγραμματιστών για την κατάργηση των bugs σχεδόν ποτέ δεν απασχολεί τον κόσμο του Linux. Ένα αποτέλεσμα της "πολιτικής συχνών και πρωίμων εκδόσεων" είναι η ελαχιστοποίηση τέτοιων διπλασιασμών με την γρήγορη αναδραστική [fed-back] διάδοση των διορθώσεων.

Ο Brooks έκανε μια παρατήρηση σχετικά με εκείνη του Jeff: "Το συνολικό κόστος συντήρησης ενός ευρέως χρησιμοποιούμενου προγράμματος είναι, τυπικά, το 40% ή και περισσότερο του κόστους κατασκευής του. Αυτό το κόστος επηρεάζεται πολύ από τον αριθμό των χρηστών. *Περισσότεροι χρήστες βρίσκουν περισσότερα bugs*".

Περισσότεροι χρήστες βρίσκουν περισσότερα bugs, επειδή η προσθήκη περισσότερων χρηστών προσθέτει περισσότερους διαφορετικούς τρόπους δοκιμών του προγράμματος. Αυτό το φαινόμενο ενισχύεται όταν οι χρήστες είναι συν-προγραμματιστές. Κάθε προγραμματιστής προσεγγίζει την εργασία χαρακτηρισμού του bug με ελαφρά διαφορετικά αντιληπτικά κι αναλυτικά εργαλεία, από διαφορετική γωνιά, σε σύγκριση με άλλους. Το "Φαινόμενο των Δελφών" φαίνεται να έχει ισχύει ακριβώς εξαιτίας αυτής της ποικιλίας. Στο πλαίσιο της κατάργησης των bugs η ποικιλία αυτή τείνει επίσης να μειώσει τον διπλασιασμό της προσπάθειας.

Έτσι, η προσθήκη περισσότερων δοκιμαστών ίσως δεν μειώνει την πολυπλοκότητα του δύσκολου bug, από την σκοπιά του *προγραμματιστή*, αυξάνει όμως την πιθανότητα ότι τα εργαλεία κάποιου προγραμματιστή θα ταιριάξουν με το πρόβλημα με τέτοιο τρόπο ώστε το bug να καθίσταται επιπόλαιο *γι' αυτό το πρόσωπο.*

Ο Linus, όμως, φυλάει και τα ρούχα του. Σε περίπτωση που *υπάρχουν* σοβαρά bugs, η έκδοση του πυρήνα αριθμείται με τέτοιο τρόπο ώστε οι χρήστες να είναι σε θέση να επιλέξουν είτε την εγκατάσταση της τελευταίας "σταθερής" έκδοσης ή να ρισκάρουν με bugs για να απολαύσουν νέα χαρακτηριστικά. Αυτή η τακτική δεν βρίσκει μιμητές ανάμεσα στους hackers του Linux, αλλά ίσως θα έπρεπε. Το γεγονός ότι και οι δύο επιλογές είναι στην διάθεση των χρηστών τις κάνει ελκυστικές._

5 Ποτε Ένα Τριαντάφυλλο δεν Είναι Τριαντάφυλλο?

Έχοντας μελετήσει την συμπεριφορά του Linus και σχηματίσει μια θεωρεία για τον λόγο που ήταν επιτυχής, πήρα την σοβαρή απόφαση να δοκιμάσω την θεωρεία μου στο νέο μου σχέδιο (πολύ λιγότερο περίπλοκο και φιλόδοξο).

Αλλά το πρώτο πράγμα που έκανα ήταν να οργανώσω εκ νέου και να απλοποιήσω πολύ τον popclient. Η υλοποίηση του Carl Harris ήταν πολύ ηχηρή αλλά παρουσίαζε μια μη αναγκαία πολυπλοκότητα, κοινή σε πολλούς προγραμματιστές C. Μεταχειρίστηκε τον κώδικα σαν να είναι το κέντρο και τις δομές δεδομένων σαν υποστήριξη του κώδικα. Με αποτέλεσμα ο κώδικας να είναι όμορφος αλλά οι δομές δεδομένων σχεδιασμένες ad hoc και μάλλον άσχημες (τουλάχιστον σύμφωνα με τα υψηλά στάνταρ αυτού του παλιού LISP hacker).

Είχα κι ένα άλλο επιπλέον σκοπό εκτός της βελτίωσης του κώδικα και τον σχεδιασμό της δομής των δεδομένων την επανεγγραφή του κώδικα της εφαρμογής. Αυτός ήταν για να εμπλακώ σε κάτι που καταλάβαινα πλήρως. Δεν είναι διασκεδαστικό να έχεις την ευθύνη για διόρθωση των bugs ς' ένα πρόγραμμα που δεν καταλαβαίνεις.

Τον πρώτο μήνα ακολουθούσα την υλοποίηση του βασικού σχεδιασμού του Carl. Η πρώτη σοβαρή αλλαγή που έκανα ήταν να εισάγω υποστήριξη IMAP. Αυτό το έκανα με την εκ νέου οργάνωση των πρωτοκόλλων ς' έναν γενικό driver και σε τρεις πίνακες μεθόδων (για POP 2, POP 3 και IMAP). Αυτή και η προηγούμενη αλλαγή απεικονίζουν μια γενική αρχή που είναι καλό να έχουν κατά νου οι προγραμματιστές, ιδιαίτερα σε γλώσσες όπως η C που κανονικά δεν διαθέτουν δυναμική δακτυλογράφηση.

9. Έξυπνη δομή δεδομένων και κουτός κώδικας δουλεύουν καλύτερα απ' το αντίστροφο.

Brooks, Κεφάλαιο 9: "Αν μου δείξεις τον κώδικά σου και κρύψεις την δομή των δεδομένων σου, θα παραμείνω σε σύγχυση. Αν μου δείξεις την δομή των δεδομένων σου, δεν θα χρειαστώ τον κώδικά σου. Σύντομα θα τον μάθω".

Στην πραγματικότητα ο Brooks, στην παραπάνω παράγραφο, αναφέρεται σε "διαγράμματα ροής" και "πίνακες". Χρησιμοποιώντας, όμως, τριάντα χρόνων ορολογικής /πολιτιστικής μεταλλαγής, είναι σχεδόν το ίδιο.

Σ' αυτό το σημείο (αρχές Σεπτέμβρη 96, έξι μήνες από την ώρα μηδέν) άρχισα να σκέφτομαι ότι μια αλλαγή ονόματος ίσως είναι επιθυμητή-στο κάτω της γραφής, δεν ήταν πια απλά ένας POP client. Δίστασα, όμως, επειδή τίποτα στον σχεδιασμό δεν ήταν γνήσια καινούριο. Η έκδοση του popclient μου έπρεπε να αποκτήσει δική της ταυτότητα.

Αυτή η κατάσταση άλλαξε ριζικά όταν το fetchmail έμαθε πώς να προωθεί τα παραληφθέντα μηνύματα στην θύρα SMTP. Θα μιλήσω γι' αυτό σε λίγο. Είπα προηγουμένως ότι είχα αποφασίσει να χρησιμοποιήσω αυτό το σχέδιο για να δοκιμάσω την θεωρία μου σχετικά με την συμβολή του Linus Torvalds. Αυτό το έκανα ως εξής:

1. Εξέδιδα νωρίς και συχνά (Τουλάχιστον μια φορά στις δέκα μέρες. Στη διάρκεια περιόδων έντονου προγραμματισμού, κάθε μέρα).

2. Μεγάλωσα την λίστα των δοκιμαστών beta προσθέτοντας ς' αυτή κάθε έναν που επικοινωνούσε μαζί μου για το fetchmail.

3. Έστελνα φιλικές ανακοινώσεις στα μέλη της λίστας μετά από κάθε έκδοση, ενθαρρύνοντας τον κόσμο να πάρει μέρος στο εγχείρημα.

4. άκουγα τους δοκιμαστές μου, σφυγμομετρώντας ανάμεσά τους για τις αποφάσεις του σχεδιασμού, ευχαριστώντας τους κάθε φορά που απαντούσαν κι έστελναν διορθώσεις.

Το αποτέλεσμα αυτών των απλών μέτρων ήταν άμεσο. Απ' την αρχή του σχεδίου ελάμβανα αναφορές για bugs, τις οποίες οι περισσότεροι προγραμματιστές θα έδιναν τα πάντα για να έχουν, συχνά με προσκολλημένα καλές διορθώσεις. Λάμβανα συλλογισμένη κριτική, διασκεδαστικά μηνύματα, έξυπνες προτάσεις. Όλ' αυτά οδήγησαν στο:

10. Αν φέρεσαι στους δοκιμαστές σου σαν να είναι η πιο πολύτιμη πηγή σου, θα ανταποκριθούν σαν η πιο πολύτιμη πηγή σου.

Ένα μέτρο της επιτυχίας του fetchmail είναι το σπουδαίο μέγεθος της λίστας των δοκιμαστών, των φίλων του fetchmail. Μέχρι στιγμής έχει 249 μέλη και προστίθενται δύο ή τρεις κάθε βδομάδα.

Όπως διαπίστωσα στα τέλη Μαΐου 1997, η λίστα άρχισε να χάνει από τα περίπου 300 μέλη της, που είναι και ο μεγαλύτερος αριθμός τους, για έναν σημαντικό λόγο. Πολλοί άνθρωποι μου ζήτησαν να τους διαγράψω από την λίστα επειδή το fetchmail δούλευε τόσο καλά γι' αυτούς που δεν χρειάζονταν πλέον να είναι στη λίστα! Ίσως, κάτι τέτοιο είναι μέρος του φυσικού κύκλου ζωής κάθε ώριμου σχεδίου που υιοθετεί το στυλ "παζαριού".

6 O Popclient Γίνεται Fetchmail.

Το πραγματικό σημείο καμπής του σχεδίου ήταν όταν ο Harry Hochheiser μου έστειλε το προσχέδιο του κώδικά του για προώθηση ταχυδρομείο στην θύρα SMTP του υπολογιστή που θα φιλοξενούσε τον client. Σχεδόν αμέσως κατάλαβα ότι μια σωστή υλοποίηση αυτού του χαρακτηριστικού θα καθιστούσε κάθε άλλη μέθοδο παράδοσης ταχυδρομείου απαρχαιωμένη.

Για πολλές εβδομάδες εμβάθυνα στο fetchmail όλο και πιο πολύ και, ταυτόχρονα, ένιωθα ότι το interface άντεχε στην χρήση αλλά ήταν απεριποίητο, άκομψο και με πάρα πολλά μικρά options εδώ κι εκεί. Η επιλογή διοχέτευσης του ταχυδρομείου που είχε ήδη παραληφθεί στο αρχείο mailbox ή στην στάνταρ έξοδο με ενοχλούσε, αλλά δεν καταλάβαινα γιατί.

Αυτό που είδα, όταν σκέφτηκα την προώθηση μηνυμάτων μέσω SMTP, ήταν ότι ο popclient προσπαθούσε να κάνει πάρα πολλά πράγματα. Είχε σχεδιαστεί να είναι ένα πρόγραμμα μεταφοράς ταχυδρομείου (MTA) κι ένα πρόγραμμα τοπικής διανομής ταχυδρομείου (MDA). Με την προώθηση SMTP θα μπορούσε να απαλλαγεί απ' την λειτουργία MDA και να είναι ένα απλό MTA, μεταφέροντας τα μηνύματα σε άλλα προγράμματα, για τοπική παραλαβή, όπως ακριβώς το sendmail.

Γιατί να περιπλέκω τα πράγματα ρυθμίζοντας ένα πρόγραμμα διανομής ταχυδρομείου ή να εφαρμόζω την λειτουργία lock-and-append ς' ένα mailbox, τη στιγμή που η θύρα 25 εγγυημένα υπάρχει σε κάθε πλατφόρμα με υποστήριξη TCP/IP απ' την αρχή; Ιδιαίτερα όταν αυτό σημαίνει ότι, η αλληλογραφία που παραλαμβάνεται μοιάζει σίγουρα με ένα κανονικό ταχυδρομείο SMTP που ενεργοποιείται απ' τον αποστολέα, πράγμα το οποίο είναι αυτό που ζητάμε.

Τα μαθήματα εδώ είναι πολλά. Πρώτον, αυτή η ιδέα προώθησης μέσω SMTP ήταν το σημαντικότερο που κέρδισα απ' την συνειδητή προσπάθειά μου να εξομοιώσω τις μεθόδους του Linus. Ένας χρήστης μου έδωσε αυτή την πολύ καλή ιδέα-αυτό που έπρεπε να κάνω ήταν να καταλάβω τις επιπλοκές.

11. Το πιο σπουδαίο μετά από το να έχεις καλές ιδέες, είναι να αναγνωρίζεις τις καλές ιδέες των χρηστών σου. Μερικές φορές το τελευταίο είναι καλύτερο.

Σύντομα θα καταλάβεις ότι, αν είσαι εντελώς ειλικρινής για το πόσα χρωστάς στους άλλους και κάνεις αυτοκριτική, ο κόσμος θα σου συμπεριφέρεται σαν να είναι δική σου η επινόηση και θα είναι μετριόφρονες για την έμφυτη ευφυία σου. Όλοι μπορούμε να δούμε πόσο καλά αυτό λειτούργησε για τον Linus!

(Όταν έδωσα αυτό το κείμενο σε ένα συνέδριο με θέμα την Perl τον Αύγουστο του 1997, Ο Larry Wall κάθονταν στην πρώτη γραμμή. Επειδή εγώ καθόμουν στην τελευταία σειρά μου φώναξε, με στυλ θρησκευτικής αναζωπύρωσης [religious-revival style], "Πέστα, πέστα, αδερφέ!". Όλο το ακροατήριο γέλασε , επειδή ήξεραν ότι οι απόψεις μου είχαν δουλέψει και για τον δημιουργό της Perl.)

Μετά από λίγες εβδομάδες δουλεύοντας στο σχέδιο με το ίδιο πνεύμα άρχισα να παίρνω παρόμοιους επαίνους όχι μόνο από τους χρήστες μου, αλλά κι από άλλους ανθρώπους που έμαθαν όλ' αυτά. Φύλαξα σαν θησαυρό κάποια από αυτά τα Email, μερικές φορές τα ξανακοιτάω όταν αρχίζω να αναρωτιέμαι αν η ζωμού αξίζει τον κόπο :-)

Όμως, εδώ υπάρχουν άλλα δύο θεμελιώδη μη πολιτικά μαθήματα, που εφαρμόζονται σε κάθε είδος σχεδιασμού.

12. Συχνά, οι πιο φανερές και καινοτομικές λύσεις εμφανίζονται όταν βλέπεις ότι η αντίληψη που έχεις για το πρόβλημα είναι λάθος.

Προσπαθούσα να λύσω λάθος πρόβλημα με το να συνεχίζω να αναπτύσσω τον popclient σαν ένα συνδυασμό MTA/MDA, με όλα τα είδη μεθόδων τοπικής διανομής ταχυδρομείου. Έπρεπε να σκεφτώ πάλι το fetchmail απ' την αρχή, σαν ένα καθαρό MTA, σαν μέρος του κανονικού SMTP Internet ταχυδρομείου.

Όταν φτάνεις σε αδιέξοδο-όταν νομίζεις ότι δυσκολεύεσαι να σκεφτείς το επόμενο patch-συχνά είναι καιρός να σκεφτείς όχι αν έχεις την σωστή απάντηση, αλλά αν ρωτάς την σωστή ερώτηση. Ίσως πρέπει να το πρόβλημα να διευθετηθεί εκ νέου.

Αυτό κι έκανα. Ξεκάθαρα, αυτό που έπρεπε να κάνω ήταν (1) να εισάγω υποστήριξη προώθησης SMTP στον γενικό driver, (2) να την ρυθμίσω ως εξ ορισμού λειτουργία και, (3) να απαλλαγώ από όλες τις άλλες λειτουργίες διανομής, ιδιαίτερα την αποθήκευση των μηνυμάτων σε αρχείο και την μεταβίβαση στην στάνταρ έξοδο.

Καθυστέρησα για λίγο στο στάδιο (3), φοβούμενος να αναστατώσω τους χρήστες που χρησιμοποιούσαν για μεγάλο διάστημα popclient και ήσαν εξαρτημένοι απ' τους εναλλακτικούς μηχανισμούς παραλαβής. Θεωρητικά, θα μπορούσαν να μεταβαίνουν στα αρχεία .forward ή στα non-sendmail ισοδύναμά τους, και να έχουν το ίδιο αποτέλεσμα. Στην πράξη, η μετάβαση αυτή φαινόταν ακατάστατη.

Όταν, όμως, τα κατάφερα το κέρδος αποδείχτηκε μεγάλο. Τα δυσκολότερα μέρη του driver εξαφανίστηκαν. Η ρύθμιση έγινε ριζικά απλούστερη-τέρμα η αναζήτηση για το MDA του συστήματος και το mailbox του χρήστη, τέλος στις ανησυχίες για το εάν το λειτουργικό πρόγραμμα υποστηρίζει κλείδωμα των αρχείων [file locking].

Επίσης, ο μόνος τρόπος να χάσεις μηνύματα εξαφανίστηκε. Αν έχεις ορίσει η παραλαβή των μηνυμάτων να προσκολληθεί ς' ένα αρχείο κι ο δίσκος είναι γεμάτος, τα μηνύματά σου χάνονταν. Αυτό δεν συμβαίνει με την προώθηση μέσω SMTP, επειδή ο SMTP listener δεν θα επιστρέψει OK μέχρις ότου τα μηνύματα παραληφθούν ή, τουλάχιστον, μπουν στην σειρά για παράδοση κάποια άλλη στιγμή.

Ακόμη, βελτιώθηκε η απόδοση (αν και δεν θα το προσέξει κανείς με μια απλή χρήση του προγράμματος). Ένα άλλο όχι ασήμαντο όφελος αυτής της αλλαγής ήταν ότι το εγχειρίδιο χρήσης έγινε απλούστερο.

Αργότερα, έπρεπε να ρυθμίσω την παραλαβή των μηνυμάτων μέσω ενός τοπικού MDA καθορισμένο από τον χρήστη, ώστε να επιτρέψω τον έλεγχο κάποιων σκοτεινών καταστάσεων σχετικά με την dynamic SLIP. Βρήκα, όμως, έναν πιο απλό τρόπο για να το κάνω.

Ηθικό δίδαγμα; Μην διστάζεις να πετάξεις παρωχημένα χαρακτηριστικά όταν μπορείς να το κάνεις χωρίς απώλειες στην αποτελεσματικότητα. Ο Αντουάν Σαιντ-Εξπερύ (που ήταν πιλότος και σχεδιαστής αεροσκαφών, όταν δεν έγραφε κλασικά βιβλία για παιδιά) έχει πει:

13. Η τελειότητα (στον σχεδιασμό) είναι δυνατή όχι όταν δεν υπάρχει κάτι για να προσθέσεις, αλλά μάλλον όταν δεν υπάρχει κάτι για να αφαιρέσεις.

Όταν ο κώδικάς σου γίνεται καλύτερος και απλούστερος, τότε *ξέρεις* ότι είναι σωστός. Και στην πορεία το fetchmail απέκτησε δική του ταυτότητα, διαφορετική απ' τον πρόγονό του popclient.

Ήταν ώρα για την αλλαγή του ονόματος. Το νέο πρόγραμμα έμοιαζε περισσότερο μ' ένα διπλό sendmail που διέθετε ο παλιός poclient. Και τα δύο είναι MTA, αλλά εκεί που το sendmail σπρώχνει την παράδοση του ταχυδρομείου, ο νέος popclient την ελκύει. Έτσι, μετά από δύο μήνες λειτουργίας χωρίς εμπόδια, τον ονόμασα fetchmail.

7 Το Fetchmail Ενηλικιώνεται.

Να μια, λοιπόν, μ' ένα καλοφτιαγμένο και νέο σχεδιασμό, μ' έναν κώδικα που ήξερα ότι επεξεργάστηκα καλά επειδή τον χρησιμοποιούσα κάθε μέρα, και μία λίστα δοκιμαστών που μεγάλωνε. Βαθμιαία άρχισα να καταλαβαίνω ότι δεν έκανα πλέον κάποιον τετριμμένο προσωπικό προγραμματισμό που ίσως να ήταν χρήσιμος για μερικούς ανθρώπους. Είχα στα χέρια μου ένα πρόγραμμα που κάθε hacker με Unix και σύνδεση SLIP/PPP χρειαζόταν στ' αλήθεια.

Με το χαρακτηριστικό της προώθησης μέσω SMTP, το πρόγραμμα μπήκε πρώτο στον συναγωνισμό κι έγινε ο "φονιάς" της κατηγορίας του, έγινε ένα κλασικό πρόγραμμα που γεμίζει τόσο καλά την γωνιά που καταλαμβάνει ώστε τα εναλλακτικά προς αυτό προγράμματα όχι απορρίφθηκαν αλλά σχεδόν ξεχάστηκαν.

Είναι δύσκολο να έχεις τέτοια αποτελέσματα. Πρέπει να σπρωχτείς ς' αυτά με το να σχεδιάζεις νέες ιδέες τόσο ισχυρές ώστε, στο τέλος, τα αποτελέσματα να φαίνονται αναπόφευκτα, φυσικά. Ο μόνος τρόπος για να έχεις τέτοιες ιδέες είναι να έχεις πολλές ιδέες-ή να έχεις την σχεδιαστική κρίση να οδηγείς τις καλές ιδέες άλλων ανθρώπων πέρα απ' το σημείο που οι εμπνευστές τους θα μπορούσαν να το οδηγήσουν.

Ο Andrew Tanenbaum είχε την αρχική ιδέα να δημιουργήσει ένα απλό τοπικό σύστημα Unix για 386, για να το χρησιμοποιήσει σαν διδακτικό εργαλείο. Ο Linus Torvalds ώθησε την ιδέα του Minix πιο πέρα απ' τον Andrew. Με τον ίδιο τρόπο (αν και σε μικρότερη κλίμακα) πήρα μερικές ιδέες των Carl Harris και Harry Hochheiser και τις ώθησα στα άκρα. Κανείς μας δεν ήταν "γνήσια" ιδιοφυία.

Τα αποτελέσματα ήσαν μεθυστικά-ήταν ακριβώς η επιτυχία που κάθε προγραμματιστής λατρεύει! Η επιτυχία αυτή σήμαινε ότι έπρεπε να θέσω τα στάνταρ μου ψηλότερα. Για να κάνω το fetchmail όσο καλό έβλεπα ότι μπορεί να γίνει, έπρεπε να το κατασκευάσω όχι μόνο για τις δικές μου ανάγκες, αλλά επίσης να συμπεριλάβω και να υποστηρίξω χαρακτηριστικά χρήσιμα για άλλους, έξω όμως από την δική μου τροχιά. Κι αυτό να το κάνω διατηρώντας το πρόγραμμα απλό και γερό.

Το πρώτο και πολύ σημαντικό χαρακτηριστικό, που συμπεριέλαβα μετά απ' όλ' αυτά, ήταν η υποστήριξη multidrop-η δυνατότητα παραλαβής του ταχυδρομείου από τα mailboxes που έχουν συσσωρεύσει όλα τα μηνύματα μιας ομάδας χρηστών και η δρομολόγηση του κάθε μηνύματος στον παραλήπτη του.

Αποφάσισα να κάνω μερική προσθήκη υποστήριξης multidrop επειδή μερικοί χρήστες διαμαρτύρονταν, αλλά κυρίως επειδή νόμιζα ότι θα ξεκαθάριζε τα ελαττώματα του κώδικα single-drop αναγκάζοντάς με να ασχοληθώ εξολοκλήρου με την διαδικασία διευθυνσιοδότησης / καταχώρησης. Κι έτσι αποδείχθηκε. Πήρα το *RFC 822* `<http://www.internic.net/rfc/rfc822.txt>` και η εισαγωγή αυτής της παραμέτρου στην υπορουτίνα με καθυστέρησε αρκετό χρόνο, όχι επειδή τα ξεχωριστά κομμάτια της είναι δύσκολα, αλλά επειδή ενέπλεκε ένα σωρό από αλληλοεξαρτώμενες και ιδιότροπες λεπτομέρειες.

Όμως, η διευθυνσιοδότηση multidrop κατέληξε να γίνει μια πολύ καλή σχεδιαστική απόφαση.

14. Κάθε εργαλείο θα πρέπει να είναι χρήσιμο με κάθε αναμενόμενο τρόπο, αλλά ένα πραγματικά σπουδαίο εργαλείο προσφέρεται για χρήσεις που ποτέ δεν θα περίμενες.

Η αναπάντεχη χρήση του multidrop fetchmail είναι η δυνατότητα λειτουργίας μιας λίστας αλληλογραφίας (mailing lists) που διατηρείτε στην πλευρά του *client* της σύνδεσης SLIP/PPP, και ταυτόχρονα ς' αυτή έχει γίνει η επέκταση του alias. Αυτό σημαίνει ότι, κάποιος που χρησιμοποιεί έναν προσωπικό υπολογιστή, μέσω ενός λογαριασμού σε κάποιον ISP, μπορεί να χειριστεί την λίστα αλληλογραφίας χωρίς να διατηρεί συνεχώς την πρόσβασή του στα αρχεία ψευδωνύμων του ISP.

Μια άλλη ενδιαφέρουσα αλλαγή που απαίτησαν οι δοκιμαστές ήταν η υποστήριξη λειτουργίας 8-bit MIME. Αυτό ήταν εύκολο να το κάνω, επειδή είχα προσέξει να διατηρήσω τον 8-bit κώδικα καθαρό. Όχι επειδή προέβλεψα την απαίτηση γι' αυτό το χαρακτηριστικό, αλλά μάλλον υπακούοντας σε έναν άλλο κανόνα:

15. Όταν κατασκευάζεις gateway λογισμικό οποιουδήποτε είδους, προσπάθησε να αλλοιώσεις το stream των δεδομένων όσο το δυνατόν λιγότερο-και ΠΟΤΕ μην πετάς πληροφορίες αν οι παραλήπτες σου δεν σε αναγκάσουν!

Αν δεν υπάκουα ς' αυτό τον κανόνα η υποστήριξη 8-bit MIME θα ήταν δύσκολη και γεμάτη bugs. Όπως ήταν τώρα, αυτό που έπρεπε να κάνω ήταν να διαβάσω το *RFC 1652* `<http://www.internic.net/rfc/rfc1652.txt>` και να προσθέσω τετριμμένα bits για δημιουργία κεφαλίδων.

$Μερικοί Ευρωπαίοι χρήστες μου ζητούσαν να προσθέσω μια επιλογή περιορισμού του αριθμού των μηνυμάτων που παραλάβαιναν σε κάθε χρήση του fetchmail (ώστε να ελέγχουν τα έξοδο σύνδεσής τους στα ακριβά τηλεφωνικά δίκτυα). Αντιστάθηκα ς' αυτή την απαίτηση για αρκετό καιρό, και δεν μπορώ να πω ότι είμαι χαρούμενος γι' αυτό. Αλλά όταν γράφεις για τον κόσμο πρέπει ν' ακούς τους πελάτες σου-αυτό δεν αλλάζει επειδή δεν σε πληρώνουν με χρήματα.

8 Μερικά Ακόμη Μαθήματα απο το Fetchmail.

Πριν επιστρέψουμε σε θέματα γενικού ενδιαφέροντος παραγωγής λογισμικού, υπάρχουν ακόμη μερικά μαθήματα απ' την εμπειρία του fetchmail για να μελετήσουμε.

Η σύνταξη των αρχείων rc περιλαμβάνει προαιρετικές δεσμευμένες λέξεις "θορύβου" [noise keyword] που αγνοούνται πλήρως από τον προγραμματιστή. Η σύνταξη Αγγλικού τύπου που προτιμούν είναι πολύ πιο αναγνώσιμη απ' τα παραδοσιακά λιτά ζεύγη τιμών δεσμευμένων λέξεων που παίρνει κανείς όταν απογυμνώσει αυτές τις δεσμευμένες λέξεις.

Αυτό άρχισε σαν ένα πείραμα, όταν πρόσεξα πόσο πολύ οι ανακοινώσεις των αρχείων rc άρχιζαν να μοιάζουν μια επιβεβλημένη μικρή γλώσσα. (Αυτός είναι ο λόγος που άλλαξα την αρχική δεσμευμένη λέξη του popclient από "server" σε "poll").

Μου φαινόταν ότι αν προσπαθούσα να κάνω αυτή την επιβεβλημένη μικρή γλώσσα να μοιάζει με τα Αγγλικά θα γινόταν πιο εύχρηστη. Όμως, αν και είμαι αφοσιωμένος οπαδός της σχολής σχεδιασμού που αποκαλείτε "κάνετο μια γλώσσα", όπως μας παραδειγματίζουν το Emacs η HTML και πολλές μηχανές βάσεων δεδομένων, δεν είμαι σπουδαίος οπαδός της σύνταξης Αγγλικού τύπου.

Οι παραδοσιακοί προγραμματιστές έτειναν να ευνοούν σύνταξη ελέγχου που είναι πολύ ακριβής και συμπαγής και δεν είχαν καθόλου περιττές επαναλήψεις. Αυτό είναι μια πολιτισμική κληρονομιά από την εποχή που οι πόροι των υπολογιστών ήσαν ακριβοί, ώστε η ανάλυση των σταδίων να είναι όσο το δυνατόν πιο φτηνή κι απλή. Τα Αγγλικά, με περίπου 50% περιττών επαναλήψεων, φαινόταν τότε πολύ ακατάλληλο μοντέλο.

Δεν είναι αυτός ο λόγος που αποφεύγω την σύνταξη Αγγλικού τύπου. Την αναφέρω εδώ μόνο για να την καταργήσω. Έχοντας φτηνούς κύκλους και πυρήνα η λακωνικότητα δεν θα έπρεπε να είναι αυτοσκοπός. Σήμερα, είναι πιο σημαντικό η γλώσσα να είναι βολική για τους ανθρώπους από το να είναι φτηνή για τους υπολογιστές.

Υπάρχουν, όμως, σοβαροί λόγοι να ανησυχούμε. Ένας, είναι το κόστος που δημιουργεί η περιπλοκή των αναλυμένων σταδίων-κανείς δεν θέλει να αυξήσει αυτή την περιπλοκή στο σημείο να είναι πηγή bugs και σύγχυσης των χρηστών. Ένας άλλος είναι ότι, προσπαθώντας να κάνουμε μια γλώσσα σύνταξης να μοιάζει με Αγγλικά συχνά απαιτείται τα "Αγγλικά" που μιλά αυτή να ξεφεύγουν αρκετά από κάθε γνωστή μορφή, τόσο ώστε η επιφανειακή ομοιότητα προς την φυσική γλώσσα να είναι τόσο συγχυτική όσο μπορεί να είναι κάθε παραδοσιακή σύνταξη. (Αυτό το συναντά κανείς σε πολλές εμπορικές γλώσσες που εφαρμόζουν αναζήτηση μέσω βάσεων δεδομένων και τις λεγόμενες γλώσσες "τέταρτης γενιάς".

Η σύνταξη ελέγχου του fetchmail φαίνεται να αποφεύγει αυτά τα προβλήματα, επειδή ο χώρος κυριαρχίας της γλώσσας του είναι πολύ περιορισμένος. Δεν είναι μια γενική γλώσσα. Τα πράγματα που λέει αυτή δεν είναι τόσο περιπλεγμένα, ώστε υπάρχει μικρή πιθανότητα σύγχυσης όταν κινούμαστε μεταξύ ενός μικρού υποσυνόλου της Αγγλικής και της πραγματικής γλώσσας ελέγχου. Εδώ υπάρχουν ευρύτερα διδάγματα:

16. Όταν μια γλώσσα δεν είναι πλήρης κατά Turing (Turing-complete), τότε η καλύτερη λύση είναι η συντακτική ζάχαρη.

Ένα άλλο μάθημα είναι για την ασάφεια της ασφάλειας. Μερικοί χρήστες του fetchmail μου ζήτησαν να κάνω αλλαγές στο πρόγραμμα ώστε να αποθηκεύει κρυπτογραφημένους κωδικούς στο αρχείο rc, ώστε να μην μπορούν να τους δουν τυχαία οι αδιάκριτοι.

Δεν το έκανα, επειδή κάτι τέτοιο δεν προσφέρει ασφάλεια. Οποιοσδήποτε έχει άδεια να διαβάσει το αρχείο rc θα μπορεί να χρησιμοποιήσει το fetchmail-κι αν ψάχνουν τον κώδικά σας θα μπορούν να απομονώσουν τον αποκωδικοποιητή μέσα από τον ίδιο τον κώδικα του fetchmail.

Όποια κρυπτογράφηση κώδικα στο αρχείο .fetchmailrc κι αν επιχειρηθεί δίνει μια εσφαλμένη αίσθηση ασφάλειας. Ο γενικός κανόνας είναι ο εξής:

17. Ένα σύστημα ασφαλείας είναι τόσο ασφαλές όσο είναι και το μυστικό του. Προσοχή στα ψευδο-μυστικά.

9 Προϋποθέσεις για το Στυλ Παζαριού.

Οι πρώτοι αναγνώστες αυτού του κειμένου έθεταν ερωτήσεις για τις προϋποθέσεις για έναν επιτυχημένο προγραμματισμό σε στυλ παζαριού, συμπεριλαμβανομένων τόσο των προσόντων του επικεφαλή του εγχειρήματος, όσο και την κατάσταση του κώδικα κατά τη στιγμή που κάποιος, δημόσια, ξεκινά την προσπάθεια οργάνωσης μιας ομάδας συν-προγραμματιστών.

Είναι ξεκάθαρο ότι κανείς δεν μπορεί, με το παζαριώτικο στυλ, να δημιουργήσει κώδικα εκ του μηδενός στο Παζαριώτικο στυλ. Μπορεί κανείς να δοκιμάσει, να βελτιώσει και να κάνει debug κάποιου κώδικα, αλλά θα ήταν πολύ δύσκολο να *δημιουργήσει* εκ νέου ένα project μ' αυτή τη μέθοδο. Ούτε εγώ ούτε κι ο Linus το δοκιμάσαμε. Η ομάδα προγραμματισμού που μόλις δημιουργήσατε έχει ανάγκη κάτι που είναι χρησιμοποιήσιμο και δοκιμασμένο, για να ξεκινήσει.

Όταν ξεκινά κανείς την δημιουργία της ομάδας προγραμματισμού πρέπει να μπορείτε να *παρουσιάσετε αληθοφανείς υποσχέσεις.* Το πρόγραμμά σας δεν χρειάζεται να δουλεύει ιδιαίτερα καλά. Μπορεί να είναι αργό, όλο bugs, ατελές και φτωχά τεκμηριωμένο. Εκεί που δεν πρέπει να αποτύχετε είναι να πείσετε τους πιθανούς συν-προγραμματιστές σας ότι μπορεί να εξελιχθεί σε κάτι πετυχημένο μέσα στο προβλεπτό μέλλον.

Το Linux και το fetchmail δημοσιοποιήθηκαν έχοντας στιβαρό, ελκυστικό σχεδιασμό. Πολλοί άνθρωποι που σκέφτονται με το μοντέλο του παζαριού όπως την παρουσίασα, έχουν λάβει σοβαρά υπόψη αυτό το κρίσιμο σημείο και κατέληξαν στο συμπέρασμα ότι είναι απόλυτα αναγκαίο ο επικεφαλής του σχεδίου να διαθέτει σχεδιαστική διαίσθηση κι ευφυία.

$Αλλά ο Linus πήρε τα σχέδιά του από το Unix. Εγώ πήρα τα δικά μου αρχικά απ' το popclient (αν και αργότερα θα άλλαζε πάρα πολύ και περισσότερο αναλογικά με το Linux). Πρέπει, λοιπόν, ο επικεφαλής, συντονιστής μιας προσπάθειας παζαριώτικου στυλ να έχει εξαιρετικό σχεδιαστικό ταλέντο, ή μπορεί να προχωρήσει ενισχύοντας το σχεδιαστικό ταλέντο άλλων;

Δεν νομίζω ότι είναι σημαντικό ο συντονιστής να μπορεί να δημιουργεί σχέδια εξαιρετικής ευφυΐας, αλλά είναι απόλυτα σημαντικό να μπορεί να *αναγνωρίσει τις καλές σχεδιαστικές ιδέες άλλων.*

Αυτό το απέδειξαν τα project του Linux και του fetchmail. Ο Linus, ενώ δεν ήταν (όπως είπαμε και πριν) ένας γνήσιος σχεδιαστής, επέδειξε δεξιότητα στο να αναγνωρίζει τα καλά σχέδια και να τα ολοκληρώνει στον πυρήνα του Linux. Κι έχω ήδη περιγράψει πώς η πιο ισχυρή σχεδιαστική ιδέα για το fetchmail (δηλαδή, η προώθηση μέσω SMTP) προήλθε από κάποιον άλλο.

Οι πρώτοι αναγνώστες αυτού του κειμένου με κολακεύουν υπαινισσόμενοι ότι έχω προδιάθεση να υποεκτιμώ την σχεδιαστική γνησιότητα σχεδίων παζαριώτικου στυλ, επειδή διαθέτω ο ίδιος αρκετή από αυτή και, έτσι, την θεωρώ δεδομένη. Ίσως αυτό να αληθεύει. Ο σχεδιασμός είναι σίγουρα η καλύτερη ικανότητά μου.

Το πρόβλημα, όμως, όταν κανείς είναι ευφυής στον σχεδιασμό λογισμικού είναι όταν κάτι τέτοιο αρχίζει να γίνεται συνήθεια-αρχίζεις να κάνεις πράγματα περίπλοκα όταν θα έπρεπε να είναι δυνατά και απλά. Πολλά προγράμματα κατέρρευσαν επειδή έκανα αυτό το λάθος, αλλά το απέφυγα στο fetchmail.

Πιστεύω, λοιπόν, ότι το project του fetchmail πέτυχε εν μέρει επειδή εμπόδισα την τάση μου να είμαι έξυπνος. Κάτι τέτοιο έρχεται σε αντίθεση με την σχεδιαστική γνησιότητα που είναι ουσιώδης για ένα επιτυχημένο παζαριώτικο project. Υποθέστε ότι ο Linus προσπαθούσε να πετύχει θεμελιώδης καινοτομίες κατά τη διάρκεια του προγραμματισμού. Φαίνεται απίθανο ότι ο πυρήνας που θα έπαιρνε μ' αυτό τον τρόπο να ήταν σταθερός και πετυχημένος ;

Ένα συγκεκριμένο επίπεδο σχεδιαστικής και προγραμματιστικής ικανότητας είναι απαραίτητο, βέβαια, αλλά αναμένω ότι όλοι όσοι σκέφτονται σοβαρά να ασχοληθούν με την μέθοδο παζαριού να είναι ήδη πάνω απ' αυτό το μίνιμουμ. Η εσωτερική αγορά της κοινότητας ανοιχτού κώδικα ασκεί λεπτές πιέσεις στους ανθρώπους να μην ασκούν προγραμματιστικές προσπάθειες τις οποίες δεν μπορούν να ακολουθήσουν ως το τέλος. Ως τώρα αυτή η συμβουλή εισακούεται.

Υπάρχει κι ένα άλλο είδος ικανότητας που κανονικά δεν συνδέεται με την ανάπτυξη λογισμικού η οποία, νομίζω, είναι τόσο σημαντική για το παζαριώτικο project όσο και η σχεδιαστική ευφυία-και ίσως περισσότερο. Ένας συντονιστής χαλαρού σχεδίου πρέπει να έχει καλούς συνεργάτες και ικανότητες επικοινωνίας.

Για να οργανώσεις μια ομάδα προγραμματιστών πρέπει να έλξεις τους ανθρώπους, να προκαλέσεις το ενδιαφέρον τους σε ό,τι κάνεις και να τους προκαλείς την ευχαρίστηση για την ποσότητα εργασίας που κάνουν. Τα τεχνικά σκαμπανεβάσματα θα είναι πολλά μέχρις ότου να το καταφέρετε, αλλά δεν είναι αυτό το θέμα μας. Η προσωπικότητα που προβάλετε έχει σημασία.

Δεν είναι σύμπτωση που ο Linus είναι καλό άτομο και κάνει τους ανθρώπους να θέλουν να τον βοηθήσουν. Δεν είναι τυχαίο που είμαι ένας ενεργητικός εξωστρεφής τύπος που του αρέσει να δουλεύει με άλλους. Για να έχει αποτελέσματα το παζαριώτικο μοντέλο, βοηθάει αν μπορείς να συνεργαστείς με ευχάριστους ανθρώπους.

10 Το Κοινωνικό Πλαίσιο του Λογισμικού Ανοιχτού Κώδικα.

Αυτό που λένε είναι σωστό: οι καλύτερες παρεμβάσεις στο λογισμικό [hacks] ξεκινούν σαν προσωπικές λύσεις στα καθημερινά προβλήματα του προγραμματιστή κι εξαπλώνονται επειδή αυτά γίνονται τυπικά για μεγάλο αριθμό χρηστών. Αυτό μας παραπέμπει στο θέμα του κανόνα (1), που μπορεί να διατυπωθεί με πιο χρήσιμο τρόπο:

18. Για να λύσεις ένα ενδιαφέρον πρόβλημα, βρες ένα πρόβλημα που είναι ενδιαφέρον για σένα.

Αυτό συνέβη με τον Carl Harris και το παλιό popclient, αυτό συνέβη και με μένα και το fetchmail. Αλλά αυτό ήδη το έχουμε καταλάβει. Το ενδιαφέρον σημείο, το σημείο στο οποίο οι ιστορίες του Linux και του fetchmail φαίνεται να απαιτούν να εστιάσουμε επάνω τους, είναι το επόμενο στάδιο-η εξέλιξη του λογισμικού παρουσία μιας μεγάλης και δραστήριας κοινότητας χρηστών και συν-προγραμματιστών.

Στο βιβλίο του "The Mythical Man-Month" ο Fred Brooks παρατηρεί ότι, ο χρόνος του προγραμματιστή δεν είναι ανταλλάξιμος. Η προσθήκη προγραμματιστών ς' ένα καθυστερημένο project λογισμικού το καθυστερεί περισσότερο. Ισχυρίζεται ότι, το κόστος της πολυπλοκότητας και της επικοινωνίας ενός project αυξάνονται γεωμετρικά, ενώ η εργασία που ολοκληρώνεται αυξάνεται αριθμητικά. Αυτή η άποψη έγινε γνωστή σαν "ο Νόμος του Brooks" και αναγνωρίζεται από πολλούς σαν κάτι τελείως πασιφανές. Αλλά αν ο Νόμος του Brooks απέδιδε την όλη εικόνα το Linux θα ήταν αδύνατο να υπάρξει.

Το κλασικό κείμενο "Η Ψυχολογία του Προγραμματισμού με Ηλεκτρονικό Υπολογιστή", του Gerald Weinberg, μας δίνει μια ζωτική διόρθωση της άποψης του Brooks. Στην συζήτησή του για τον "χωρίς εγωισμό" προγραμματισμό ο Weinberg παρατηρεί ότι, σε εργασίες που προγραμματιστές δεν απαιτούν "εδαφικά δικαιώματα" για τον κώδικα τους και ενθαρρύνουν άλλους ανθρώπους να αναζητήσουν bugs και πιθανές βελτιώσεις, η βελτίωση συμβαίνει δραματικά γρηγορότερα από οπουδήποτε αλλού.

Οι επιλογές ορολογίας που έκανε ο Weinberg ίσως εμποδίζουν την ανάλυσή του να κερδίσει την αποδοχή που της αξίζει-κάποιος θα χαμογελούσε στον χαρακτηρισμό των hackers του Internet σαν "χωρίς εγωισμό". Αλλά πιστεύω ότι οι ισχυρισμοί του μοιάζουν σήμερα περισσότερο επιβλητικοί από ποτέ.

Η ιστορία του Unix θα έπρεπε να μας έχει προετοιμάσει για ό,τι μαθαίνουμε απ' το Linux (και για ότι έχω εγώ διαπιστώσει, πειραματικά και σε μικρότερη κλίμακα όταν σκόπιμα αντέγραψα τις μεθόδους του Linus). Ότι, δηλαδή, ενώ η σύνταξη κώδικα παραμένει μια ουσιωδώς μοναχική δραστηριότητα, οι πραγματικά καλές προγραμματιστικές εργασίες προέρχονται όταν χρησιμοποιείται η προσοχή και η διανοητική δύναμη της ομάδας. Ο προγραμματιστής που χρησιμοποιεί το μυαλό του του μόνος ή μόνη ς' ένα κλειστό project θα μείνει πίσω απ' τον προγραμματιστή που ξέρει πώς να δημιουργήσει ένα ανοιχτό, εξελικτικό πλαίσιο στο οποίο ο εντοπισμός των bugs και η επίτευξη βελτιώσεων καταφέρονται από εκατοντάδες ανθρώπων.

Όμως, ο παραδοσιακός κόσμος του Unix εμποδίστηκε στην προσπάθειά του να ωθήσει αυτή την προσέγγιση στην τελική της φάση από πολλούς παράγοντες. Τέτοιοι ήταν οι νομικοί περιορισμοί διαφόρων αδειών χρήσης πνευματικών δικαιωμάτων, επαγγελματικά μυστικά και τα εμπορικά συμφέροντα. Ένας άλλος ήταν ότι το Internet δεν ήταν ακόμη αρκετά καλό.

Πριν από την έλευση του φτηνού Internet υπήρχαν κάποιες γεωγραφικά συμπαγείς ομάδες των οποίων η κουλτούρα ενεθάρρυνε τον "μη εγωιστικό" προγραμματισμό του Weinberg κι ένας προγραμματιστής μπορούσε χωρίς δυσκολία να προσελκύσει πολλούς ικανούς παρατηρητές και προγραμματιστές. Τα Bell Labs, MIT AI Lab, το UC Berkeley έγιναν οίκοι θρυλικών καινοτομιών που είναι ακόμη σε ισχύ.

Το Linux ήταν το πρώτο project που έκανε συνειδητές και επιτυχείς προσπάθειες να αξιοποιηθεί ολόκληρος ο *κόσμος* σαν η δική του δεξαμενή ταλέντων. Δεν νομίζω ότι είναι σύμπτωση ότι η γόνιμη περίοδος του Linux εμφανίζεται την ίδια περίοδο με εκείνη της γέννησης του Παγκόσμιου Ιστού (World Wide Web-WWW) και ότι το Linux εγκατέλειψε την παιδική του ηλικία κατά την περίοδο 1993-1994 η οποία είδε την απογείωση της βιομηχανίας Παροχέων Υπηρεσιών Internet (ISP) και της έκρηξης του μεγάλου ρεύματος ενδιαφέροντος για το Internet. Ο Linus ήταν ο πρώτος άνθρωπος που έμαθε πώς να παίζει με τους νέους κανόνες που δημιουργούσε το Internet.

Ενώ το φτηνό Internet ήταν μια αναγκαία συνθήκη για την ανάπτυξη του μοντέλου Linux, νομίζω πως από μόνο του δεν ήγαν μια επαρκής συνθήκη. Ένας άλλο ζωτικός παράγοντας ήταν η ανάπτυξη ενός ηγετικού στυλ κι ενός συνόλου συνεργατικών συνηθειών που επέτρεψαν στους προγραμματιστές να προσελκύσουν συν-προγραμματιστές και να αποκομίσουν την μέγιστη ισχύ.

Τι είναι, όμως, αυτό το ηγετικό στυλ και οι συνήθειες; Δεν μπορούν να βασίζονται σε σχέσεις εξουσίας-ακόμη κι αν μπορούσαν, η εξαναγκαστική εξουσία δεν μπορεί να προκαλέσει τα αποτελέσματα που διαπιστώνουμε. Ο Weinberg παραθέτει την αυτοβιογραφία του Πιοτρ Αλεξέιεβιτς Κροπότκιν, Ρώσου αναρχικού του 19ου αιώνα, "Οι αναμνήσεις ενός Επαναστάτη".

"Έχοντας ανατραφεί από μια οικογένεια που είχε στην ιδιοκτησία της δουλοπάροικους ξεκίνησα την ζωή μου, όπως όλοι οι νέοι της εποχής μου, με πίστη μεγάλη στην αναγκαιότητα του να διατάζεις, να μαλώνεις, να τιμωρείς, και όλ' αυτά. Αλλά όταν, αρκετά νωρίς, έπρεπε να λειτουργήσω σοβαρές επιχειρήσεις και να συσχετιστώ με [ελεύθερους] ανθρώπους, και όταν το παραμικρό λάθος μπορούσε να οδηγήσει σε οδυνηρές επιπτώσεις, άρχισα να εκτιμώ την διαφορά μεταξύ της δράσης βάσει των αρχών της εντολής και της πειθαρχίας και της δράσης βάσει της αρχής της κοινής κατανόησης. Οι πρώτες εφαρμόζονται θαυμάσια σε μια στρατιωτική παρέλαση, αλλά δεν αξίζει το παραμικρό στην πραγματική ζωή όπου ο στόχος μπορεί να επιτευχθεί μόνο με τις σκληρή προσπάθεια πολλών συγκλινουσών θελήσεων"

Η "σκληρή προσπάθεια πολλών συγκλινουσών θελήσεων" είναι ακριβώς αυτό που απαιτεί το project του Linux-και η "αρχή της διαταγής" είναι αδύνατον να εφαρμοστεί μεταξύ εθελοντών, μέσα στον αναρχικό παράδεισο που λέγεται Internet. Για να λειτουργήσουν και συναγωνιστούν αποτελεσματικά, οι προγραμματιστές που θέλουν να ηγηθούν ενός συνεργατικού project πρέπει να μάθουν να στρατολογούν και να ενεργοποιούν αποτελεσματικές ομάδες προγραμματιστών με τον τρόπο που προτείνει η "αρχή της κατανόησης" του Κροπότκιν. Πρέπει να μάθουν να χρησιμοποιούν τον Νόμο του Linus.

Νωρίτερα αναφέρθηκα στο "φαινόμενο των Δελφών" ως μια πιθανή εξήγηση του Νόμου του Linus. Υπάρχουν, όμως, κι άλλες εξηγήσεις όπως οι ισχυρές αναλογίες των προσαρμόσιμων συστημάτων στην βιολογία και την οικονομία. Ο κόσμος του Linux συμπεριφέρεται από πολλές απόψεις σαν μια ελεύθερη αγορά ή ένα οικοσύστημα, σαν ένα σύνολο εγωιστικών δυνάμεων που προσπαθούν να μεγιστοποιήσουν την ωφέλεια η οποία, στην διαδικασία, δημιουργεί μια αυτό-διορθωτική αυθόρμητη τάξη περισσότερο περίτεχνη και αποτελεσματική από κάθε τέτοια που μπορεί να καταφέρει οποιοδήποτε ποσότητα κεντρικού σχεδιασμού. Μετά, ς' αυτό το σημείο πρέπει να αναζητήσουμε την "αρχή της κατανόησης".

Η "ωφέλιμη λειτουργία" που προσπαθούν να μεγιστοποιήσουν οι προγραμματιστές Linux δεν είναι οικονομική αλλά μια απροσδιόριστη ικανοποίηση του εγώ και της φήμης τους μεταξύ των άλλων προγραμματιστών. (Κάποιος μπορεί να ονομάσει το κίνητρό τους "αλτρουιστικό", αλλά θα αγνοεί ότι ο αλτρουισμός είναι ο ίδιος είναι μια μορφή ικανοποίησης του εγώ των αλτρουιστών). Η εθελοντική κουλτούρα που λειτουργεί μ' αυτό τον τρόπο στην πραγματικότητα είναι συνηθισμένη. Μια άλλη κουλτούρα, στην οποία συμμετείχα κι εγώ, είναι οι οπαδοί της επιστημονικής φαντασίας που, αντίθετα με τους προγραμματιστές αναγνωρίζει "τον μπαμπούλα του εγώ" (τον εμπλουτισμό της φήμης κάποιου μεταξύ άλλων οπαδών) σαν το βασικό κίνητρο πίσω απ' την εθελοντική δραστηριότητα.

Έχοντας με επιτυχία βάλει τον εαυτό του στη θέση του φύλακα του project κατά το οποίο η ανάπτυξη του λογισμικού γίνεται από άλλους και καλλιεργώντας το ενδιαφέρον για το project μέχρις ότου αυτό έγινε αυτο-συντηρούμενο, ο Linus επέδειξε μια οξυδερκή αντίληψη της "αρχής της κοινής κατανόησης" του Κροπότκιν. Αυτή η ημι-οικονομική όψη του κόσμου του Linux μας αναγκάζει να εξετάσουμε πώς εφαρμόζεται αυτή η κατανόηση.

Μπορούμε να δούμε τις μεθόδους του Linus σαν ένα τρόπο σύνδεσης του εγωισμού του ξεχωριστού προγραμματιστή όσο το δυνατόν πιο στενής με δύσκολους στόχους, που μπορούν να επιτευχθούν μόνο με παρατεταμένη συνεργασία. Με το project του fetchmail απέδειξα (μολονότι σε μικρότερη κλίμακα) ότι αυτή η μέθοδος μπορεί να επαναληφθεί με καλά αποτελέσματα. Ίσως τα κατάφερα λίγο πιο συνειδητά και συστηματικά από εκείνον.

Πολλοί άνθρωποι (ιδιαίτερα εκείνοι που πολιτικά δυσπιστούν ενώπιον της ελεύθερης αγοράς) θα περίμεναν από μια κουλτούρα αυτό- διευθυνόμενων ατόμων να είναι κερματισμένη, τοπική, άχρηστη, περιττή κι εχθρική. Αυτές οι προσδοκίες, όμως, ακυρώνονται από την εκπληκτική ποικιλία, ποιότητα και βάθος της τεκμηρίωσης του Linux, για να αναφέρω ένα παράδειγμα. Οι προγραμματιστές *απεχθάνονται* την τεκμηρίωση. Πώς, τότε, οι προγραμματιστές του Linux παράγουν τόση πολύ; Προφανώς, η ελεύθερη αγορά στο Linux δουλεύει παρέχοντας καλή, διαφορετική συμπεριφορά από εκείνη των μαζικά χρηματοδοτούμενων εργαστηρίων παραγωγής τεκμηρίωσης των εμπορικών λογισμικών προϊόντων.

Τόσο το project του πυρήνα του fetchmail όσο και του Linux δείχνουν ότι, ανταμείβοντας κατάλληλα τους προγραμματιστές ένας καλός προγραμματιστής/συντονιστής μπορεί να χρησιμοποιήσει το Internet για να εκμεταλλευτεί τα οφέλη όταν έχει πολλούς προγραμματιστές χωρίς, ταυτόχρονα, να καταστρέφεται το project μέσα σε μια χαοτική ανακατωσούρα. Έτσι, στον Νόμο Brooks αντιπροτείνω το εξής:

19. Με τον όρο ο συντονιστής προγραμματιστής να έχει ένα μέσον τουλάχιστον τόσο καλό όσο το Internet και γνωρίζοντας πώς να ηγηθεί χωρίς καταπίεση, πολλά μυαλά αναπόφευκτα γίνονται καλύτερα από ένα.

Πιστεύω ότι, το μέλλον του *λογισμικού ανοιχτού κώδικα* ανήκει όλο και περισσότερο στους ανθρώπους που ξέρουν πώς να παίξουν το παιχνίδι του Linus, σε ανθρώπους που αφήνουν πίσω τους το καθεδρικό στυλ κι αγκαλιάζουν το στυλ παζαριού. Μ' αυτό δεν εννοούμε ότι η άποψη και η ευφυία του καθενός δεν έχουν σημασία. Αλλά μάλλον ότι, η αιχμή του λογισμικού ανοιχτού κώδικα θα ανήκει σε ανθρώπους που ξεκινούν απ' την ατομική άποψη κι ευφυία και μετά τις ενισχύουν με την αποτελεσματική δημιουργία εθελοντικών ομάδων κοινού ενδιαφέροντος.

Και ίσως όχι μόνο το μέλλον του λογισμικού ανοιχτού κώδικα. Κανένας προγραμματιστής κλειστού κώδικα μπορεί να συγκεντρώσει όλα αυτά τα ταλέντα που η κοινότητα του Linux μπορεί να απασχολήσει για την επίλυση κάποιου προβλήματος. Πολλοί λίγοι μπορούν να αντεπεξέλθουν στο οικονομικό έξοδο πρόσληψης το πολύ διακοσίων ανθρώπων, όσων συνεισέφεραν στο fetchmail!

Ίσως, στο τέλος, η κουλτούρα του ανοιχτού κώδικα θριαμβεύσει όχι επειδή η συνεργασία είναι ηθικώς ορθή ή η "φραγή" του λογισμικού είναι ηθικώς λανθασμένη (υποθέτω ότι θα συμφωνείτε με το τελευταίο), αλλά απλά επειδή ο κόσμος του κλειστού κώδικα δεν μπορεί να κερδίσει έναν εξελικτικό αγώνα με τις κοινότητες ανοιχτού κώδικα, πολύ περισσότερο κι έμπειρο χρόνο για την επίλυση ενός προβλήματος.

11 Ευχαριστίες.

Αυτό το κείμενο βελτιώθηκε συζητώντας το με μεγάλο αριθμό ανθρώπων, που βοήθησαν στην αποσαφήνισή του. Ευχαριστώ ιδιαιτέρως τον Jeff Dutky <dutky@wam.umd.edu> που πρότεινε τον κανόνα "το ξεκαθάρισμα είναι παραλληλήσιμο" και βοήθησε στην επεξεργασία της ανάλυσης που εξάγεται από αυτόν. Επίσης, ευχαριστώ την Nancy Lebovitz<nancyl@universe.digex.net> για την πρότασή της ότι μιμούμαι τον Weinberg παραθέτοντας τον Κροπότκιν. Οξύνους κριτική άσκησαν η Joan Eslinger <wombat@kilimanjaro.engr.sgi.com> και η Marty Franz <marty@net-link.net> της λίστας General Technics. O Paul Eggert <eggert@twinsun.com> παρατήρησε την σύγκρουση μεταξύ του ανοιχτού και του GPL μοντέλου. Είμαι ευγνώμων στα μέλη του PLUG, ομάδα Χρηστών Linux της Φιλαδέλφεια, που πρώτοι διάβασαν την πρώτη δημοσίευση του παρόντος. Τέλος, τα σχόλια του Linus Torvalds με βοήθησαν πολύ και με ενθάρρυνε η πρώιμη υποστήριξή του.

12 Για Περαιτέρω Μελέτη.

Παρέθεσα πολλά τμήματα του βιβλίου "*The Mythical Man-Month*", του Frederick P. Brooks. Θερμά προτείνω την έκδοση της 25 επετείου από τις εκδόσεις Addison-Wesley (ISBN 0-201-83595-9) στην οποία έχει προστεθεί το κείμενο του 1986 "*No Silver Bullet*".

Η έκδοση είναι εμπλουτισμένη με μια ανεκτίμητη 20χρονη αναδρομή, στην οποία ο Brooks ευθέως παραδέχεται την κριτική του αρχικού κειμένου το οποίο δεν άντεξε την δοκιμασία του χρόνου. Διάβασα για πρώτη φορά την αναδρομή μετά την ουσιαστική σύνταξη αυτού του κειμένου και εξεπλάγην όταν διαπίστωσα ότι ο Brooks απέδιδε πρακτικές σε παζαριώτικο στυλ στην Microsoft!

Το "*The Psychology of Computer Programming*" του Gerald M. Weinberg εισήγαγε την μάλλον κακώς εννοούμενη αντίληψη του "προγραμματισμού χωρίς εγωισμό". Αν και δεν ήταν ο άνθρωπος που θα αναγνώριζε την ματαιότητα της "αρχής της εντολής", ήταν ίσως ο πρώτος που αναγνώρισε και διαφώνησε με αυτό το σημείο, συνδέοντάς το ιδιαίτερα με την ανάπτυξη λογισμικού.

O Richard P. Gabriel μελετώντας την κουλτούρα του Unix στην προ-Linux εποχή, με απροθυμία διαφωνούσε με την υπεροχή ενός πρωτόγονου παζαριώτικου στυλ στο κείμενό του "*Lisp: Good News, Bad News, and How To Win Big*". Αν και παλαιό από μερικές απόψεις, αυτό το δοκίμιο δικαίως εξυμνείται από τους οπαδούς του Lisp (μεταξύ αυτών κι από εμένα). Ένας αλληλογράφος μου υπενθύμισε ότι το κεφάλαιο με τίτλο "Worse Is Better" είναι σαν πρόβλεψη για το Linux. Το κείμενο βρίσκετε στο Internet, `<http://www.naggum.no/worse-is-better.html>`.

Το βιβλίο *Peopleware: Productive Projects and Teams* (New York; Dorset House, 1987; ISBN 0-932633-05-6), των De Marco και Lister, βιβλίο που δεν εκτιμάται όπως του αναλογεί, το οποίο με ευχαρίστηση είδα να το παραθέτει στην αναδρομή του ο Brooks. Αν και λίγα απ' όσα λένε οι συγγραφείς του είναι εφαρμόσιμα στο Linux ή στον ανοιχτό κώδικα, η βαθιά γνώση τους για τις αναγκαίες συνθήκες για δημιουργική δουλειά είναι οξυδερκής κι αξίζει για κάθε έναν που προσπαθεί να εισάγει κάποιες αξίες του μοντέλου παζαριού στο εμπορικό λογισμικό.

Τέλος, οφείλω να παραδεχτώ ότι είχα ονομάσει αυτό το κείμενο σχεδόν "Ο Καθεδρικός και η Αγορά", όπου ο τελευταίος όρος σημαίνει την Ελληνική απόδοση μιας ανοιχτής αγοράς ή ενός δημόσιου χώρου συνεύρεσης. Οι γόνιμες εργασίες για τα "αγοραία συστήματα" των Marc Miller και Eric Drexler, περιγράφοντας τις εμφανιζόμενες ιδιότητες των αγοραίων υπολογιστικών συστημάτων με βοήθησαν να προετοιμαστώ να σκεφτώ ξεκάθαρα για ανάλογα φαινόμενα στην κουλτούρα του ανοιχτού κώδικα του Linux, τα οποία διαπίστωσα πέντε χρόνια αργότερα. Αυτά τα δοκίμια βρίσκονται στο Internet, `<http://www.agorics.com/agorpapers.html>`.

13 Επίλογος: Η Netscape Αγκαλιάζει το Παζάρι!

Είναι παράξενο να νιώθεις ότι βοηθάς να γραφτεί ιστορία...

Στις 22 Ιανουαρίου 1998, περίπου εφτά μήνες μετά την πρώτη δημοσίευση αυτού του κειμένου, η Netscape Communications, Inc. εξήγγειλε την πρόθεσή της*να δημοσιεύσει τον κώδικα του Netscape Communicator* `<http://www.netscape.com/newsref/pr/newsrelease558.html>`. Μέχρι την ημέρα της εξαγγελίας δεν πίστευα ότι θα γινόταν κάτι τέτοιο.

Ο Eric Hahn, αντιπρόεδρος και Διευθυντής Τεχνολογίας της Netscape μου έστειλε, αμέσως μετά, το εξής μήνυμα: "Στο όνομα όλων υμών στην Netscape, θέλω να σας ευχαριστήσω που μας βοηθήσατε να φτάσουμε πρώτοι ς' αυτό το σημείο. Οι σκέψεις και τα γραπτά σας ήσαν θεμελιώδης εμπνεύσεις για την απόφασή μας".

Την επόμενη εβδομάδα πήγα στην Silicon Valley, μετά από πρόσκληση της Netscape για μια ημερίδα (στις 4 Φεβρουαρίου 1998), μαζί με κορυφαία στελέχη και τεχνικούς. Σχεδιάσαμε την στρατηγική έκδοσης του κώδικα και της άδειας και επεξεργαστήκαμε κάποια άλλα σχέδια που ελπίζουμε ότι τελικά θα έχουν μακρόπνοες και θετικές επιπτώσεις στην κοινότητα του ανοιχτού κώδικα. Είναι πολύ νωρίς για να μπω σε λεπτομέρειες. Σε λίγες εβδομάδες θα υπάρχουν νεότερα.

Η Netscape πρόκειται να μας παραχωρήσει μιας μεγάλης κλίμακας αληθινή δοκιμασία του παζαριώτικου μοντέλου στον εμπορικό κόσμο. Η κουλτούρα του ανοιχτού κώδικα τώρα αντιμετωπίζει ένα κίνδυνο: αν το σχέδιο της Netscape αποτύχει, η ιδέα του ανοιχτού κώδικα θα υποβιβαστεί και ο εμπορικός κόσμος δεν θα την ξανασκεφτεί για άλλη μια δεκαετία.

Από την άλλη, αυτό είναι μια θεαματική ευκαιρία. Η αρχική αντίδραση του Χρηματιστηρίου ήταν συγκρατημένα θετική. Δοκιμαζόμαστε κι εμείς, επίσης. Αν η Netscape κερδίσει σημαντικό μερίδιο αγοράς από αυτή την κίνηση ίσως εκδηλωθεί μια καθυστερημένη επανάσταση στην βιομηχανία υπολογιστών.

Το επόμενο έτος θα είναι πολύ διδακτικό κι ενδιαφέρον.