Open-Source: Η εναλλακτική επιλογή που κερδίζει έδαφος

Χάρης Β. Γεωργίου*

March 24, 2013

Abstract

Μέχρι πριν μερικά χρόνια, η φράση «free / open-source software» (FOSS)[1] ήταν σχεδόν συνώνυμη με μαλλιαρούς φοιτητές που έψαχναν για «τζάμπα κώδικα» για τις εργασίες τους ή με «ψαγμένους κομπιουτεράδες» που ήθελαν να κάνουν δωρεάν κάποια δουλειά τους. Σήμερα, ευτυχώς, τα πράγματα έχουν προχωρήσει αρκετά. Το Ελεύθερο Λογισμικό και Λογισμικό Ανοικτού Κώδικα (ΕΛ/ΛΑΚ) αποτελεί πλέον μια βιώσιμη, δοκιμασμένη και κατά περίπτωση πολύ αποδοτική επιλογή, εναλλακτική διαφόρων άλλων εμπορικών πακέτων και λειτουργικών συστημάτων.

Οι διαστάσεις του ζητήματος

Υπάρχουν γενικά τρεις διαστάσεις να δει κανείς το θέμα του ΕΛ/ΛΑΚ: (α) η εμπορική διάσταση, (β) η άποψη της Τεχνολογίας Λογισμικού (Software Engineering) και (γ) πως το αντιμετωπίζει ο τελικός χρήστης (end-user). Η σημαντικότερη ίσως διαφορά του ΕΛ/ΛΑΚ σε σχέση με το κλασικό «εμπορικό» λογισμικό είναι το (α), δηλαδή πως αυτό εμπλέκεται και λειτουργεί ως εμπορικό προϊόν, αν και δεν είναι πάντα τέτοιο. Το γεγονός ότι *ο κώδικας είναι «ανοικτός»*, δηλαδή διαθέσιμος σε οποιονδήποτε με τις απαραίτητες γνώσεις να τον δει και ίσως (αν είναι επιπλέον open-licensed) να τον τροποποιήσει κιόλας φέρνοντάς τον στα μέτρα του, δεν σημαίνει αυτομάτως ότι είναι και δωρεάν. Φυσικά η εταιρία που τον διαθέτει πρέπει να «πουλάει» κάτι παραπάνω από ένα CD με ωραίο εξώφυλλο, μια και ο καθένας μπορεί ήδη να «κατεβάσει» από κάπου τον κώδικα εφόσον πρόκειται για ΕΛ/ΛΑΚ. Συνεπώς, το μέχρι τώρα εμπορικό μοντέλο «λογισμικό-ως-προϊόν» (software as a product) αντικαθίσταται με το μοντέλο «λογισμικό-ως-υπηρεσία» (software as a service). Δεν είναι καθόλου τυχαίο που οι τεχνολογίες Cloud Computing γνωρίζουν τόσο μεγάλη άνθηση τα τελευταία χρόνια, ενώ ήδη εταιρίες που κλασικά ακολουθούσαν πιστά το πρώτο μοντέλο τώρα στρέφονται σταδιακά και στο δεύτερο: Εκδόσεις του MS-Office 2013 είναι ήδη διαθέσιμες ως cloud-only σε πολύ χαμηλή τιμή. Αυτό που καλείται να πληρώσει ο «πελάτης» είναι είτε η «ενοικίαση» λογισμικού (cloud-based services) είτε η υποστήριξή του σε βάθος χρόνου (support contracting). Οι εταιρίες αρχίζουν να καταλαβαίνουν σταδιακά

*(MSc, PhD) Τμήμα Πληροφορικής & Τηλεπικοινωνιών, Εθνικό Καποδιστριακό Πανεπιστήμιο Αθηνών (ΕΚΠΑ) – Email: xgeorgio@di.uoa.gr

[1] http://en.wikipedia.org/wiki/FOSS

ότι το λογισμικό αποτελεί πολύ ιδιαίτερο τύπο «προϊόντος» και δεν μπορούν να το διαχειρίζονται εμπορικά όπως ακριβώς ένα αυτοκίνητο ή ένα πακέτο μανταλάκια.

Μια δεύτερη διάσταση του ΕΛ/ΛΑΚ που δείχνει πόσο διαφορετικά είναι τα πράγματα σε σχέση με το κλασικό «εμπορικό» λογισμικό είναι αυτή *του τρόπου ανάπτυξής του*. Αν και αυτό δεν σχετίζεται με τη διαθεσιμότητα ή όχι του κώδικα, συνήθως μιλάμε για ομάδες εθελοντών από όλο τον κόσμο, κοινότητες που κατασκευάζουν και υποστηρίζουν τεράστιες βάσεις κώδικα, συχνά πολλών εκατομμυρίων γραμμών σε διάφορες γλώσσες προγραμματισμού και πλατφόρμες υλοποίησης. Εννοείται πως δεν είναι καθόλου εύκολο να βρεθούν όλοι μαζί για καφέ και να κουβεντιάσουν πως πρέπει να είναι η επόμενη έκδοση του πακέτου, αλλά συχνά ούτε καν να ανταλλάξουν emails με τις σχετικές λεπτομέρειες, αφού πρόκειται για ανθρώπους που μιλάνε διαφορετικές γλώσσες, έχουν διαφορετικό πολιτισμό και συνήθειες, έχουν διαφορετικό χρόνο διαθέσιμο και φυσικά αν είναι εθελοντές δεν ανέχονται εύκολα κάποιον να τους δίνει οδηγίες και να τους λέει τι να κάνουν. Το σύστημα αυτό φαίνεται εξαιρετικά χαοτικό και όντως έτσι είναι. Μοιάζει απίθανο να παραχθεί μέσω αυτού κάτι πραγματικά αξιόλογο και αξιόπιστο. Κι όμως, αυτό που για τους περισσότερους είναι το βασικό επιχείρημα εναντίον του ΕΛ/ΛΑΚ, έχει δημιουργήσει μερικά από τα καλύτερα προϊόντα στην ιστορία των Η/Υ. Εκτός από το Linux, το γνωστότερο ίσως παράδειγμα, δεν χρειάζεται να σκεφτεί κανείς πολύ για να εντοπίσει την τετράδα LAMP (Linux-Apache-MySQL-PHP)[2] που αυτή τη στιγμή «τρέχει» περίπου το 70% των websites παγκοσμίως, το Joomla και γενικά τα περισσότερα web content management systems (CMS), μέχρι και τον GNU C/C++ compiler (GCC) που είναι συγκρίσιμος σε απόδοση μόνο με τον αντίστοιχο compiler της «μαμάς» Intel για τους επεξεργαστές x86. Δεν υπάρχει σχεδόν τίποτα που να μην έχει φτιαχτεί ως ΕΛ/ΛΑΚ από κοινότητες χρηστών και μάλιστα πολλές φορές καλύτερα από ότι εταιρίες που πουλάνε αντίστοιχα εμπορικά προϊόντα. Η μέθοδος που «βάζει τάξη στην αταξία» λέγεται Agile, ακολουθείται είτε εκούσια από την αρχή είτε «προκύπτει» αυθόρμητα στην πορεία, ακόμα και μεταξύ προγραμματιστών που δεν έχουν ακούσει ποτέ για αυτή αλλά μαθαίνουν να δουλεύουν από κοινού πολύ αποδοτικά παράγοντας αξιόλογο κώδικα, χωρίς να βασίζονται σε πολύ αυστηρές προδιαγραφές και βαριά τεκμηρίωση (documentation).

Η τρίτη, ίσως η γνωστότερη, διάσταση του ΕΛ/ΛΑΚ είναι αυτή της *οπτικής του τελικού χρήστη* (end-user view). Δεν είναι τυχαίο που ο όρος «ανοικτός κώδικας» για τους περισσότερους είναι συνώνυμο του «δωρεάν», και πράγματι συνήθως έτσι είναι. Μάλιστα, όσο πιο αρχάριος είναι ο χρήστης, τόσο πιο δύσκολο του είναι να καταλάβει πως γίνεται να υπάρχει διαθέσιμο λογισμικό χωρίς κανένα απολύτως κόστος, το οποίο μπορεί να κατεβάσει από κάποιο website και να το τρέξει, χωρίς να χρειάζεται να ψάξει «πειρατικές» πηγές και «σπαστήρια» προγραμμάτων. Η αντίληψη ότι το λογισμικό (από μόνο του) πάντα κοστίζει είναι δυστυχώς μια αντίληψη που έχει ριζωθεί για τα καλά στην κοινή αντίληψη περί Η/Υ και προγραμμάτων. Αυτό που δεν ξέρουν και που συχνά δεν το εξηγεί επαρκώς κανείς είναι ότι, αυτό που αποτελεί βασικό πλεονέκτημα του ΕΛ/ΛΑΚ (μηδενική τιμή) συχνά είναι και το βασικότερο μειονέκτημά του: Αν το λογισμικό δεν σου κάνει, δεν υπάρχει κανείς να τον πάρεις τηλέφωνο και να τον βρίσεις για τα λεφτά που έδωσες. Μέχρι πρόσφατα, οι χρήστες ΕΛ/ΛΑΚ έπρεπε να είναι χρήστες Η/Υ με γνώσεις αρκετά άνω του μετρίου για να μπορέσουν απλά και μόνο να εγκαταστήσουν κάποιο πακέτο. Σήμερα ευτυχώς όλα σχεδόν τα δημοφιλή πακέτα , όπως για παράδειγμα οι πιο γν-

[2] http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29

ωστές διανομές του Linux (Ubuntu, OpenSUSE, Fedora, κτλ)[3], ενσωματώνουν εργαλεία και γραφικά περιβάλλοντα τόσο καλοφτιαγμένα που είναι πλέον προσιτά στον κάθε αρχάριο χρήστη.

Μπορεί να είναι εμπορική επιτυχία;

Σε όλα τα είδη προϊόντων, οι εταιρίες βασίζονται κυρίως σε τέσσερις διαφορετικές επιλογές προώθησης: (α) τη διαφοροποίηση ή εξειδίκευση σε σχέση με τους ανταγωνιστές, (β) στην εξαιρετικά χαμηλή τιμή, (γ) στην ποιοτική διαφορά ή (δ) στην καινοτομία. Επειδή και τα τέσσερα κοστίζουν, οι εταιρίες συνήθως βασίζονται σε ένα από αυτά και σχεδόν ποτέ σε συνδυασμό περισσότερων. Φυσικά, το βασικό πλεονέκτημα του ΕΛ/ΛΑΚ είναι το δεύτερο: Κανείς δεν μπορεί να ανταγωνιστεί σε τιμή αυτό που είναι διαθέσιμο δωρεάν! Από την άλλη, το βασικό μειονέκτημα είναι συχνά το τρίτο, μια και το ΕΛ/ΛΑΚ χαρακτηρίζεται κατά κανόνα από εξαιρετικά χαλαρό έλεγχο ποιότητας (λόγω τρόπου εργασίας) και προδιαγραφών (συχνά δεν υπάρχουν συγκεκριμένες). Στο μοντέλο Agile, αυτό αντιμετωπίζεται με την πολύ καλή συνεργασία των μελών της ομάδας ανάπτυξης, της σταδιακής κατασκευής του λογισμικού σε μικρά διαδοχικά βήματα, καθώς και από το εκτεταμένο testing. Σε ότι αφορά το (α) και το (δ), δηλαδή τη διαφοροποίηση ή/και την καινοτομία, δεν είναι λίγες οι φορές που πακέτα ΕΛ/ΛΑΚ αποτέλεσαν πραγματική επανάσταση (breakthroughs) στον τρόπο που βλέπουμε το λογισμικό και τις υπηρεσίες που προσφέρονται μέσω αυτού. Χαρακτηριστικό παράδειγμα είναι οι ανοικτού κώδικα πλατφόρμες CMS[4] όπως το Joomla[5], οι οποίες άλλαξαν ριζικά την έννοια του «website» και το πως το διαχειριζόμαστε σήμερα.

Σε ότι αφορά το κόστος, είναι πολλές οι χώρες που ήδη εφαρμόζουν βασικές πολιτικές μερικής ή αποκλειστικής χρήσης ΕΛ/ΛΑΚ σε δημόσιες υπηρεσίες και σχολεία, όπως για παράδειγμα η Βραζιλία και άλλες χώρες της Νότιας Αμερικής. Αλλά και στις πιο πλούσιες χώρες του Ευρωπαϊκού Βορρά, όπως η Δανία και η Σουηδία, το ΕΛ/ΛΑΚ θεωρείται η πρώτη και κύρια επιλογή σε ότι έχει σχέση με τις υπηρεσίες του δημοσίου, έτσι ώστε η πρόσβαση των δεδομένων να είναι συμβατή και ανοικτή σε όλους (open standards). Στις υποανάπτυκτες χώρες, όπως αυτές της Υποσαχάριας Αφρικής, διεθνή προγράμματα ICT4D όπως το «One Laptop Per Child» (OLPC/XO)[6] βασίζονται αποκλειστικά σε λογισμικό ΕΛ/ΛΑΚ, αλλά και υλικό (hardware) σχεδιασμένο με την ίδια ακριβώς φιλοσοφία.

Αξίζει να επενδύσει κανείς σε αυτό;

Θα πει κανείς, εντάξει όλα αυτά, αλλά το βασικό ερώτημα είναι ένα και απλό: Υπάρχει πρακτικό όφελος από τη χρήση ΕΛ/ΛΑΚ; Δηλαδή, μπορεί κανείς να γλιτώσει χρήματα χωρίς να θυσιάσει την ποιότητα του λογισμικού που χρειάζεται για τη δουλειά του και φυσικά να μην αποκτήσει νέους πονοκεφάλους και προβλήματα; Όπως σε όλα τα πράγματα, η απάντηση είναι απλή, αλλά δεν είναι σύντομη.

Τα άμεσα πλεονεκτήματα από τη χρήση ΕΛ/ΛΑΚ συνδέονται με όσα περιγράφηκαν παραπάνω σχετικά με το πως βλέπουν οι ίδιες οι εταιρίες το «προϊόν» λογισμικού. Το πιο χαρακτηριστικό πλεονέκτημα είναι ίσως το κόστος: Είναι

[3] http://en.wikipedia.org/wiki/Linux_distribution#Popular_distributions
[4] http://en.wikipedia.org/wiki/Content_management_systems
[5] http://www.joomla.org
[6] http://en.wikipedia.org/wiki/OLPC

μηδενικό! Αυτό και μόνο μπορεί να είναι ικανό κίνητρο μερικές φορές για να εγκαταλειφθεί η λύση ενός πολύ διαδεδομένου εμπορικού πακέτου και να αντικατασταθεί από ένα αντίστοιχο «δωρεάν» ΕΛ/ΛΑΚ, όπως για παράδειγμα η εγκατάσταση συνδυασμού Ubuntu & LibreOffice αντί για MS-Windows & MS-Office. Το πρώτο κοστίζει μηδέν, ενώ το δεύτερο αρκετές εκατοντάδες ως και χιλιάδες ευρώ. Υπάρχει όμως μια σημαντική παράμετρος: Το πρώτο μπορεί να είναι κατάλληλο για ένα σχολικό εργαστήριο ή για τον Η/Υ ενός περιστασιακού χρήστη, ίσως όμως δεν είναι και τόσο αξιόπιστη λύση όταν πρόκειται για εταιρία που απαιτεί συμβόλαια υποστήριξης με διασφαλισμένη ποιότητα (SLA). Η εν λόγω εταιρία μπορεί σαφέστατα να στηριχθεί σε αμιγώς ΕΛ/ΛΑΚ λύση, αλλά μόνο όταν ξέρει ότι πρέπει να έχει ήδη ή να αποκτήσει εξειδικευμένο προσωπικό για να το υποστηρίξει. Όσο εξελιγμένοι κι αν φαίνονται, οι Η/Υ εξακολουθούν να είναι η πιο πολύπλοκη (μεν) μηχανή (δε) που έχει κατασκευαστεί ποτέ. Όπως ακριβώς δεν μπορεί ο οποιοσδήποτε να παίξει σωστά βιολί Στρατιβάριους, έτσι ακριβώς δεν μπορεί οποιοσδήποτε «μάστορης» να υποστηρίξει ένα data center, όσο καλή διάθεση και να έχει.

Αλλά πέρα από τα βασικά πλεονεκτήματα που πιθανόν να έχει κάποιος από τη χρήση ΕΛ/ΛΑΚ, υπάρχουν και αρκετά έμμεσα, που δεν γίνονται αντιληπτά αμέσως. Η λέξη «ανάπτυξη» και «καινοτομία» είναι λέξεις πολύ της μόδας σήμερα, αλλά ελάχιστοι ξέρουν τι ακριβώς εννοούν με αυτές όταν αναφέρονται στο λογισμικό. Μια μικρή οικονομία, όπου δεν υπάρχουν τεράστιες πολυεθνικές εταιρίες βαριάς παραγωγής, βασίζεται κυρίως στη μεταποίηση και στις εξειδικεύσεις - ακριβώς το ίδιο ισχύει και στο λογισμικό. Οι εξατομικευμένες λύσεις, με προσαρμογή στο τοπικό περιβάλλον (localization) διευκολύνεται αφάνταστα όταν πρόκειται για ΕΛ/ΛΑΚ και όχι για προϊόντα κλειστού κώδικα. Αυτό σημαίνει ότι εκατοντάδες μικρομεσαίες εταιρίες γίνονται παραγωγοί, κώδικα και εργαλείων, επειδή ακριβώς η πρόσβαση σε διαδεδομένες πλατφόρμες τύπου ΕΛ/ΛΑΚ είναι εξασφαλισμένη και δεν εξαρτάται από την καλή διάθεση της «μαμάς» εταιρίας που παρέχει κάποιο κλειστό πακέτο στην αγορά μέσω μεταπωλητών (box-movers). Επιπλέον, παρόλο που διαισθητικά ίσως φαίνεται λάθος προσέγγιση, η πραγματικότητα λέει πως η ασφάλεια και η αξιοπιστία εξαρτάται άμεσα από το κατά πόσο είναι διαθέσιμες οι λεπτομέρειες: Όσο πιο ανοικτό είναι το σύστημα (κώδικας), τόσο πιο πιθανό είναι να το ελέγξουν πολλά μάτια και άρα να εντοπιστούν τα όποια προβλήματα νωρίς. Χαρακτηριστικό παράδειγμα είναι το σοβαρότατο σφάλμα που εντοπίστηκε και διορθώθηκε σε μια βιβλιοθήκη κώδικα του Debian (Linux distribution), το οποίο είχε προκληθεί από λάθος τροποποίηση μιας ρουτίνας παραγωγής κρυπτογραφικών κλειδιών. Πολλοί επικρίνουν το γεγονός με το σκεπτικό ότι ακριβώς επειδή το ΕΛ/ΛΑΚ ευνοεί τέτοιου είδους κρίσιμα σφάλματα από αρχάριους προγραμματιστές, άλλοι όμως επισημαίνουν ότι αν ο κώδικας ήταν κλειστός ως εμπορικό προϊόν κάποιας εταιρίας θα ήταν σχεδόν αδύνατο το σφάλμα να εντοπιστεί και να διορθωθεί εγκαίρως πριν προκληθεί ακόμα μεγαλύτερη ζημιά σε επίπεδο ασφάλειας.

Φυσικά όταν κάτι πάει στραβά, πελάτες και προμηθευτές ενδιαφέρονται περισσότερο για το ποιος ευθύνεται νομικά (ώστε να αρχίσει ο χαρτοπόλεμος με μηνύσεις και αγωγές), παρά με το πως δημιουργήθηκε το πρόβλημα και πως θα διορθωθεί. Ο κλειστός κώδικας κάνει δύσκολο το διαχωρισμό της ευθύνης (liabilities) του κατασκευαστή από αυτή του χρήστη, καθώς ο πρώτος μπορεί να κρύψει τις όποιες ατέλειες ή τα προβλήματα του λογισμικού του, ενώ ο δεύτερος μπορεί να επικαλεστεί το επιχείρημα του «μαύρου κουτιού» (δεν-ξέρω-πως-λειτουργεί-αυτό-το-πράγμα). Με το ΕΛ/ΛΑΚ τα πράγματα γίνονται πιο ξεκάθαρα και ο καθένας

αναγκάζεται να αναλάβει τις ευθύνες του, καθώς ο μεν κατασκευαστής δίνει ένα προϊόν «διάφανο» στον καθένα ειδικό να το εξετάσει σε κάθε του λεπτομέρεια, ενώ ο τελικός χρήστης δεν μπορεί πλέον να βολεύεται στο εύκολο τηλεφώνημα στην εταιρία υποστήριξης με το παραμικρό. Η δουλειά των δικηγόρων, βέβαια, γίνεται αρκετά πιο δύσκολη τώρα, γιατί τα αντίστοιχα συμβόλαια πρέπει να είναι περισσότερο τεχνικά και λιγότερο δικολαβίστικα, αλλά αυτό δεν είναι και τόσο δυσάρεστο για όλους τους υπόλοιπους (το αντίθετο μάλιστα!) που αναγκάζονται να τα ακολουθούν έτσι κι αλλιώς κατά γράμμα.

Μαζί με το ίδιο το λογισμικό, σχεδόν πάντα «ανοίγουν» και τα αντίστοιχα δεδομένα και τα πρωτόκολλα επεξεργασίας τους, πράγμα που σημαίνει ότι από αμιγώς κλειστά formats τα δεδομένα αποθηκεύονται και διακινούνται πλέον με τρόπο απόλυτα συμβατό με όλα σχεδόν τα συστήματα. Χαρακτηριστικό παράδειγμα είναι η εφαρμογή της XML ως βασικό εργαλείο «συμβατής» text-based αποθήκευσης οποιουδήποτε άλλου format αρχείων ή πρωτοκόλλων δικτύων, έτσι ώστε να είναι δυνατή η εύκολη διασύνδεση μεταξύ ασύμβατων γενικά συστημάτων. Η οπτική αυτή είναι ιδιαίτερα σημαντική όταν πρόκειται για δεδομένα που αφορούν πολλούς διαφορετικούς φορείς και οργανισμούς (π.χ. τις δημόσιες υπηρεσίες), καθώς και τομείς όπου η εύκολη διάθεση και ανταλλαγή δεδομένων είναι ένας από τους πιο κρίσιμους παράγοντες επιτυχίας, όπως για παράδειγμα στην έρευνα και ανάπτυξη (research & development - R&D).

Τέλος, σε ότι αφορά το ίδιο το λογισμικό και τον τρόπο ανάπτυξής του, η προσέγγιση Agile, που όπως αναφέρθηκε παραπάνω ταιριάζει απόλυτα με την πρακτική της σταδιακής ανάπτυξης από ομάδες, προσφέρει σημαντικά πλεονεκτήματα έναντι των πιο παραδοσιακών μεθόδων (π.χ. Waterfall) αν εφαρμοστεί σωστά. Ίσως το πιο χαρακτηριστικό είναι η δυνατότητα το λογισμικό να κατασκευάζεται μέσω μιας σειράς μικρών, σταδιακών βελτιώσεων, ουσιαστικά «χτίζεται» κομμάτι-κομμάτι ενώ βρίσκεται ήδη σε χρήση, δοκιμαστική ή κανονική. Πρακτικά, ο τελικός χρήστης έχει ένα πακέτο λογισμικού το οποίο από την πρώτη στιγμή της δημιουργίας του είναι διαθέσιμο, με κάποιες λίγες δυνατότητες και λειτουργίες, οι οποίες όμως συνεχώς βελτιώνονται και επαυξάνονται, συνήθως μέσω άμεσων ενημερώσεων μέσω δικτύου (online updates). Το μοντέλο αυτό αποτελεί σήμερα το καθιερωμένο πρότυπο για μια σειρά πολύ επιτυχημένων πακέτων ΕΛ/ΛΑΚ, από το Mozilla Firefox μέχρι την πλατφόρμα Wordpress (CMS). Αξίζει να σημειωθεί ότι σε λογισμικό που αναπτύσσεται με πιο παραδοσιακό τρόπο, όπως π.χ. Waterfal αντί Agile, για κάθε νέο update απαιτούνται πολύ πιο χρονοβόροι κύκλοι ανάπτυξης που περιλαμβάνουν πλήρη ανάλυση/σχεδίαση, επικύρωση, κτλ - για το λόγο αυτό άλλωστε σε αυτές τις περιπτώσεις προτιμάται σχεδόν πάντα η λύση των service packs αντί των εκτεταμένων online updates.

Κάπου εδώ θα ρωτήσει κάποιος: Όλα είναι τέλεια με το ΕΛ/ΛΑΚ; Δεν υπάρχει κανένα μειονέκτημα; Ασφαλώς και υπάρχουν, αρκετά. Μερικά αναφέρθηκαν ήδη παραπάνω. Το γεγονός και μόνο ότι το μοντέλο ανάπτυξης του λογισμικού συνήθως είναι πολύ πιο «χαλαρό» από ότι σε ένα κλειστό εμπορικό προϊόν αποτελεί επαρκές κίνητρο (liability risk) για μια εταιρία-πελάτη ή για μια εταιρία-κατασκευαστή να στραφεί σε πιο παραδοσιακές λύσεις. Πολλοί είναι οι χρήστες που ευχαρίστως θα δοκίμαζαν την τύχη τους με το Ubuntu (Linux), αλλά ελάχιστοι θα δεχόντουσαν μια ανάλογη «δοκιμή» όταν ήξεραν πως το σύστημα ελέγχου του αεροπλάνου μέσα στο οποίο βρίσκονται έχει κατασκευαστεί εξ' ολοκλήρου από μια ομάδα καλόπιστων μεν ερασιτεχνών δε εθελοντών προγραμματιστών. Αντίστοιχα, ένας αρκετά έμπειρος προγραμματιστής δεν θα είχε κανένα πρόβλημα να επιλέξει (ΕΛ/ΛΑΚ) Apache & MySQL αντί (Microsoft) IIS & MSSQL ως βασική πλατ-

φόρμα για σοβαρή εγκατάσταση web server, ενώ ένας αρχάριος πωλητής-με-χρέη-admin θα προτιμούσε το δεύτερο μαζί με ένα καλό συμβόλαιο support 24ωρης διαθεσιμότητας. Αυτό δεν σημαίνει ότι η επιλογή του ενός ή του άλλου μοντέλου λέει κάτι για το επίπεδο των ανθρώπων που τα χρησιμοποιούν, αλλά σίγουρα λέει πολλά για τις απαιτήσεις σε επίπεδο γνώσεων και in-house εξειδίκευσής τους στο χώρο εργασίας τους.

Ένα πρόσθετο ζήτημα προβληματικής εφαρμογής λύσεων ΕΛ/ΛΑΚ είναι σε περιπτώσεις όπου το «κλειστός» αποτελεί αναπόσπαστο κομμάτι του «ασφαλές». Για παράδειγμα, σε ένα λειτουργικό σύστημα για στρατιωτική χρήση, όπου οι απαιτήσεις ασφάλειας είναι επιπέδου A1 (υψηλότερη δυνατή), η εύκολη υποστήριξη από κάποιο help desk ή ακόμα και η ευχρηστία του συστήματος δεν αποτελούν πρώτη προτεραιότητα. Αντίθετα, το να κρατηθεί μυστική κάθε εσωτερική λεπτομέρεια του λογισμικού, ειδικά αν περιέχει σφάλματα, συχνά είναι ζήτημα ζωής ή θανάτου. Υπάρχουν, βέβαια, αρκετά παραδείγματα τέτοιων συστημάτων τα οποία, ακριβώς επειδή οι κατασκευαστές τους θεώρησαν τον όρο «κλειστό» ισοδύναμο με το «ασφαλές» εκτέθηκαν ανεπανόρθωτα, όπως για παράδειγμα το (πολύ εύκολο όπως φάνηκε) σπάσιμο του πρωτοκόλλου επικοινωνιών και τηλεμετρίας των drones της USAF, που είχαν σαν αποτέλεσμα όχι μόνο την υποκλοπή των video feeds αλλά και την «κλοπή» τους με προσγείωση σε εχθρικά αεροδρόμια (GPS hijacking)[7]. Αν ο αντίστοιχος κώδικας και τα πρωτόκολλά τους ήταν εξ' αρχής ανοικτά πιθανότατα τα σχεδιαστικά ελαττώματα και οι ελλείψεις ασφάλειας θα είχαν εντοπιστεί εγκαίρως, αλλά βέβαια ίσως να είχαν δώσει ακόμα περισσότερα εργαλεία σε αυτούς που ήθελαν να εκμεταλλευτούν ακριβώς αυτά τα ελαττώματα.

Και τώρα τι;

Δεν χρειάζεται πολύ σκέψη για να καταλάβει κάποιος γιατί το λογισμικό τύπου ΕΛ/ΛΑΚ είναι εσκεμμένα υπο-εκτιμημένο και υπο-προβεβλημένο παγκοσμίως, κυρίως με ευθύνη των εταιριών που κατασκευάζουν αντίστοιχα, αμιγώς εμπορικά, «κλειστά» προϊόντα. Όταν ο βασικός και συχνά μοναδικός στόχος είναι το κέρδος, ο χειρότερος αντίπαλος είναι το «δωρεάν». Παρόλα αυτά, είναι γεγονός ότι το ΕΛ/ΛΑΚ είναι κατάλληλο για πολλές, αλλά όχι για όλες τις περιπτώσεις, καθώς εξαρτάται από τις απαιτήσεις συμβολαίων υποστήριξης ή/και διασφάλισης ποιότητας (σε νομικό επίπεδο), καθώς επίσης και από το επίπεδο γνώσεων των χρηστών (αν πρόκειται για κάποια λιγότερο «φιλική» εφαρμογή). Συνεπώς, το «δωρεάν» δεν είναι ούτε και πρέπει να καθορίζεται ποτέ ως το μοναδικό κριτήριο επιλογής λύσεων ΕΛ/ΛΑΚ.

Η έννοια του ΕΛ/ΛΑΚ συνδέει βέβαια δύο πράγματα που δεν ταυτίζονται κατ' ανάγκη: Αυτό του «ελεύθερου λογισμικού» (ΕΛ) και του «λογισμικού ανοικτού κώδικα» (ΛΑΚ). Το πρώτο είναι δωρεάν, ως «ελεύθερο», αλλά δεν σημαίνει αυτόματα ότι πάντα συνοδεύεται από κώδικα που είναι πλήρως διαθέσιμος στον οποιονδήποτε. Αντίστοιχα, το δεύτερο είναι μεν «ανοικτό» εξ' ορισμού, αλλά δεν σημαίνει πως είναι κατ' ανάγκη και «ελεύθερο» (δωρεάν). Χρειάζεται ιδιαίτερη προσοχή στο τι ακριβώς είναι και τι δεν είναι ένα λογισμικό ΕΛ/ΛΑΚ, γιατί από αυτό μπορεί να εξαρτώνται πολύ περισσότερα πράγματα από το κόστος του.

Δυστυχέστατο παράδειγμα: Η γλώσσα προγραμματισμού Java ξεκίνησε το 1992 από τη Sun Microsystems ως ένα open-source project, ως μια γλώσσα

[7] http://en.wikipedia.org/wiki/Iran%E2%80%93U.S._RQ-170_incident

αμιγώς αντικειμενοστραφής (OOP), με πολύ καλή δομή και μηχανισμούς προστασίας του προγραμματιστή από σφάλματα, και με κυριότερο πλεονέκτημα την ανεξαρτησία του εκτελέσιμου προγράμματος από το λειτουργικό σύστημα ή τη συσκευή (platform-independence). Για περίπου 15 χρόνια αποτέλεσε ένα από τα βασικά εργαλεία προγραμματισμού για διαδικτιακές εφαρμογές και όχι μόνο, ενώ σήμερα θεωρείται μαζί με τη (C/C++) η πιο δημοφιλής επιλογή για ανάπτυξη λογισμικού γενικής χρήσης. Σταδιακά, όμως, άρχισε να γίνεται όλο και πιο «κλειστή», καθώς έγινε προστατευόμενο εμπορικό προϊόν (trademarked) και πριν από 3 χρόνια, με την εξαγορά της Sun Microsystems από την Oracle, κατέληξε στα αζήτητα της δεύτερης, αφού δεν έδειξε ενδιαφέρον να τη στηρίξει ως βασικό προϊόν της. Το αποτέλεσμα: Εδώ και ένα μήνα περίπου ανακαλύφθηκαν πάνω από 50 «θαμμένα» σοβαρότατα σφάλματα ασφάλειας (zero-day exploits)[8], τα οποία μάλιστα οι ειδικοί λένε πως είναι αδύνατο να διορθωθούν εντελώς σε διάστημα συντομότερο από 1-2 χρόνια(!)[9] και συνεπώς συμβουλεύουν το περίπου 1 δισεκατομμύριο χρήστες της Java να την απενεργοποιήσουν (web plug-ins) ή να την απεγκαταστήσουν εντελώς. Δεν είναι βέβαιο κατά πόσο η επιλογή του κλειστού κώδικα συνέβαλε σε αυτή την τραγική κατάληξη μιας καθ' όλα επιτυχημένης γλώσσας προγραμματισμού, όμως είναι βέβαιο ότι αν διατηρούνταν εντελώς ανοικτός (όπως άλλωστε ήταν τα πρώτα χρόνια) αυτά τα προβλήματα πιθανότατα θα είχαν εντοπιστεί εγκαίρως και δεν θα απαιτούνταν τόσο δραστικά μέτρα.

[8] http://www.infoworld.com/t/java-programming/java-zero-day-holes-appearing-the-rate-of-one-day-213898

[9] http://www.networkworld.com/community/blog/oracle-releases-emergency-java-patch-experts-warn-flaws-may-take-2-years-fix