Απαιτείται μια ομάδα: Μέγεθος & συνάθροιση

(Τεχνολογία Ανάπτυξης Λογισμικού ως Υπηρεσίας §10.1)

David Patterson

Η τεχνολογία λογισμικού πλέον είναι ομαδικό άθλημα

- Τώρα βρισκόμαστε σε μια εποχή πέρα από τον προγραμματιστή-υπερήρωα
- Έχει ανέβει ο πήχης λειτουργικότητας/ ποιότητας => δεν μπορεί να κάνει κάποιος μόνος του την μεγάλη καινοτομία
- Επιτυχημένη καριέρα στην τεχνολογία λογισμικού => καταμερισμός προγραμματιστικών εργασιών ΚΑΙ καλή συνεργασία με άλλους ΚΑΙ βοηθάει την *ομάδα* να κερδίσει
- *«Δεν υπάρχουν νικητές σε μια χαμένη ομάδα και ηττημένοι σε μια νικήτρια ομάδα.»*

– Fred Brooks, Jr.

Παράδειγμα: Ηλεκτρονικά παιχνίδια

Παιχνίδι	Έτος	Πλατφόρμα	Προγρ. ομάδα
Space Invaders	1981	Arcade console	1
Super Mario Bros.	1985	NES (Nintendo)	8
Sonic the Hedgehog	1999	Sega Dreamcast	30
Resident Evil 6	2013	PC, PS3, Xbox 360	600

Εναλλακτική οργάνωση ομάδας;

- Η προσέγγιση σχεδιασμού και τεκμηρίωσης απαιτεί εκτεταμένη τεκμηρίωση και σχεδιασμό και εξαρτάται από έναν πεπειραμένο διευθυντή
- Πώς θα πρέπει να οργανώσουμε μια Ευέλικτη ομάδα;
- Υπάρχει εναλλακτική της ιεραρχικής οργάνωσης με έναν διευθυντή που επιβλέπει το έργο;

Συνάθροιση: Οργάνωση ομάδας


- Μέγεθος ομάδας για «2 πίτσες» (4 με 9 άτομα)
- Ο όρος «συνάθροιση» (scrum) προέρχεται από τις συχνές σύντομες συναντήσεις
  - 15 λεπτά κάθε μέρα στο ίδιο μέρος και ώρα
  - Για να μάθετε περισσότερα: *Agile Software Development with Scrum* από τους Schwaber & Beedle

Καθημερινή ατζέντα συνάθροισης



- Απάντησε 3 ερωτήσεις στις «καθημερινές συναθροίσεις»:
  1. Τι έχεις κάνει από χθες;
  2. Τι σχεδιάζεις να κάνεις σήμερα;
  3. Υπάρχουν δυσκολίες ή εμπόδια;
- Βοήθησε κάθε άτομο προσδιορίζοντας τι χρειάζεται

Ρόλοι συνάθροισης

- Ομάδα: μέγεθος ομάδας για 2 πίτσες που παραδίδει λογισμικό
- ScrumMaster: μέλος ομάδας που
  - Ενεργεί ως ενδιάμεσος μεταξύ της ομάδας και των εξωτερικών περισπασμών
  - Διατηρεί την ομάδα επικεντρωμένη στην τρέχουσα εργασία
  - Επιβάλλει κανόνες ομάδας (πρότυπο συγγραφής κώδικα)
  - Υπερνικά δυσκολίες που εμποδίζουν την πρόοδο της ομάδας

8

8

Ρόλοι συνάθροισης (συνέχεια)

- Ιδιοκτήτης προϊόντος: Ένα μέλος ομάδας (όχι ο ScrumMaster) που αναπαριστά την άποψη του πελάτη και θέτει προτεραιότητες στις ιστορίες χρηστών

9

9

Επίλυση διενέξεων

- π.χ. Διαφορετικές απόψεις για τη σωστή τεχνική κατεύθυνση

1. Πρώτα βάλτε σε μια λίστα όλα τα θέματα στα οποία συμφωνούν οι πλευρές
   - αντί να ξεκινήσετε με μια λίστα διαφωνιών
   - Μήπως βλέπετε ότι συμφωνείτε περισσότερο από όσο νομίζετε;
2. Κάθε ομάδα διατυπώνει τα επιχειρήματα της άλλης, ακόμη και αν δεν συμφωνεί με μερικά
   - Αποφεύγεται η σύγχυση σχετικά με όρους ή παραδοχές, που μπορεί να είναι η πραγματική αιτία της διένεξης

10

Επίλυση διενέξεων

3. Εποικοδομητική αντιπαράθεση (Intel)
   - Αν έχεις μια ισχυρή άποψη ότι ένα άτομο προτείνει το λάθος πράγμα από τεχνική άποψη, είσαι υποχρεωμένος να το αναφέρεις, ακόμη και αν πρόκειται για τους προϊσταμένους σου
4. Διαφώνησε και δεσμεύσου (Intel)
   - Όταν παρθεί μια απόφαση, πρέπει να τη δεχτείς και να προχωρήσεις
   - «Διαφωνώ, αλλά θα βοηθήσω ακόμη και αν δεν συμφωνώ».

- Η επίλυση διενέξεων βοηθά και στην προσωπική ζωή!

11

Περίληψη συνάθροισης

- Βασικά, αυτό-οργανωμένη μικρή ομάδα με καθημερινές σύντομες συναντήσεις στα όρθια
- Δουλεύει με «σπριντ» των 2-4 εβδομάδων
- Προτείνεται τα μέλη να αναλαμβάνουν ρόλους εκ περιτροπής (ειδικά του Ιδιοκτήτη Προϊόντος) σε κάθε επανάληψη

Συμβουλές για επιτυχία από τους συναδέλφους σας

Armando Fox

#3. Συνάθροιση/Συναντήσεις

- «Οι πεντάλεπτες καθημερινές συναντήσεις πραγματικά μας βοήθησαν να μείνουμε επικεντρωμένοι στον στόχο μας, και να ανταλλάσσουμε πληροφορίες όταν κολλούσαμε»
- «Η μεγαλύτερη πρόκληση για εμάς ήταν η επικοινωνία/συντονισμός ομάδας»
- «Να υπάρχει ένας καθοδηγητής συνάθροισης κάθε φορά, και να αλλάζει εκ περιτροπής η θέση»
- «1 συνάντηση την εβδομάδα δεν είναι αρκετή»

14

Ποια πρόταση για τις ομάδες είναι ΑΛΗΘΗΣ;

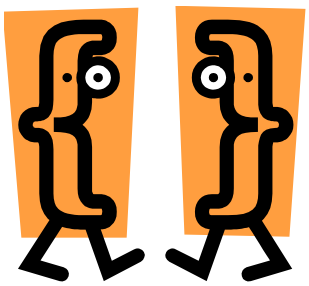
- ☐ Αν συγκρίνουμε την προσέγγιση Σ&Τ με το Scrum, ο διευθυντής έργου στον Σ&Τ ενεργεί και ως Scrum Master και ως Ιδιοκτήτης Προϊόντος
- ☐ Οι ομάδες θα πρέπει να αποφεύγουν τις διενέξεις μεταξύ των μελών τους με κάθε κόστος
- ☐ Ο Σ&Τ έχει πολύ μεγαλύτερες ομάδες από το Scrum, με τις ομάδες να δίνουν αναφορά απευθείας στον διευθυντή έργου
- ☐ Καθώς μελέτες δείχνουν ότι το 84%-90% των έργων ολοκληρώνονται εντός χρόνου και προϋπολογισμού, οι διευθυντές στον Σ&Τ μπορούν με βεβαιότητα να υπόσχονται στους πελάτες ένα σύνολο χαρακτηριστικών για ένα συμφωνημένο κόστος μέχρι μια συμφωνημένη ημερομηνία

15

15

ΤΕΛΟΣ

Προγραμματισμός σε ζεύγος



(Τεχνολογία Ανάπτυξης Λογισμικού ως Υπηρεσίας §10.2)

David Patterson

Είναι 2 μυαλά καλύτερα από 1;

- Στερεότυπο: ο μοναχικός λύκος που δουλεύει όλο το βράδυ πίνοντας Red Bull
- Υπάρχει πιο κοινωνικός τρόπος προγραμματισμού;
  - Ποια θα ήταν τα οφέλη από το να προγραμματίζουν πολλά άτομα μαζί;

- Πώς θα αποτρέπατε την κατάσταση όπου ένα άτομο να κάνει όλη τη δουλειά και οι άλλοι πίνουν καφέδες και χαζεύουν στο Facebook;

18

Προγραμματισμός σε ζεύγος

- Σκοπός: βελτίωση της ποιότητας λογισμικού, μείωση του χρόνου ολοκλήρωσης με 2 άτομα να αναπτύσσουν τον ίδιο κώδικα
  - Μερικά άτομα (και εταιρείες) το λατρεύουν
  - Συμμετάσχετε σε συζητήσεις για να δείτε αν σας αρέσει
  - Μερικοί σπουδαστές στο παρελθόν το λάτρευαν και το χρησιμοποιούσαν σε έργα

19

Προγραμματισμός σε ζεύγος


- Κάθονται δίπλα-δίπλα κοιτώντας την οθόνη μαζί
  - Δεν υπάρχει ένας προσωπικός υπολογιστές – υπάρχουν πολλοί που χρησιμοποιούνται από ένα ζεύγος
  - Για την αποφυγή περισπασμών, δεν υπάρχει πρόγραμμα ηλεκτρονικού ταχυδρομείου ή φυλλομετρητής

20

20

Προγραμματισμός σε ζεύγος

- Ο οδηγός καταχωρίζει κώδικα και σκέφτεται μεθοδικά πώς να ολοκληρώσει την τρέχουσα εργασία, εξηγώντας τις σκέψεις του ενώ πληκτρολογεί
- Ο παρατηρητής αναθεωρεί κάθε γραμμή κώδικα καθώς πληκτρολογείται, και ενεργεί ως δίκτυ ασφαλείας για τον οδηγό
- Ο παρατηρητής σκέφτεται στρατηγικά για τα μελλοντικά προβλήματα, δίνει συμβουλές στον οδηγό
- Θα πρέπει να υπάρχει πολύ συζήτηση και συγκέντρωση
- **Το ζεύγος εναλλάσσει ρόλους**

21

21

Αξιολόγηση προγραμματισμού σε ζεύγος

- Ο ΠΖ γρηγορότερος όταν η πολυπλοκότητα της εργασίας είναι μικρή
- Με τον ΠΖ επιτυγχάνεται υψηλότερη ποιότητα όταν η πολυπλοκότητα είναι μεγαλύτερη
  - Ανεπίσημα, μερικές φορές παράγεται *πιο αναγνώσιμος κώδικας*
- Αλλά απαιτείται περισσότερη προσπάθεια από τους προγραμματιστές που εργάζονται μόνοι τους;
- Επιπλέον μεταφέρει γνώση μεταξύ του ζεύγους
  - προγραμματικοί ιδιωματισμοί, κόλπα με εργαλεία, εταιρικές διαδικασίες, τελευταίες τεχνολογίες…
  - Μερικές ομάδες σκόπιμα εναλλάσσουν *συνεργάτες* ανά εργασία => τελικά όλοι δημιουργούν ζεύγος με όλους («ανάμικτο ζευγάρωμα»)

22

22

Τα *πρέπει* και *δεν πρέπει* στον προγραμματισμό σε ζεύγος

- Δεν πρέπει να ασχολείστε με το κινητό σας όταν είστε παρατηρητής
- Πρέπει να σκεφτείτε το ενδεχόμενο δημιουργίας ζεύγους με κάποιον που έχει διαφορετικό επίπεδο πείρας—θα μάθετε και οι δύο!
  - Οι επεξηγήσεις είναι ένας εξαιρετικός τρόπος για καλύτερη κατανόηση
- Πρέπει να εναλλάσσετε συχνά ρόλους—η μάθηση είναι αμφίδρομη, και κάθε ρόλος απαιτεί διαφορετικές δεξιότητες
  - Ο παρατηρητής μαθαίνει να εξηγεί το σκεπτικό του στον οδηγό

23

23

Ποια πρόταση σχετικά με τον προγραμματισμό σε ζεύγος είναι ΑΛΗΘΗΣ;

- ☐ Ο προγραμματισμός σε ζεύγος είναι ταχύτερος, καλύτερης ποιότητας, φθηνότερος, και απαιτεί λιγότερη προσπάθεια από ό,τι ο μοναχικός προγραμματισμός
- ☐ Ο οδηγός εργάζεται στην τρέχουσα εργασία, ο παρατηρητής σκέφτεται στρατηγικά για τις μελλοντικές εργασίες
- ☐ Κάθε ζεύγος τελικά θα διαπιστώσει ποιος είναι καλύτερος οδηγός και ποιος καλύτερος παρατηρητής, και θα διατηρήσει αυτούς τους ρόλους
- ☐ Το «ανάμικτο ζευγάρωμα» είναι μια μακροπρόθεσμη λύση στο πρόβλημα της έλλειψης προγραμματιστών

24

#6. Προγραμματισμός σε ζεύγος

- «Μας βοήθησε να αποφύγουμε χαζά λάθη που θα χρειάζονταν καιρό να διορθωθούν»
- «Η συχνή εναλλαγή συνεργατών έκανε την ομάδα πιο συνεκτική»

ΤΕΛΟΣ

40 Χρόνια ελέγχου εκδόσεων

SCCS & RCS (1970s)

CVS (1986)

Subversion (2001)

Git (2005)

Image © TheSun.au

Αναθεωρήσεις σχεδιασμού, αναθεωρήσεις κώδικα, προοπτική σχεδιασμού και τεκμηρίωσης στην διαχείριση έργου

(Τεχνολογία Ανάπτυξης Λογισμικού ως Υπηρεσίας §10.3, §10.7-10.9)

David Patterson

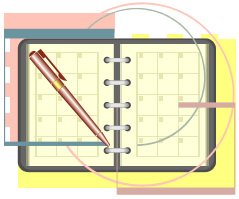
28

Αναθεωρήσεις σχεδιασμού/κώδικα

- Αναθεώρηση σχεδιασμού: συνάντηση κατά την οποία η συγγραφείς παρουσιάζουν τον σχεδιασμό
  - όφελος από την πείρα των παρευρισκόμενων
- Αναθεώρηση κώδικα: γίνεται μετά την υλοποίηση του σχεδιασμού

29

Ατζέντα αναθεώρησης



- Προετοίμασε λίστα ερωτήσεων/θεμάτων για συζήτηση
- Ξεκίνα με μια υψηλού επιπέδου περιγραφή των επιθυμιών του πελάτη
- Παρουσίασε την αρχιτεκτονική λογισμικού, δείχνοντας τα API και επισημαίνοντας τα σχεδιαστικά υποδείγματα (Κεφ. 11) σε κάθε επίπεδο αφαίρεσης
- Εξήγησε τον κώδικα *και* την τεκμηρίωση: σχέδιο έργου, χρονοδιάγραμμα, σχέδιο ελέγχου…: *επαλήθευση & επικύρωση (Verification &Validation, V&V*) του έργου

30

Καλές συναντήσεις: SAMOSAS

(Φωτογραφία του K.S. Poddar. Χρησιμοποιείται κατόπιν άδειας CC-BY-SA-2.0.)

- Η συνάντηση ξεκινά και τελειώνει εγκαίρως (Start)
- Η ατζέντα δημιουργείται εκ των προτέρων – χωρίς ατζέντα δεν υπάρχει συνάντηση (Agenda)
- Καταγράφονται τα πρακτικά ώστε όλοι να θυμούνται τα αποτελέσματα (Minutes)
- Μιλάει ένας κάθε φορά – δεν διακόπτεται ο ομιλητής (One)
- Αποστολή του υλικού εκ των προτέρων, αφού η ανάγνωση είναι πιο γρήγορη (Send)
- Στοιχεία δράσης στο τέλος της συνάντησης ώστε ο καθένας να ξέρει τι θα κάνει ως αποτέλεσμα της συνάντησης (Action)
- Ορισμός της ημερομηνίας και ώρας της επόμενης συνάντησης (Set)

Τα πρακτικά και τα σημεία δράσης καταγράφουν τα αποτελέσματα της συνάντησης, ξεκινήστε την επόμενη συνάντηση με ανασκόπηση των στοιχείων ενεργειών

31

Καλύτερες αναθεωρήσεις;

- Shalloway*: οι επίσημες αναθεωρήσεις σχεδιασμού και κώδικα συχνά γίνονται πολύ αργά στη διαδικασία για να έχουν σημαντική επίδραση
- Καλύτερα έγκαιρες, συντομότερες συναντήσεις; «αναθεωρήσεις προσέγγισης».
  - Μερικοί πεπειραμένοι προγραμματιστές βοηθούν την ομάδα να βρει μια προσέγγιση για να λύσει ένα πρόβλημα
  - Η ομάδα ανταλλάσσει ιδέες για διαφορετικές προσεγγίσεις
- Αν πρέπει να κάνετε μια επίσημη αναθεώρηση σχεδιασμού, προτείνει πρώτα να κάνετε μια «μίνι αναθεώρηση σχεδιασμού» για προετοιμασία

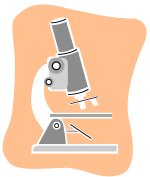
*Alan Shalloway, *Agile Design and Code Reviews*, 2002, www.netobjectives.com/download/designreviews.pdf

32

32

Ποσοτικά μετρικά και αναθεωρήσεις;

- Μελέτη πολλών έργων για καταγραφή μέσων όρων, ορισμό βάσης για νέα έργα, σύγκριση με αυτό:
  - Μέγεθος κώδικα (KLOC), προσπάθεια (μήνες)
  - Ολοκληρωμένα ορόσημα, περιπτώσεις ελέγχου
  - Ανακάλυψη ελαττωμάτων, ρυθμός επιδιορθώσεων/μήνα
- Συσχετίζονται ώστε να μπορούν να αντικαταστήσουν τις αναθεωρήσεις;;



Όμως, βρισκόμαστε πολύ μακριά από την ιδανική κατάσταση, και δεν υπάρχουν ενδείξεις ότι η αυτοματοποιημένη αξιολόγηση ποιότητας θα γίνει πραγματικότητα στο προβλέψιμο μέλλον — Sommerville 2010

33

33

Ευέλικτη προσέγγιση & αναθεωρήσεις;

- Pivotal Labs – Ο προγραμματισμός σε ζεύγος σημαίνει συνεχής αναθεώρηση => χωρίς ειδικές αναθεωρήσεις
- GitHub – *Αιτήσεις έλξης (Pull Requests)* αντί για αναθεωρήσεις
  - Ο προγραμματιστής τη συγχώνευση του κώδικά του με την κύρια βάση κώδικα
  - Όλοι οι προγραμματιστές βλέπουν κάθε αίτηση & αποφασίζουν πώς θα επηρεάσει τον δικό τους κώδικα
  - Αν υπάρχουν δισταγμοί, γίνεται διαδικτυακή συζήτηση για την αίτηση έλξης (pull request)
  - Όπως συμβαίνει καθημερινά, οι μίνι αναθεωρήσεις είναι συνεχείς, χωρίς ειδικές αναθεωρήσεις

34

Ποια πρόταση σχετικά με τις αναθεωρήσεις και συναντήσεις είναι ΛΑΘΟΣ;

- ☐ Σκοπός τους είναι η βελτίωση της ποιότητας του προϊόντος λογισμικού με τις γνώσεις των παρευρισκόμενων
- ☐ Οδηγούν σε ανταλλαγή τεχνικών πληροφοριών και μπορούν να είναι πολύ διδακτικές για τους νέους
- ☐ Μπορούν να είναι ωφέλιμες και για τους παρουσιαστές *και* για τους παρευρισκόμενους
- ☐ Τα Α στη SAMOSA αντιστοιχούν στην ατζέντα (Agenda) και τη δράση (Action), που είναι προαιρετικά τμήματα των καλών συναντήσεων

35

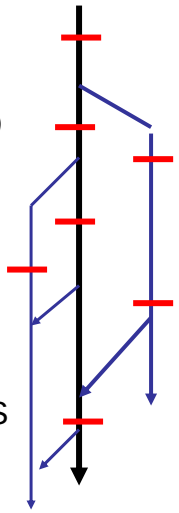
Αποτελεσματική διακλάδωση μέρος 1: Διακλάδωση ανά χαρακτηριστικό

(ESaaS §10.5)

Armando Fox

Διακλαδώσεις

- Κύρια διακλάδωση ανάπτυξης (**master**) και **διακλαδώσεις**
  - Η δημιουργία διακλάδωσης είναι *φθηνή!*
  - εναλλαγή μεταξύ διακλαδώσεων: *εξαγωγή (checkout)*
- Ξεχωριστές ιστορίες επαλήθευσης ανά *διακλάδωση*
- *Συγχώνευση* (merge) διακλάδωσης στην κύρια διακλάδωση
  - ...ή με *ώθηση* (push) αλλαγών διακλάδωσης
  - Οι περισσότερες διακλαδώσεις τελικά διαγράφονται
- Φοβερή περίπτωση χρήσης για λογισμικό SaaS με το Ευέλικτο μοντέλο: *διακλάδωση ανά χαρακτηριστικό*

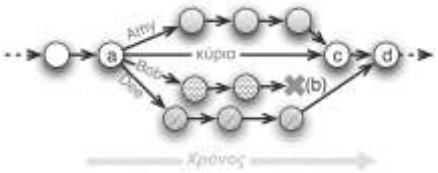


38

Δημιουργία νέων χαρακτηριστικών χωρίς διακοπή του λειτουργικού κώδικα

1. Για να εργαστείτε σε ένα νέο χαρακτηριστικό, δημιουργήστε νέα διακλάδωση *μόνο για το χαρακτηριστικό αυτό*
   - πολλά χαρακτηριστικά μπορούν να βρίσκονται σε εξέλιξη την ίδια στιγμή
2. Χρησιμοποίησε τη διακλάδωση *μόνο* για τις αλλαγές που χρειάζονται για το *χαρακτηριστικό αυτό*, μετά συγχώνευση στην κύρια διακλάδωση
3. Αναίρεση του χαρακτηριστικού ⇔ ακύρωση της συγχώνευσης

Σε καλά δομημένη εφαρμογή, 1 χαρακτηριστικό δεν πρέπει να επηρεάζει πολλά μέρη της εφαρμογής



39

Μηχανισμοί

- Δημιουργία νέας διακλάδωσης & εναλλαγή σε αυτή

```
git branch CoolNewFeature
git checkout CoolNewFeature
```
← *τρέχουσα διακλάδωση*

- Επεξεργασία, προσθήκη, επαληθεύσεις, κ.λπ στη διακλάδωση
- Ώθηση της διακλάδωσης στο αποθετήριο προέλευσης (προαιρετικό):

```
git push origin CoolNewFeature
```

  – δημιουργεί *διακλάδωση παρακολούθησης* στο απομακρυσμένο αποθετήριο

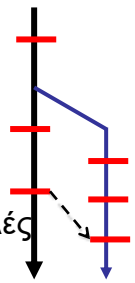
- Εναλλαγή στην κύρια διακλάδωση και συγχώνευση:

```
git checkout master
git merge CoolNewFeature
```
← *προειδοποίηση!!*

40

Αλλαγή βάσης (rebasing)

- Αλλαγή βάσης με βάση το *x* == προσποίηση ότι είναι διακλάδωση του *x*
- Γιατί με αυτόν τον τρόπο;
- Πρέπει να επιλύσει διενέξεις συγχώνευσης όπως με κανονικές συγχωνεύσεις
- Προαιρετικό: μπορείτε να *συνδυάσετε (squash)* πολλές επαληθεύσεις σε μία για να απλοποιήσετε την συγχώνευση αργότερα
- *Κλειδί για να μην εκραγεί το μυαλό σας:* σκεφτείτε με όρους *συνόλων αλλαγών* (changesets), όχι εκδόσεων



41

Αν προσπαθήσετε να ωθήσετε αλλαγές σε ένα απομακρυσμένο αποθετήριο και λάβετε το σφάλμα «non-fast-forward (error): failed to push some refs», ποια πρόταση είναι ΛΑΘΟΣ;

- ☐ Μερικές επαληθεύσεις που εμφανίζονται στο απομακρυσμένο αποθετήριο δεν εμφανίζονται στο τοπικό
- ☐ Πρέπει να πραγματοποιήσετε συγχώνευση/έλξη πριν μπορέσετε να ολοκληρώσετε την ώθηση
- ☐ Πρέπει να διορθώσετε «με το χέρι» διενέξεις συγχωνεύσεων σε ένα ή περισσότερα αρχεία
- ☐ Το τοπικό σας αποθετήριο δεν είναι ενημερωμένο με το απομακρυσμένο

42

42

ΤΕΛΟΣ

43

43

Αποτελεσματική διακλάδωση μέρος 2 : Διακλαδώσεις & εγκατάσταση

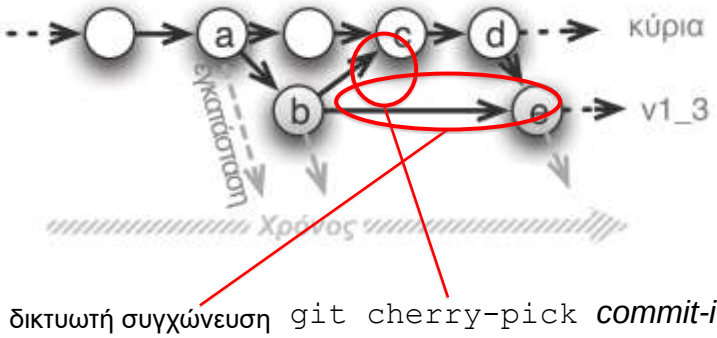
(ESaaS §10.5)

Armando Fox

Διακλαδώσεις & εγκατάσταση

- Οι διακλαδώσεις χαρακτηριστικών θα πρέπει να έχουν μικρή διάρκεια ζωής
  - αλλιώς αποσυγχρονίζονται από την κύρια διακλάδωση, και είναι δύσκολο να συμβαδίσουν
  - Μπορεί να χρησιμοποιηθεί η `git rebase` για «αυξητική» συγχώνευση
  - Μπορεί να χρησιμοποιηθεί η `git cherry-pick` για τη συγχώνευση μόνο συγκεκριμένων επικυρώσεων
- Η «εγκατάσταση από την κύρια διακλάδωση» είναι η πιο συχνή
- Η «διακλάδωση ανά κυκλοφορία» είναι μια εναλλακτική στρατηγική

Διακλαδώσεις κυκλοφορίας/διόρθωσης σφαλμάτων και επιλεκτικές επικυρώσεις



δικτυωτή συγχώνευση `git cherry-pick` *commit-id*

Λογική: η διακλάδωση κυκλοφορίας είναι ένα σταθερό σημείο για αυξητικές διορθώσεις σφαλμάτων

46

46

Branch και Fork

- Το Git υποστηρίζει το μοντέλο συνεργασίας *fork & pull*
- Αν έχετε πρόσβαση με δικαιώματα push/admin στο αποθετήριο:
  - branch: δημιουργεί διακλάδωση σε *αυτό το αποθετήριο*
  - merge: περνά τις αλλαγές διακλάδωσης στην κύρια διακλάδωση (ή σε μια άλλη διακλάδωση)
- Αν δεν έχετε τέτοια πρόσβαση:
  - fork: κλωνοποιεί *ολόκληρο το αποθετήριο* του GitHub σε ένα όπου μπορείτε να εκτελέσετε εντολές διακλάδωσης, ώθησης, κ.λπ.
  - Ολοκληρώστε την εργασία σας στη δική σας διακλάδωση
  - Ευγενική παραχώρηση: πραγματοποιήστε αλλαγή βάσης στη διακλάδωσή σας με commit squash
  - Ανοίξτε μια αίτηση έλξης για να πάρετε την επικύρωση

47

47

Gitfalls ☺

- Ανακάλυψη αλλαγών μετά τη συγχώνευση ή την εναλλαγή διακλαδώσεων

- Εκτέλεση «απλών» αλλαγών απευθείας στην κύρια διακλάδωση

48

Αναίρεση!

```
git reset --hard ORIG_HEAD
git reset --hard HEAD
git checkout commit-id -- αρχεία…
```

- Σύγκριση/διερεύνηση:

```
git diff commit-id-or-branch -- αρχεία…
git diff "master@{01-Sep-12}" --αρχεία
git diff "master@{2 days ago}"--αρχεία
git show mydevbranch:myfile.rb
git blame αρχεία
git log αρχεία
```

49

Αν ανατίθεται σε ξεχωριστές υπο-ομάδες να δουλέψουν σε *διορθώσεις σφαλμάτων κυκλοφορίας* και *νέα χαρακτηριστικά*, θα χρειαστεί να χρησιμοποιήσετε:

- ☐ Διακλάδωση ανά κυκλοφορία
- ☐ Διακλάδωση ανά χαρακτηριστικό
- ☐ Διακλάδωση ανά κυκλοφορία + Διακλάδωση ανά χαρακτηριστικό
- ☐ Οποιοδήποτε από αυτά θα δουλέψει

ΤΕΛΟΣ

Διόρθωση σφαλμάτων: Τα πέντε R

(ESaaS §10.7)

Armando Fox

No Bug Fix Without a Test!

- Αναφορά (**R**eport)
- Αναπαραγωγή ή αλλαγή κατάταξης (**R**eproduce/**R**eclassify)
- Έλεγχος παλινδρόμησης (**R**egression)
- Επιδιόρθωση (**R**epair)
- Κυκλοφορία διόρθωσης (**R**elease the fix) (επικύρωση ή εγκατάσταση)
- Ακόμη και σε μη ευέλικτους οργανισμούς
- Όμως, οι υπάρχουσες ευέλικτες διαδικασίες μπορούν να *προσαρμοστούν* στις διορθώσεις σφαλμάτων

53

Αναφορά

- Pivotal Tracker
  - bug = ιστορία 0 σημείων (αλλά όχι μηδέν προσπάθεια!!)
  - αυτοματοποίηση: τα *άγκιστρα υπηρεσιών* του GitHub μπορούν να διευθετηθούν για να επισημαίνουν μια ιστορία του Tracker ως «παραδομένη» (delivered) όταν ωθείται μια σωστά σχολιασμένη επικύρωση
- Χαρακτηριστικό «ζητημάτων» (issues) του GitHub
- Πλήρης παρακολούθηση σφάλματος, π.χ., Bugzilla
- *Χρησιμοποιήστε το απλούστερο εργαλείο που δουλεύει σωστά για την εμβέλεια της ομάδας & του έργου σας*

54

54

Αλλαγή κατάταξης ή αναπαραγωγή + επιδιόρθωση;

- Αλλαγή κατάταξης ως «μη σφάλμα» ή «δεν θα διορθωθεί»
- Αναπαραγωγή με τον *απλούστερο δυνατό έλεγχο*, και προσθήκη στην παλινδρόμηση
  - ελαχιστοποίησε προσυνθήκες (π.χ., μπλοκ `before` στην RSpec, βήματα `Given` ή `Background` στο Cuke)
- Repair == ο έλεγχος αποτυγχάνει στην παρουσία σφάλματος, επιτυγχάνει στην απουσία σφάλματος
- Κυκλοφορία: μπορεί να σημαίνει είτε ότι έχει γίνει «ώθηση» ή «εγκατάσταση»

55

55

Υποθέστε ότι ανακαλύψατε πως η πιο πρόσφατη κυκλοφορία περιέχει ένα σφάλμα του οποίου ο έλεγχος παλινδρόμησης θα απαιτήσει εκτεταμένη δημιουργία ομοιωμάτων ή στελεχών επειδή ο εσφαλμένος κώδικας είναι μπερδεμένος. Ποια ενέργεια, αν υπάρχει, ΔΕΝ είναι κατάλληλη;

- ☐ Κάνετε την αναδόμηση χρησιμοποιώντας TDD στη διακλάδωση κυκλοφορίας, και ωθείτε τη διόρθωση σφάλματος ως νέο κώδικα με ελέγχους
- ☐ Κάνετε την αναδόμηση χρησιμοποιώντας TDD σε μια διαφορετική διακλάδωση, ωθείτε τη διόρθωση σφάλματος ως νέο κώδικα με ελέγχους, και μετά διαθέτετε επιλεκτικά (cherry-pick) τη διόρθωση σε κυκλοφορία
- ☐ Δημιουργείτε έναν έλεγχο παλινδρόμησης με τα απαραίτητα ομοιώματα και στελέχη, αν και οδυνηρό, και ωθείτε τη διόρθωση και ελέγχους για να κυκλοφορήσετε τη διακλάδωση
- ☐ Ανάλογα με τις προτεραιότητες και τη διαχείριση του έργου, οποιοδήποτε από τα παραπάνω μπορεί να είναι κατάλληλο

56

56

ΤΕΛΟΣ

57

57

Πλάνες & παγίδες, τελικές παρατηρήσεις Κεφαλαίου 10

(Τεχνολογία Ανάπτυξης Λογισμικού ως Υπηρεσίας §10.8-10.9)

David Patterson

Παγίδα

- Παγίδα: Υποδιαίρεση της δουλειάς με βάση τη στοίβα λογισμικού αντί των χαρακτηριστικών
  - Π.χ., ειδικός σε συστήματα διεπαφής/υποβάθρου (front-end/back-end), σύνδεση με πελάτη, …

- Ευέλικτο μοντέλο: καλύτερα αποτελέσματα αν κάθε μέλος της ομάδας παραδίδει *όλες* τις πτυχές μιας ιστορίας
  - Σενάρια Cucumber, έλεγχοι, έλεγχοι RSpec, προβολές, ενέργειες ελεγκτών, λογική μοντέλου, …
  - Ο καθένας στην ομάδα έχει μια άποψη της «πλήρους στοίβας» του προϊόντος

Παγίδα

- Τυχαία διαπίστωση αλλαγών μετά τη συγχώνευση ή την εναλλαγή διακλαδώσεων
  - Στη λάθος διακλάδωση, γράφεις πάνω στις συγχωνευμένες αλλαγές από την παλιά έκδοση στον διορθωτή κειμένου, …

- *Πριν* την έλξη ή τη συγχώνευση, επικύρωσε (commit) όλες τις αλλαγές
- *Μετά* την έλξη ή τη συγχώνευση, επαναφόρτωσε τα αρχεία στον διορθωτή κειμένου
  - Ή κλείστε τον διορθωτή πριν την επικύρωση

60

Παγίδα

- Αν αφήσεις το αντίγραφο του αποθετηρίου σου να αποσυγχρονιστεί πολύ από το αρχικό (επίσημο) αντίγραφο
  - Σημαίνει ότι οι συγχωνεύσεις θα είναι επώδυνες

- Κάνε `git pull` πριν ξεκινήσεις, `git push` μόλις σταθεροποιηθούν αρκετά οι τοπικά επαληθευμένες αλλαγές
- Αν η διακλάδωση είναι μακρόβια, να εκτελείς τακτικά την `git rebase`

61

Πλάνη

- Δεν υπάρχει πρόβλημα να γίνονται απλές αλλαγές στην κύρια διακλάδωση
  - Πιστεύεις ότι είναι αλλαγή 1 γραμμής, γίνεται 5 γραμμές, επηρεάζει άλλο αρχείο, μετά πρέπει να αλλάξεις ελέγχους, …
- Να δημιουργεί πάντα μια διακλάδωση χαρακτηριστικού όταν ξεκινάς καινούρια δουλειά
  - Η δημιουργία διακλάδωσης με το Git γίνεται σχεδόν στιγμή
  - Αν η αλλαγή *είναι* μικρή, μπορείς να διαγράψεις τη διακλάδωση μετά τη συγχώνευση για να αποφύγεις τη ρύπανση του χώρου ονομάτων διακλαδώσεων

62

Τελικές παρατηρήσεις

- Οι ομάδες για 2 πίτσες περιορίζουν το πρόβλημα διαχείρισης από τις μεγάλες ομάδες, αλλά δεν το εξαλείφουν
  - Η συνάθροιση (scrum) είναι ένας ανεπίσημος τρόπος οργάνωσης που ταιριάζει καλά στην Ευέλικτη ανάπτυξη
- Σημεία, ταχύτητα, Tracker => πιο προβλέψιμα
- Σ&Τ: Ο διευθυντής έργου είναι αφεντικό, σχεδιάζει & τεκμηριώνει, κάνεις αναθεωρήσεις για να μάθει από άλλους
- Όταν ολοκληρωθεί το έργο, ξόδεψε χρόνο να σκεφτείς τι έμαθες πριν πας σε άλλο
  - Τι πήγε καλά, τι δεν πήγε, τι να κάνεις διαφορετικά

63

10 εντολές για να είσαι ένας κακός παίχτης στην ομάδα λογισμικού

```
git commit -m 'deal with it' &&
git push --force origin master
```


10 εντολές για να είσαι ένας κακός παίχτης στην ομάδα λογισμικού

(και προτεινόμενες εναλλακτικές)

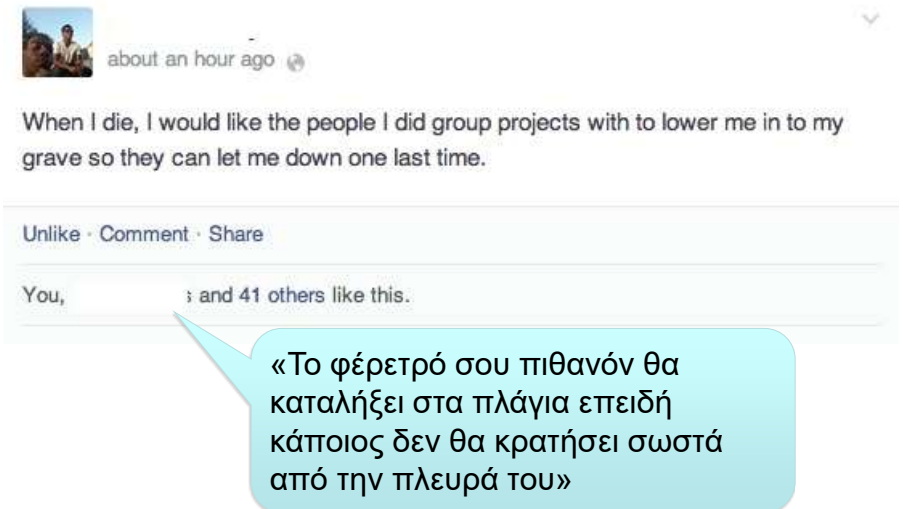
1. Οι αποτυχίες δεν πειράζουν
2. Οι διακλαδώσεις μου, το άδυτο
3. Είναι μια απλή αλλαγή
4. Είμαι ξεχωριστή περίπτωση
5. Οι καρτέλες (tabs) γλιτώνουν πολύτιμα byte
6. Η εξυπνάδα είναι εντυπωσιακή
7. Απλώς άλλαξέ το γρήγορα στον διακομιστή παραγωγής
8. Ο χρόνος που ξοδεύεται σε αναζήτηση = χαμένος χρόνος από τη συγγραφή κώδικα
9. «Πράσινος πυρετός»: πιάσε το!
10. Εβδομάδες συγγραφής κώδικα γλυτώνουν ώρες σχεδιασμού/σκέψης

1. Μην ωθείς ό,τι να ‘ναι
2. Να έχεις διακλαδώσεις με μικρή διάρκεια ζωής
3. Να είσαι πολύ προσεκτικός με κάθε αλλαγή
4. 1 έργο, 1 στυλ κώδικα
5. Μην χρησιμοποιείς καρτέλες
6. Η διαφάνεια δηλώνει ταπεινοφροσύνη
7. Κάνε κάθε αλλαγή αυτοματοποιήσιμη
8. Ξόδεψε 5 λεπτά για αναζήτηση λιγότερου/καλύτερου κώδικα
9. Περισσότεροι έλεγχοι ≠ υψηλότερη ποιότητα
10. Μελέτησε τον σχεδιασμό σου

Μερικές πρόσθετες εντολές για να κάνετε τα έργα διαχειρίσιμα

- κάθε μέθοδος με flog > 10 απορρίπτεται
- κάθε διακλάδωση με διάρκεια ζωής > ~3 ημέρες απορρίπτεται
- κάθε συγχώνευση που χαλάει τη διαδικασία δόμησης (build) ακυρώνεται και ο υπαίτιος **πρέπει** να κάνει αλλαγή βάσης ως προς την κύρια διακλάδωση
- κάθε διόρθωση σφάλματος ή κώδικας που υποβάλλεται χωρίς κάλυψη ελέγχων >90% απορρίπτεται

Μην είσαι αυτό το άτομο



68