Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ

Εθνικό Μετσόβιο Πολυτεχνείο

Τεχνολογία Λογισμικού

7ο / 9ο Εξάμηνο 2018-19

Ν.Παπασπύρου, Αν.Καθ. ΣΗΜΜΥ, nickie@softlab.ntua,gr

Β.Βεσκούκης, Αν.Καθ. ΣΑΤΜ, v.vescoukis@cs.ntua,gr

Κ.Σαΐδης, ΠΔ 407, saiko@softlab.ntua.gr

Περιεχόμενα

- Τεχνική διαχείριση έργων λογισμικού
- Εργαλεία
- Διαχείριση συστατικών λογισμικού (Software artifacts)

Το ζητούμενο

Πώς ο επικεφαλής μηχανικός

- Ο οποίος συμμετέχει σε μια ομάδα ανάπτυξης
- Διαχειρίζεται τεχνικά τη διαδικασία ανάπτυξης
- Διασφαλίζοντας την ποιότητα του τελικού αποτελέσματος

Ανεξάρτητα από

- Τους ρόλους των ανθρώπων στην ομάδα
- Τη μεθοδολογία ανάπτυξης
- Την αρχιτεκτονική του υπό ανάπτυξη λογισμικού

Τεχνική διαχείριση έργου λογισμικού (γενικά)

- Διαχείριση εκδόσεων κώδικα (version control)
- Αυτόματο "χτίσιμο" λογισμικού (build automation)
- Στατική ανάλυση κώδικα και αυτόματος εντοπισμός σφαλμάτων (bug detection)
- Εκτέλεση σεναρίων ελέγχου (tests)
- Συνεχής ολοκλήρωση (continuous integration)
- Διαχείριση συστατικών λογισμικού και των εκδόσεών τους

Συναφή με το λεγόμενο "software configuration management"

Καθημερινότητα ομάδας ανάπτυξης

- Λήψη και ενσωμάτωση των αλλαγών που έγιναν στον κώδικα από τα υπόλοιπα μέλη της ομάδας (Version control tool)
- Επανάληψη:
  - Ανάπτυξη νέου κώδικα / διόρθωση υπάρχοντος (Editor, IDE)
  - Μεταγλώττιση κώδικα, "πακετάρισμα" κι εκτέλεση τοπικών σεναρίων ελέγχου (Build automation tool)
- Διαμοιρασμός των αλλαγών / προσθηκών που κάναμε στον κώδικα στα υπόλοιπα μέλη της ομάδας (Version control tool)

Επίσης

- Συνεχής ολοκλήρωση (CI): ένας server που εκτελεί αυτόματα και μηχανικά όλα τα σενάρια ελέγχου και τις δοκιμές μετά από κάθε αλλαγή στον πηγαίο κώδικα (commit)
- Εγκατάσταση λογισμικού σε ενδιάμεσο περιβάλλον (Staging deployment)
- Εγκατάσταση λογισμικού στο παραγωγικό περιβάλλον (Production deployment)
- Δημοσίευση του λογισμικού σε κάποια αποθήκη (artifact repository)

Πριν ξεκινήσει το (όποιο) έργο

- Σύναψη κοινής "τεχνικής κουλτούρας"
- Κοινές συμβάσεις (conventions)
  - Για τον κώδικα, τη μορφοποίηση, την ομματοδοσία, την τοποθεσία, κλπ.
- Κοινά εργαλεία (tools) & διαδικασίες (procedures)
- Κοινά αποδεκτός και σαφώς ορισμένος ο επιμερισμός ευθύνης (ownership)

Επιμερισμός ευθύνης (ownership)

- Ποιος είναι αρμόδιος για ποιο κομμάτι κώδικα / λειτουργικότητας / χαρακτηριστικών ή για ποιο βήμα;
- Αποφυγή "ορφανού" κώδικα / βήματος (κανείς δεν ξέρει πως δουλεύει το component Χ ή πώς να εκτελέσει το βήμα Χ)
- Αποφυγή "αποκλεισμένου" κώδικα / βήματος (μόνο ο Χ ξέρει το Ψ)
- Τουλάχιστον δύο πρόσωπα με την ίδια αρμοδιότητα σε κάποιο κομμάτι

Bus factor

Κι αν τους χτυπήσει λεωφορείο;

> The "bus factor" is the minimum number of team members that have to suddenly disappear from a project before the project stalls due to lack of knowledgeable or competent personnel.

Wikipedia

Συμβάσεις (conventions)

Παραδείγματα

Συμβάσεις για τη δομή του κώδικα

- Τρόπος συγγραφής κώδικα
  - Ονόματα κλάσεων, μεθόδων, μεταβλητών
  - Στοίχιση κώδικα
- Παράδειγμα: Java coding conventions
  - `NameOfClass`
  - `nameOfMethod`, `nameOfVariable` και `nameOfField`
  - `NAME_OF_CONSTANT`

Συμβάσεις για τη δομή των αρχείων

- Διάρθρωση και τοποθέτηση των αρχείων
  - Source files, resources, configuration files, libraries, κτλ.
- Στο JVM τα Maven directory structure conventions αποτελούν το de facto standard
  - `src/main/java`
  - `src/main/resources`
  - `src/test/java`
  - `src/test/resources`
  - `build/classes`

Java Packages

```
package x.y.z;
class Foo {
  ...
}
```

Πηγαίος κώδικας

`src/main/java/x/y/z/Foo.java`

Class files

`build/classes/x/y/z/Foo.class`

Test Packages

```
package x.y.z;
class FooTest {
  //the tests of class Foo
  ...
}
```

Πηγαίος κώδικας

`src/test/java/x/y/z/Foo.java`

Class files

`build/classes/x/y/z/Foo.class`

Εργαλεία

- Version control
- Integrated Development Environments (IDEs)
- Building
- Testing

- Bug detection
- Continuous integration

- Continuous deployment and delivery (DevOps)
- Containers - monitoring - alerting

Version control

Ζητούμενα

- Διαχείριση του source code base (της "βάσης" του κώδικα)
- Καταγραφή των αλλαγών (ιστορικό, ποιος έκανε τι και πότε)
- Συνεργατική ανάπτυξη
- Τήρηση πολλών παράλληλων καταστάσεων του κώδικα ταυτόχρονα
- Ανάκτηση συγκεκριμένης παρελθούσας κατάστασης
- κ.ά.

Στο μάθημα

Εργαλείο git (φροντιστήριο σε λίγο)

Software build automation

Ζητούμενα

- Αυτοματοποίηση της διαδικασίας "χτισίματος" του λογισμικού
- Πώς από το source code base παράγεται το software artifact (.jar, .exe, .rpm, .deb, κλπ)

Ο βασικός "τεχνικός" στόχος κάθε έργου λογισμικού είναι η παραγωγή ενός ή περισσότερων software artifacts

Software build automation

- Αυτόματη διαχείριση εξαρτήσεων (dependencies)
- Μεταγλώττιση κώδικα (compilation)
- Εκτέλεση σεναρίων ελέγχου (testing)
- Συνεχής ολοκλήρωση του λογισμικού (Continuous integration)
- Παραγωγή των software artifacts (assemble)
- Απόθεση/δημοσίευσή τους σε κάποια αποθήκη (software release / publication)

Εργαλεία build automation

make

- Η αρχή όλων
- Stuard Feldman, Bell Labs, 1976
- 2003 ACM Software System Award
- Βασικές έννοιες
  - Targets, prerequisites, commands, macros
  - Topological sorting

Topological sorting

Γραμμική διάταξη των κόμβων ενός κατευθυνόμενου γράφου, στην οποία το u προηγείται του v αν υπάρχει κατευθυνόμενη ακμή $u -> v$.

Ο γράφος δεν πρέπει να έχει κύκλους (ακυκλικός) - Directed Acyclic Graph (DAG).

Εκτέλεση των build targets.

Εργαλεία στο Java οικοσύστημα

- Apache Ant
- Apache Ivy
- Apache Maven
- Gradle (έμφαση στο μάθημα)

Gradle

https://www.slideshare.net/KostasSaidis/an-introduction-to-gradle-for-java-developers

Θα σας δοθεί έτοιμο για χρήση Gradle project ως παράδειγμα

Διαχείριση συστατικών λογισμικού

Software artifacts

- Αυτοτελή αρχεία έτοιμα προς εκτέλεση ή
- Μερικώς αυτοτελή αρχεία προς ενσωμάτωση σε άλλες εφαρμογές (βιβλιοθήκες)
- Δομή/περιεχόμενα ανάλογα με τη γλώσσα προγραμματισμού, το λειτουργικό σύστημα, την εφαρμογή, κτλ.

Στη Java κοινότητα

- Software artifact = Jar (συνήθως) ή APK (android)
- Jar αρχείο = Zip αρχείο
- Περιέχει .class αρχεία (JVM κλάσεις) και -ενδεχομένως- metadata (manifests), resources (images), αρχεία ρυθμίσεων κ.ο.κ.
- APK αρχείο = Zip αρχείο
- Περιέχει .dex αρχεία και metadata (manifests), resources (images) κ.λπ.

Στην πράξη

Κάθε ξεχωριστή προγραμματιστική/τεχνολογική κοινότητα συνήθως:

- έχει ξεχωριστούς μορφότυπους artifacts
- έχει ξεχωριστά εργαλεία διαχείρισής τους
- έχει -όπως λέγεται- ξεχωριστό technology stack

Ας το δούμε αντίστροφα

Τι είναι ένα software artifact;

- Μπορεί να είναι οτιδήποτε:
  - Application, Library, Component, Server, Client
- Μπορεί να χρησιμοποιείται για ένα μόνο σκοπό (π.χ. standalone app) ή να είναι επαναχρησιμοποιήσιμο για πολλούς σκοπούς (π.χ. library)
- Ας πούμε ότι, γενικά, είναι ένα συστατικό λογισμικού

Συστατικά λογισμικού

- Επαναχρησιμοποιήσιμα τμήματα λογισμικού (που διατίθενται ως ξεχωριστά software artifacts)
- Αποθήκες συστατικών λογισμικού (software artifact repositories)
- Διαχείριση εκδόσεων συστατικών (software releases, artifact versioning)

Εξαρτήσεις λογισμικού (software dependencies)

- Compile-time dependencies ("static" linking): οι εξαρτήσεις του λογισμικού από συστατικά που πρέπει να είναι διαθέσιμα κατά τη μεταγλώττιση
- Runtime dependencies ("dynamic" linking): οι εξαρτήσεις του λογισμικού από συστατικά που πρέπει να είναι διαθέσιμα κατά το χρόνο εκτέλεσης

Μεταβατικές εξαρτήσεις (transitive dependencies)

Οι εξαρτήσεις των εξαρτήσεων (τα συστατικά δεν είναι πάντα αυτοτελή, μπορεί να εξαρτώνται από άλλα συστατικά / artifacts για τη λειτουργία τους)

- `Project -> Lib1, Lib2`
- `Lib1 -> Lib11`
- `Lib2 -> Lib21, Lib22`
- `Lib21 -> Lib211`

Αποθήκες συστατικών λογισμικού (software artifact repositories)

- Τήρηση των software artifacts (files)
- Σε πολλές εκδόσεις
- Τήρηση μεταδεδομένων για τις αλληλο-εξαρτήσεις τους
- Δημόσιες ή ιδιωτικές αποθήκες

Παραδείγματα δημόσιων αποθηκών (Java)

https://search.maven.org/

https://bintray.com/bintray/jcenter

https://plugins.gradle.org/

Ιδιωτικές αποθήκες

- Για μεγάλες ομάδες
- Για σύνθετο λογισμικό
- Φιλοξενία της αποθήκης στο εσωτερικό δίκτυο του οργανισμού

Δημοσίευση συστατικού

- Ανέβασμα του συστατικού σε κάποια αποθήκη (artifact publication)
- Αυτοματοποιημένη διαδικασία μέσω του build εργαλείου
- Παράδειγμα `gradle publish` η `mvn release`

Για παράδειγμα

- Εκτέλεση όλων των προαπαιτούμενων για την παραγωγή του jar αρχείου (download dependencies, compile code, run tests, assemble files)
- Παραγωγή του jar αρχείου με κάποια σύμβαση για το όνομα και την έκδοσή του (π.χ. project-name-1.2.jar)
- Ανέβασμα του αρχείου στην αποθήκη
  - σε κάποια συγκεκριμένη -κατά σύμβαση- θέση (π.χ. `/gr/ntua/softeng17b/foo/1.2/project-name-1.2.jar`)
  - μαζί με πληροφορίες/μεταδεδομένα για τις εξαρτήσεις του (π.χ. maven pom, ivy file)

Διαχείριση εκδόσεων λογισμικού

- Versioning
- Releasing

Τυποποίηση εκδόσεων λογισμικού

- Δεν υπάρχει ομοιομορφία και συνέπεια στην ανάθεση εκδόσεων
- Διαφορετικά σχήματα
- Με βάση την ημερομηνία, με βάση κάποια σύμβαση, με βάση εμπορικούς λόγους, κλπ.

Παράδειγμα

[major].[minor].[revision].[build]

- major: Σημαντική αλλαγή στο λογισμικό (major release)
- minor: Προσθήκη ή βελτίωση στο λογισμικό (minor release)
- revision: Patch, διόρθωση bug, επίλυση προβλήματος ασφαλείας, κτλ. (maintenance release)
- build: Αυτόματη αρίθμηση του build (π.χ. αύξων αριθμός, commit id, κτλ.) (internal release)

Semantic Versioning

[major].[minor].[patch]

- major: Breaking change
- minor: Add new backwards compatible functionality
- patch: Apply backwards compatible bug fixes
- Βασική έννοια: Public API
- http://semver.org/

Επίτευξη του release

Επιμέρους στάδια και ενδιάμεσες εκδόσεις

- Pre-release (internal release)
- Early Access (EA)
  - Alpha
  - Beta
  - Release candidate
- Release (General Availability, GA)

Θα επανέλθουμε στη διάλεξη για testing.

Σε κάθε βήμα του release (Pre, EA, GA)

- Publish the artifact
- Tag the commit with the published version number
- Update the version file
- Commit the change
- Push it

Αυτοματοποίηση μέσω του build εργαλείου