07 Διαχείριση έργων λογισμικού (II)

Τεχνολογία Λογισμικού

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Χειμερινό εξάμηνο 2017-18

Δρ. Κώστας Σαΐδης (saiko@di.uoa.gr)

Συνεχίζουμε την εισαγωγική κουβέντα της διάλεξης της 12/10

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

Περιεχόμενα

- Αυτοματισμός "χτισίματος" κώδικα
- Διαχείριση συστατικών και των εκδόσεών τους

Συναφή με το λεγόμενο "software configuration management"

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

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)

Παράδειγμα

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`

Παράδειγμα

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

Καθημερινές διαδικασίες

- Pull, commit, branch, tag, merge, push
- Code review (merge/pull requests)
- Build
- Test
- Deploy to CI
- Pre-release
- Deploy to staging
- Release
- Deploy to production

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 X ή πώς να κάνει το βήμα X)
- Αποφυγή "αποκλεισμένου" κώδικα / βήματος (μόνο ο X ξέρει το Ψ)
- Τουλάχιστον δύο πρόσωπα με την ίδια αρμοδιότητα σε κάποιο κομμάτι

Version control

Ζητούμενα

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

Στο μάθημα

Εργαλείο git

Build automation

Ζητούμενα

- Αυτοματοποίηση της διαδικασίας "χτισίματος" του λογισμικού
- Πώς από το source 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 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 αρχείο (συνήθως)
- Jar αρχείο = Zip αρχείο
- Περιέχει .class αρχεία (JVM κλάσεις) και -ενδεχομένως- metadata (manifests), resources (images), αρχεία ρυθμίσεων κ.ο.κ.

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

Διαχείριση συστατικών 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 dependencis ("static" linking)
- Runtime dependencies ("dynamic" linking)

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 της Java κοινότητας

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
- Κύρια 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

Πίσω στο build automation

Βασικά ζητούμενα στην πράξη

- Αυτόματη διαχείριση εξαρτήσεων
- Μεταγλώττιση κώδικα
- Εκτέλεση σεναρίων ελέγχου
- Παραγωγή των software artifacts
- Συνεχής ολοκλήρωσή τους (CI)
- Απόθεση/δημοσίευσή τους σε κάποια αποθήκη (software release)

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)

Ant

- Πρώτη έκδοση το 2000
- XML
- Πιο portable από το make (και κυρίως για χρήση σε έργα Java)
- "Χειροκίνητη" ρύθμιση όλων των παραμέτρων και της διαχείρισης των εξαρτήσεων

Παράδειγμα

build.xml

```
<project name="MyProject" default="dist" basedir=".">
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist" location="dist"/>

  <target name="init">
    <!--
    Create the build directory structure used by compile
    -->
    <mkdir dir="${build}"/>
  </target>

  <target name="compile" depends="init">
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>
```

Παράδειγμα (συνέχεια)

```
  <target name="dist" depends="compile">
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!--
    Put everything in ${build} into the MyProject.jar file
    -->
    <jar jarfile="${dist}/lib/MyProject.jar"
         basedir="${build}"/>
  </target>

  <target name="clean">
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

Χρήση

```
> ls project
src build.xml

> ant [target]
```

Το target έχει οριστεί στο build.xml (π.χ. compile, dist)

Maven

- Πρώτη έκδοση το 2004
- XML
- Αυτόματη διαχείριση εξαρτήσεων
- Δημιουργία του Maven Central (maven artifact repository)

Σημαντική συμβολή στο Java οικοσύστημα

- Convention over configuration
- Project Object Model (POM)
- Plugins (provide goals)
- Build lifecycles (provide phases)
- Dependencies
- Maven artifacts (jar + dependency metadata)
- Maven co-ordinates (groupId:artifactId:version, π.χ. org.apache.ivy:ivy:2.4.0)

Παράδειγμα

pom.xml

```
<project>

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Χρήση

```
> ls project
src pom.xml

> mvn [plugin]:[command]

> mvn [phase]
```

Default lifecycle phases

validate, generate-sources, process-sources, generate-resources, process-resources, compile, process-test-sources, process-test-resources, test-compile, test, package, install, deploy

Ivy

- Πρώτη έκδοση 2007
- XML
- Transitive dependency manager
- Συμπληρωματικό του ant

Gradle

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