

Medii de proiectare și programare

2022-2023

Conținut

- Build automation tool (Gradle)
- Conectarea la baze de date relaționale
- Inversion of Control - Spring
- Aplicații client-server - Șablonul de proiectare Proxy
- Introducere în apelul metodelor la distanță (Remote Procedure Call) - **Remoting/WCF, RMI, Spring Remoting**
- Enterprise Application Integration - Protocol buffers, Thrift, ActiveMQ, RabbitMQ
- Object Relational Mapping (Strategii, Hibernate, Entity Framework)
- REST Services
- Dezvoltarea aplicațiilor web folosind frameworkuri (React)
- Web sockets
- Securitate Web - Acces bazat pe roluri (JWT Token)

Bibliografie

- Joseph Albahari, Ben Albahari, *C# 6.0 in a Nutshell*, Sixth Edition, O'Reilley, 2015.
- Craig Larman, *Applying UML and Design Patterns: An Introduction to OO Analysis and Design and Unified Process*, Berlin, Prentice Hall, 2002.
- Martin Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.
- Hohpe, G., Woolf, B., *Enterprise integration patterns*, Addison-Wesley, 2003.
- ***, Microsoft Developer Network, Microsoft Inc., <http://msdn.microsoft.com/>
- ***, The Java Tutorials, Oracle Java Documentation, Inc. <https://docs.oracle.com/javase/tutorial/>
- Craig Walls, *Spring in Action*, 4th Edition, Ed. O'Reilley, 2015.
- Documentație Spring <http://spring.io/projects>
- Alte tutoriale

Evaluare

- Întrebări în timpul cursului (**QC**) - **10**%
- Examen (**NE**) - **50**%
- Teme în timpul laboratorului (**TL**) - **10**%
- Laboratoare (**LB**) - **30** %
- Nota finala **$NF=0.1*QC+0.5*NE+0.1*TL+0.3*LB$**
- Condiții pentru promovare:
 - **$NE \geq 5$, $LB \geq 4.5$**
 - **$NF \geq 4.5$**

Laborator

- Asignarea și proiectarea aplicației. Studenții trebuie să proiecteze și să dezvolte o aplicație **client-server**.
- Configurarea aplicației folosind *Gradle*, IoC.
- Proiectarea și implementarea persistenței (baze de date relaționale, ORM)
- Proiectarea și implementarea serviciilor (Șablonul *Proxy*).
- Enterprise Application Integration (*Protobuff/ Thrift/ gRPC*)
- Servicii REST
- Aplicații web (web sockets / securitate).

Notare laborator

- Fiecare temă de laborator are un termen de predare.
- Predare completă a temei la termen: **nota 10.**
- Pentru fiecare săptămână întârziere în care nu s-a predat nimic din temă: **penalizare 3 puncte/săpt.**
- Pentru fiecare săptămână întârziere în care s-a predat ceva (este incompletă, erori): **penalizare 1 punct/săpt,** până la predarea completă a temei.
- Tema nepredată: **nota 0.**
- Tema copiată: **nota 0.**



Titlu proiect: *Azi Student, Mâine Antreprenor!*

Acronim proiect: ASMA

Cod proiect: POCU/379/6/21/123894

Beneficiar: Universitatea Babeș-Bolyai din Cluj-Napoca

Proiect cofinanțat din FSE prin Programul Operațional Capital Uman 2014-2020

Curs 1

2022-2023

Curs 1

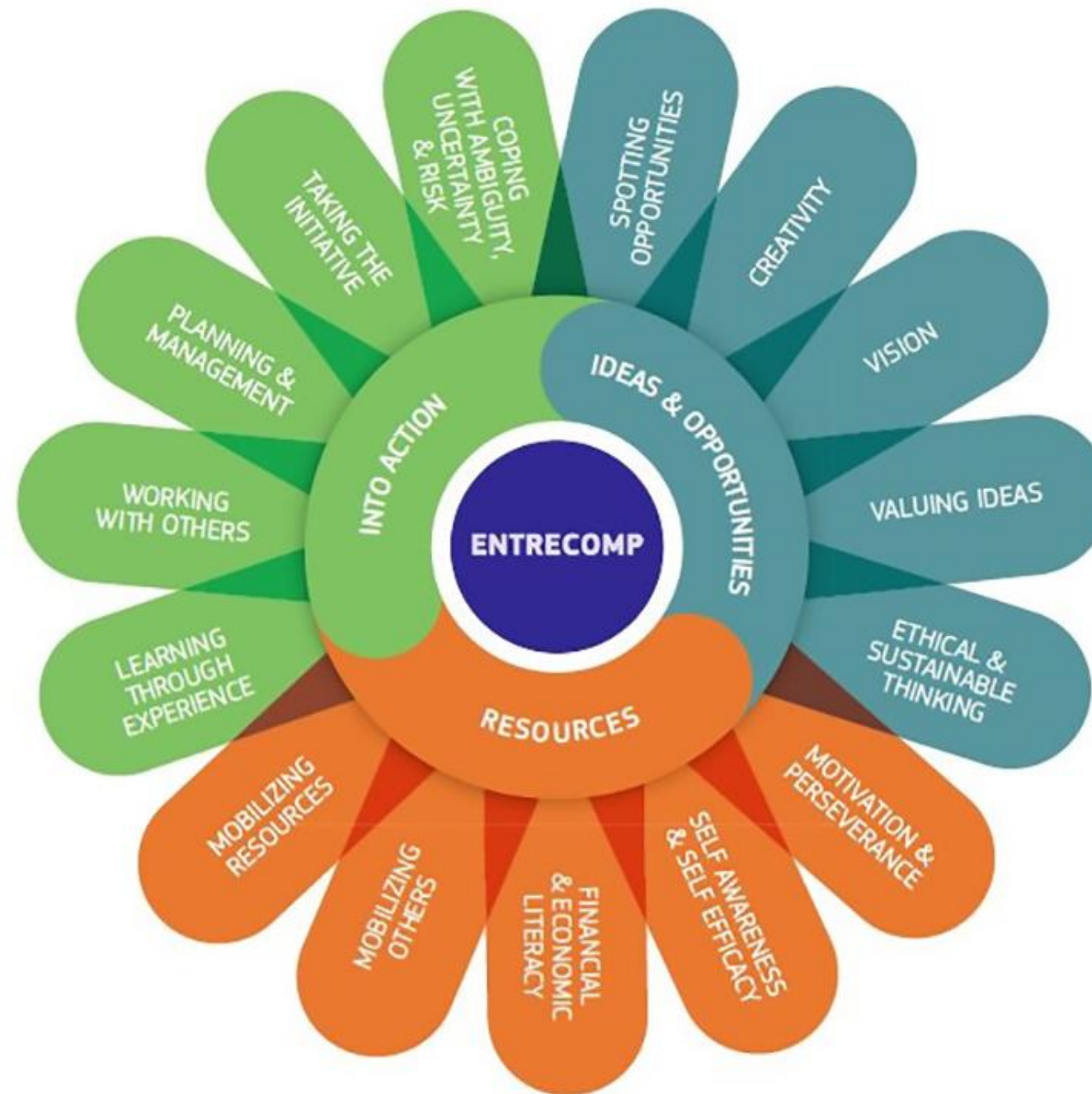
- Competențe antreprenoriale (informatică)
- Instrumente pentru construire automată
 - Gradle

Competențe antreprenoriale



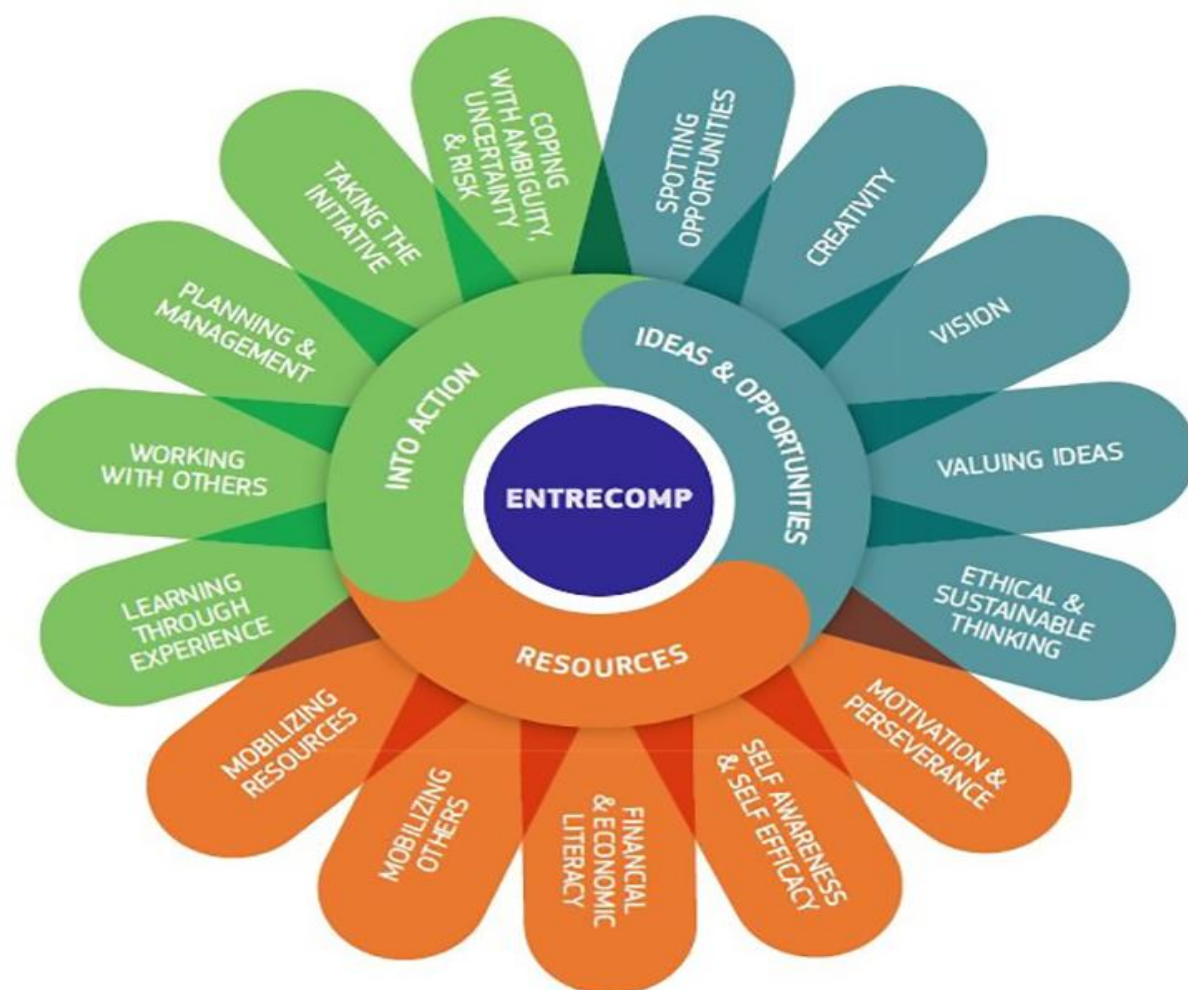
Key competences for lifelong learning, European Commission, Directorate-General for Education, Youth, Sport and Culture, 2019

Competențe antreprenoriale



EntreComp -The European Entrepreneurship Competence Framework, 2018

Informatică - competențe antreprenoriale



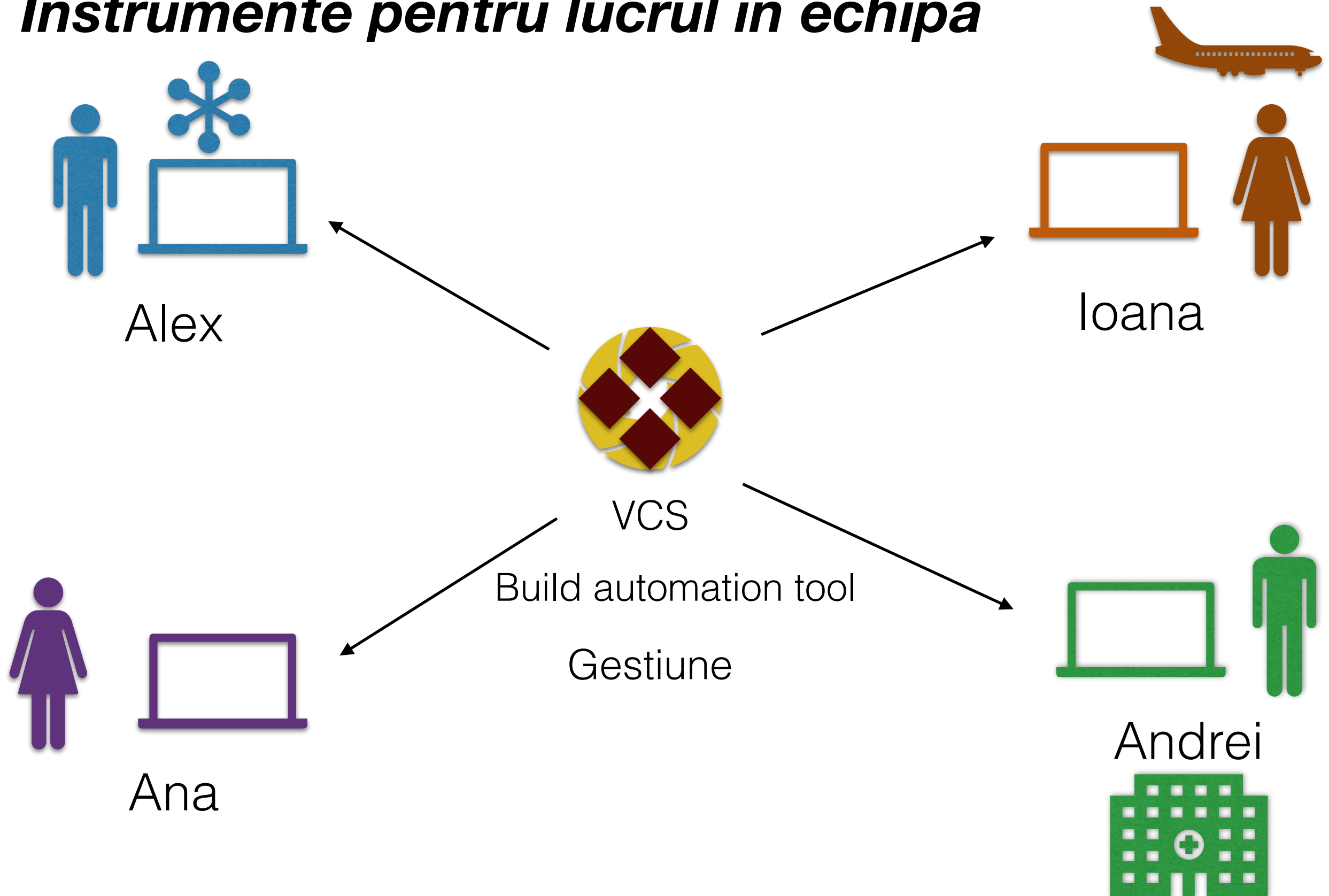
- Lucrul în echipă
- Învățarea prin experiență
- Creativitate
- Identificarea de oportunități

Proiect semestrial facultativ!

Instrumente pentru lucrul în echipă

- Gestiuinea proiectelor soft:
 - Jira, Trello, Atlassian, Zoho, etc.
- Controlul versiunilor și colaborare:
 - GitHub, Git, GitLab, Atlassian Git, BitBucket, etc.
- Construire automată:
 - Gradle, Maven, Jenkins, Travis CI, etc.
- Jurnalizare:
 - Log4j2, SLF4J, Logger.NET, NLog, etc.

Instrumente pentru lucrul în echipă



Instrumente pentru construire automată*

- **Procesul automatizat** corespunzător **construirii** unui sistem soft și a proceselor asociate (compilarea codului sursă, rularea testelor automate, împachetarea codului binar, etc).
 - Makefile
- Categori:
- a. Utilitare** pentru construire automată:
 - Generează artefactele corespunzătoare construirii în timpul compilării și/sau link-editării codului.
 - Make, MS build, Ant, **Gradle**, Maven etc.
- b. Servere** pentru construire automată:
 - Sisteme soft care execută/rulează utilitarele de construire automată la perioade de timp predefinite sau în momentul apariției anumitor evenimente.
 - TeamCity, Jenkins, CruiseControl, etc.

*Eng. *Build automation tools*

Utilitare pentru construire automată

Avantaje

- Permit **automatizarea** *sarcinilor simple și repetabile*.
- Permit **atingerea** unui scop/*obiectiv* prin execuția fiecărei sarcini dintr-un set de sarcini specifice în ordinea corectă.

Îmbunătățesc
calitatea produselor

Accelerează
compilarea și link-
editarea

Elimină sarcinile
redundante

Elimină
dependențele de
angajați 'cheie'

Oferă un **jurnal** al
construcțiilor ce
pot fi folosite când
apar **probleme**

Reduc numărul
construcțiilor
eșuate

Economisesc
timp și bani



Gradle Build Tool

- Gradle Build Tool, <https://gradle.org/>
- Guides <https://gradle.org/guides/#getting-started>
- Tutoriale:
 - <http://www.vogella.com/tutorials/Gradle/article.html#introduction-to-the-gradle-build-system>
 - <https://www.petrikainulainen.net/getting-started-with-gradle/>
- Altele ...

**Permite gestiunea
avansată a
construirii
aplicațiilor**

**Oferă suport pentru
configurarea și
descărcarea
automată a
bibliotecilor și a altor
dependențe**

**Permite
refolosirea
artefactelor
construite folosind
alte utilitare**



Gradle Build Tool

**Oferă suport pentru
depozitarele* Maven și
Ivy pentru refacerea
dependențelor**

**Oferă suport pentru
construiri multi-
proiect și multi-
artefacte**

*eng. *repositories*



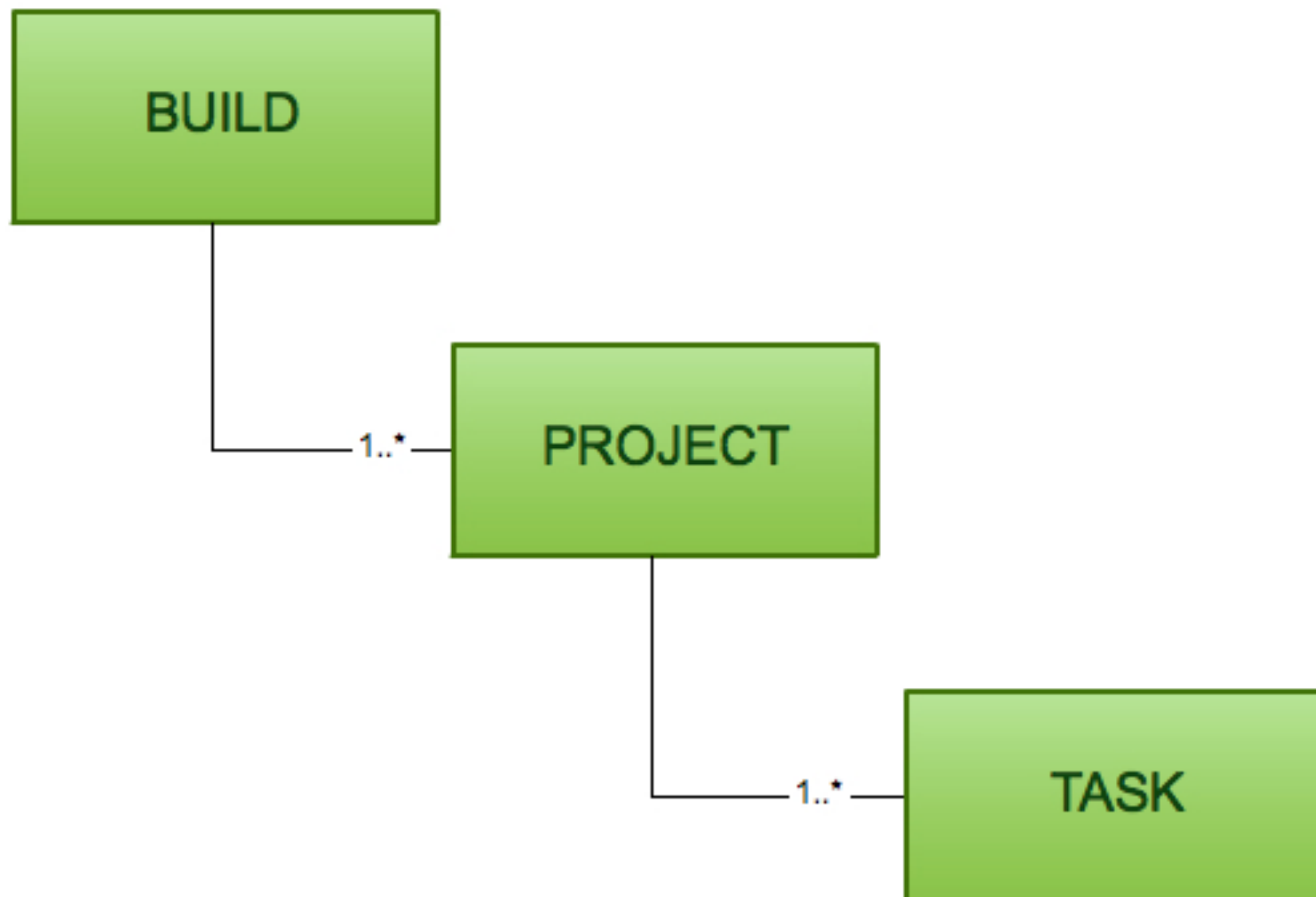
Gradle Build Tool

- Concepte de bază:
 - proiect (eng. *project*)
 - sarcini (eng. *tasks*).
- ❖ Un proiect este:
 - ceva ce se dorește *a se construi* (ex. un sistem soft, un fișier jar),
 - ceva ce se dorește *a se face* (instalarea unui aplicații pentru folosirea de către utilizatori).
- ❖ Un proiect are asociate una sau mai multe *sarcini*.
- ❖ O sarcină este o *unitate de lucru* care este *efectuată pentru construirea automată*:
 - compilarea unui proiect
 - rularea testelor automate
 - ...



Gradle Build Tool

- Fiecare construire Gradle conține unul sau mai multe proiecte.





Gradle Build Tool

- O construire Gradle se configurează folosind fişierele:
 - `build.gradle` (*Gradle build script*) - specifică un proiect şi sarcinile sale asociate.
 - `gradle.properties` (*Gradle properties file*) - este folosit pentru configurarea proprietăţilor construirii.
 - `settings.gradle` (*Gradle Settings file*) - opţional la construirile ce conţin un singur proiect.
- Dacă construirea este formată din mai multe proiecte Gradle, fişierul este obligatoriu şi descrie proiectele conţinute.
- Fiecare construire multi-proiect trebuie să aibă un fişier `settings.gradle` în directorul rădăcină al proiectului.



Gradle Build Tool

- **Exemplu fișier build.gradle**

```
apply plugin: 'java'
```

```
repositories {
```

```
//pentru rezolvarea dependențelor: Maven, JCenter, Ivy
```

```
    mavenCentral()
```

```
    //jcenter()
```

```
}
```

```
dependencies {
```

```
    //format GAV Grup:Artefact:Versiune
```

```
    compile 'com.google.guava:guava:20.0' //versiuni vechi
```

```
    testCompile group: 'junit', name: 'junit', version: '4.+' //versiuni vechi
```

```
    //alte dependențe
```

```
//versiuni noi Gradle
```

```
    implementation 'com.google.guava:guava:20.0'
```

```
    testImplementation group: 'junit', name: 'junit', version: '4.+'
```

```
}
```



Gradle Build Tool

- settings.gradle

```
rootProject.name = 'NumeProiect'
```



Gradle Build Tool

Pluginuri

Toate caracteristicile/facilitățile utile unui proiect sunt furnizate de pluginuri.

Pot să ofere o
configurație implicită
la sarcinile adăugate

Pot să adauge noi
dependențe
proiectului

Pot să adauge
sarcini noi la proiect

Pot să adauge noi
proprietăți care sunt
folosite pentru a redefini
configurația implicită a
proiectului



Gradle Build Tool

- Un plugin poate fi adăugat unui proiect folosind numele sau tipul acestuia:
- Exemplu folosind numele:

În fișierul `build.gradle`:

```
apply plugin: 'foo'
```

- Exemplu folosind tipul:

În fișierul `build.gradle`:

```
apply plugin: 'com.bar.foo'
```




Gradle Build Tool

- Blocul plugins (versiuni mai **noi** Gradle):

```
plugins {
```

```
    id 'java'
```

```
    id 'application'
```

```
}
```

- Se poate adăuga și versiunea unui plugin

```
id 'org.openjfx.javafxplugin' version '0.0.7'
```



Gradle Build Tool

- Pluginuri Gradle standard:
 - **java**: adaugă sarcini pentru compilarea, testarea și împachetarea unui proiect Java. Este un plugin de bază pentru alte pluginuri.
 - **groovy**: adaugă suport pentru proiecte Groovy.
 - **cpp**: adaugă sarcini pentru compilarea codului unui proiect C++.
 - **application**: adaugă sarcini pentru rularea și împachetarea unui proiect Java ca și aplicație executabilă. `application{mainClass='StartServer'}`
 - **distribution**: adaugă suport pentru construirea distribuțiilor de tip ZIP și TAR.
 - **signing**: adaugă abilitatea de a semna digital fișierele și artefactelor construite.
 - etc.



Gradle Build Tool

- Structura unui proiect Java:
 - *src/main/java* - directorul corespunzător codului sursă.
 - *src/main/resources* - directorul corespunzător resurselor folosite în proiect (fișiere de proprietăți, imagini, fxml etc.).
 - *src/test/java* - directorul corespunzător testelor automate.
 - *src/test/resources* - directorul corespunzător resurselor necesare testelor automate.
 - **build** - directorul ce conține toate artefactele construite folosind Gradle.
 - *classes* - conține fișierele *.class*.
 - *libs* - conține fișierele *jar*, *war*, *ear* create folosind Gradle.
 - etc.



Gradle Build Tool

- Crearea unui proiect Java

În fișierul `build.gradle`:

```
apply plugin: 'java'
```

```
plugins{  
    id 'java'  
}
```

- Crearea unui proiect Java cu structura implicită*:

```
gradle init --type 'java-library'
```

```
gradle init --type 'java-application'
```

**în directorul rădăcină al proiectului*



Gradle Build Tool

- Sarcinile asociate unui proiect Java (plugin java):
 - *assemble* - compilează codul sursă al aplicației și creează fișierul jar. Nu rulează testele automate.
 - *build* - construiește toate artefactele asociate proiectului.
 - *clean* - șterge directorul build asociat proiectului.
 - *compileJava* - compilează codul sursă al aplicației.
 - etc.



Gradle Build Tool

- **gradle tasks** - afișează lista completă a sarcinilor ce pot fi executate pentru un proiect și descrierea acestora:
 - *assemble* - Assembles the outputs of this project.
 - *build* - Assembles and tests this project.
 - *buildDependents* - Assembles and tests this project and all projects that depend on it.
 - *buildNeeded* - Assembles and tests this project and all projects it depends on.
 - *classes* - Assembles main classes.
 - *clean* - Deletes the build directory.
 - *jar* - Assembles a jar archive containing the main classes.
 - *testClasses* - Assembles test classes.
 - *check* - Runs all checks.
 - *test* - Runs the unit tests.
 - etc.



Gradle Build Tool

- Împachetarea aplicației (obținerea artefactelor):

- *gradle assemble*

- :compileJava*

- :processResources*

- :classes*

- :jar*

- :assemble*



Gradle Build Tool

- Împachetarea aplicației (obținerea artefactelor și rularea testelor automate):

- *gradle build*

:compileJava

:processResources

:classes

:jar

:assemble

:compileTestJava

:processTestResources

:testClasses

:test

:check

:build

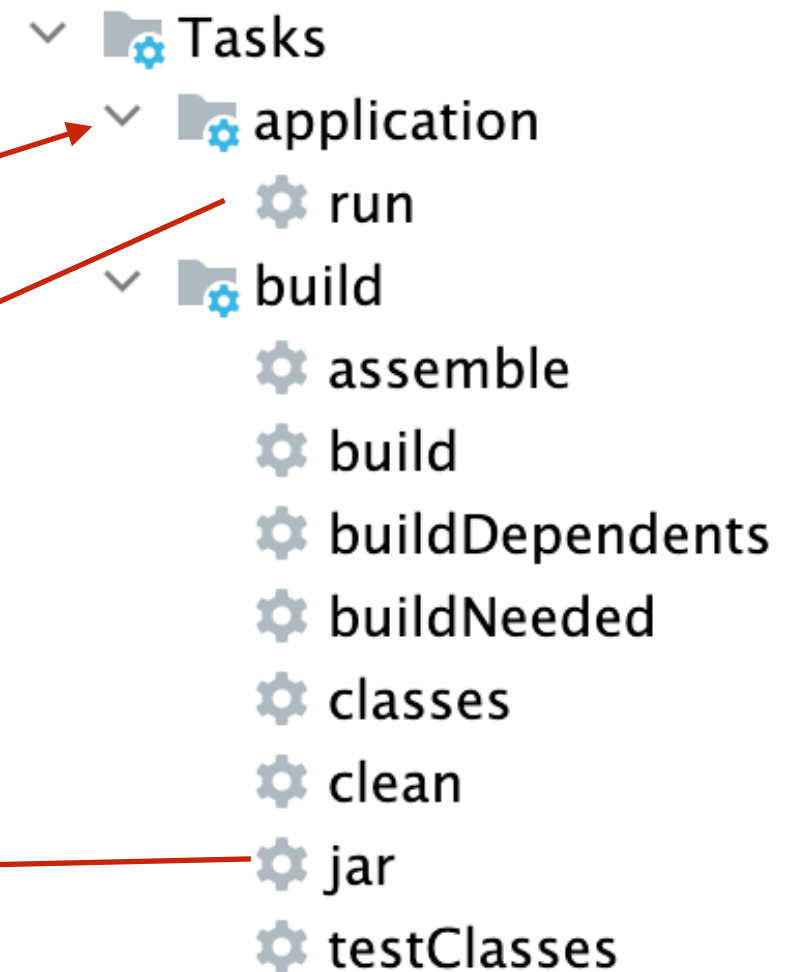


Gradle Build Tool

- Fișiere *jar* executabile:

- build.gradle*

```
plugins{  
    id 'java'  
    id 'application'  
}  
application{  
    mainClass='NumeClasaCuMain'  
}  
jar {  
    manifest {  
        attributes('Main-Class':'NumeClasaCuMain')  
    }  
  
    from {  
        configurations.runtimeClasspath.collect { it.isDirectory() ? it : zipTree(it) }  
    }  
}
```





Gradle Build Tool

- Proiecte multiple:
- Fiecare proiect (subproiect) are aceeași structură - corespunzătoare proiectelor Gradle Java.
- Fiecare proiect (subproiect) va conține fișierul `build.gradle` propriu, cu configurările specifice.
- Proiectul rădăcină (root) conține obligatoriu și fișierul `settings.gradle`:

```
include 'A'
```

```
include 'B'
```



Gradle Build Tool

- Dependente între (sub)proiecte: Subproiectul B depinde de subproiectul A:
- `build.gradle` corespunzător subproiectului B:

```
dependencies {  
    implementation project(':A')  
}
```



Gradle Build Tool

- Proiectul A: `build.gradle`

```
plugins{  
    id 'java'  
}  
repositories {  
    mavenCentral()  
}  
dependencies {  
    implementation 'com.google.guava:guava:20.0'  
    testImplementation 'junit:junit:4.11'  
}
```



Gradle Build Tool

- Proiectul B: `build.gradle`

```
plugins{  
    id 'application'  
    id 'java'  
}  
repositories {  
    mavenCentral()  
}  
application{  
    mainClass='StartApp'  
}  
dependencies {  
    testImplementation 'junit:junit:4.11'  
    implementation project(':A')  
}
```



Gradle Build Tool

- Project Root: `build.gradle`

```
allprojects {  
    plugins{  
        id 'java'  
    }  
    repositories {  
        mavenCentral()  
    }  
    dependencies {  
        testImplementation 'junit:junit:4.11'  
    }  
}
```



Gradle Build Tool

- Proiectul A: `build.gradle` modificat

```
dependencies {  
    implementation 'com.google.guava:guava:20.0'  
}
```

- Proiectul B: `build.gradle` modificat

```
plugins{  
    id 'application'  
}  
application{  
    mainClass='StartApp'  
}  
dependencies {  
    implementation project(':A')  
}
```



Gradle Build Tool

- Proiectul Root: `build.gradle`

```
subprojects {  
    //Configurări comune tuturor subproiectelor  
}  
  
project(':A') {  
    //Configurări specifice proiectului A  
}  
  
project(':B') {  
    //Configurări specifice proiectului B  
}
```