Ingineria Programării

Cursul 4 – 15,16 Martie 2022

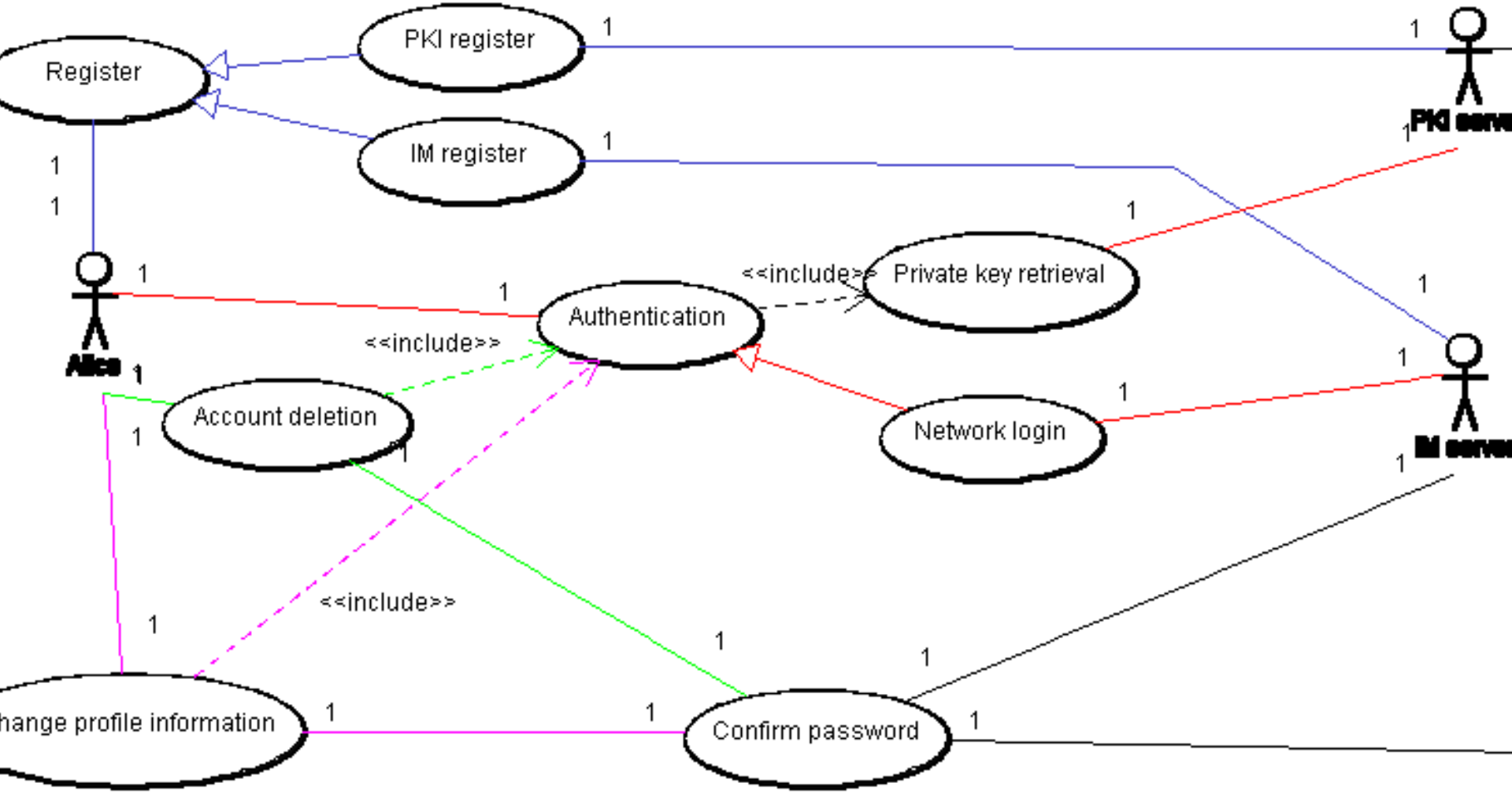
adiftene@info.uaic.ro

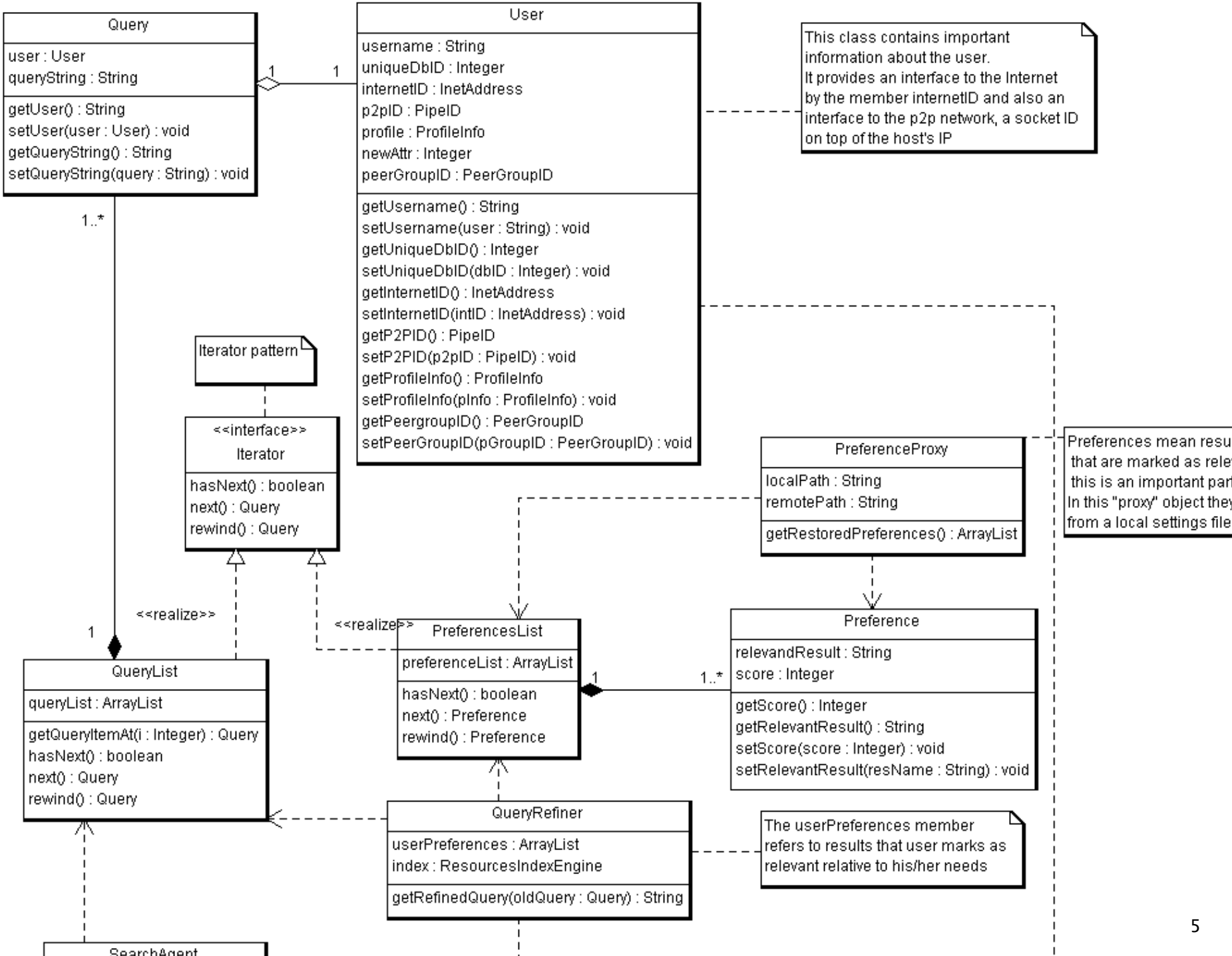
Cuprins

- Din Cursurile trecute…
- Diagrame UML – Exemple
- C4 Model
- Forward & Reverse Engineering

Din cursurile trecute…

- Diagrame
- Diagrame UML
- Diagrame Use Case
- Diagrame de Clase


Register
PKI register
IM register
Alice
PKI server
IM server
Authentication
Private key retrieval
<<include>>
<<include>>
Account deletion
Network login
<<include>>
hange profile information
Confirm password
1


Query
user : User
queryString : String
getUser() : String
setUser(user : User) : void
getQueryString() : String
setQueryString(query : String) : void
1
1
User
username : String
uniqueDbID : Integer
internetID : InetAddress
p2pID : PipeID
profile : ProfileInfo
newAttr : Integer
peerGroupID : PeerGroupID
getUsername() : String
setUsername(user : String) : void
getUniqueDbID() : Integer
setUniqueDbID(dbID : Integer) : void
getInternetID() : InetAddress
setInternetID(intID : InetAddress) : void
getP2PID() : PipeID
setP2PID(p2pID : PipeID) : void
getProfileInfo() : ProfileInfo
setProfileInfo(pInfo : ProfileInfo) : void
getPeergroupID() : PeerGroupID
setPeerGroupID(pGroupID : PeerGroupID) : void
This class contains important information about the user.
It provides an interface to the Internet by the member internetID and also an interface to the p2p network, a socket ID on top of the host's IP
1..*
Iterator pattern
<<interface>>
Iterator
hasNext() : boolean
next() : Query
rewind() : Query
PreferenceProxy
localPath : String
remotePath : String
getRestoredPreferences() : ArrayList
Preferences mean resu
that are marked as rele
this is an important par
In this "proxy" object they
from a local settings file
<<realize>>
<<realize>>
1
QueryList
queryList : ArrayList
getQueryItemAt(i : Integer) : Query
hasNext() : boolean
next() : Query
rewind() : Query
PreferencesList
preferenceList : ArrayList
hasNext() : boolean
next() : Preference
rewind() : Preference
1
1..*
Preference
relevandResult : String
score : Integer
getScore() : Integer
getRelevantResult() : String
setScore(score : Integer) : void
setRelevantResult(resName : String) : void
QueryRefiner
userPreferences : ArrayList
index : ResourcesIndexEngine
getRefinedQuery(oldQuery : Query) : String
The userPreferences member refers to results that user marks as relevant relative to his/her needs
SearchAgent

UML2.0 – 13 Tipuri de Diagrame

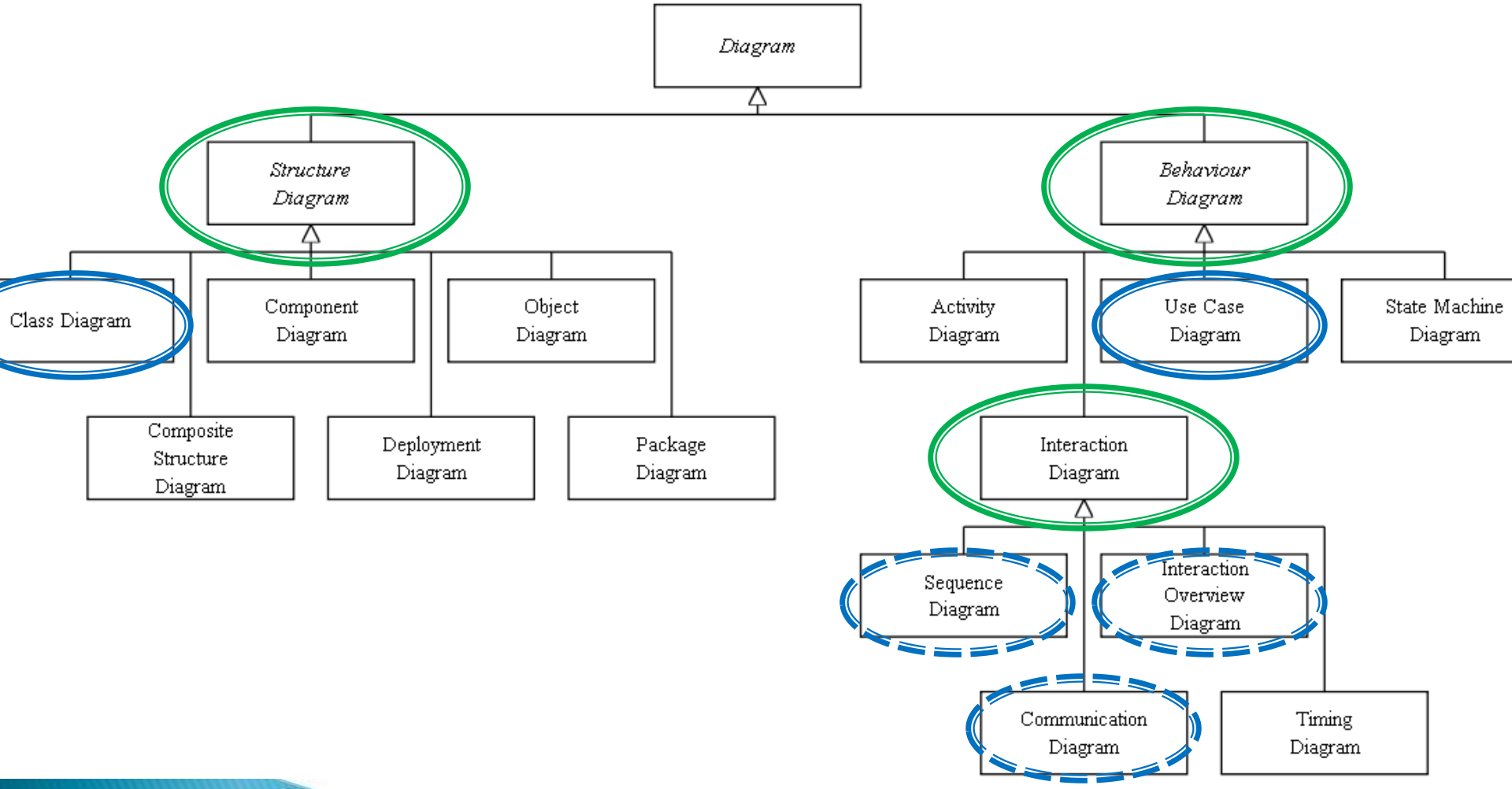
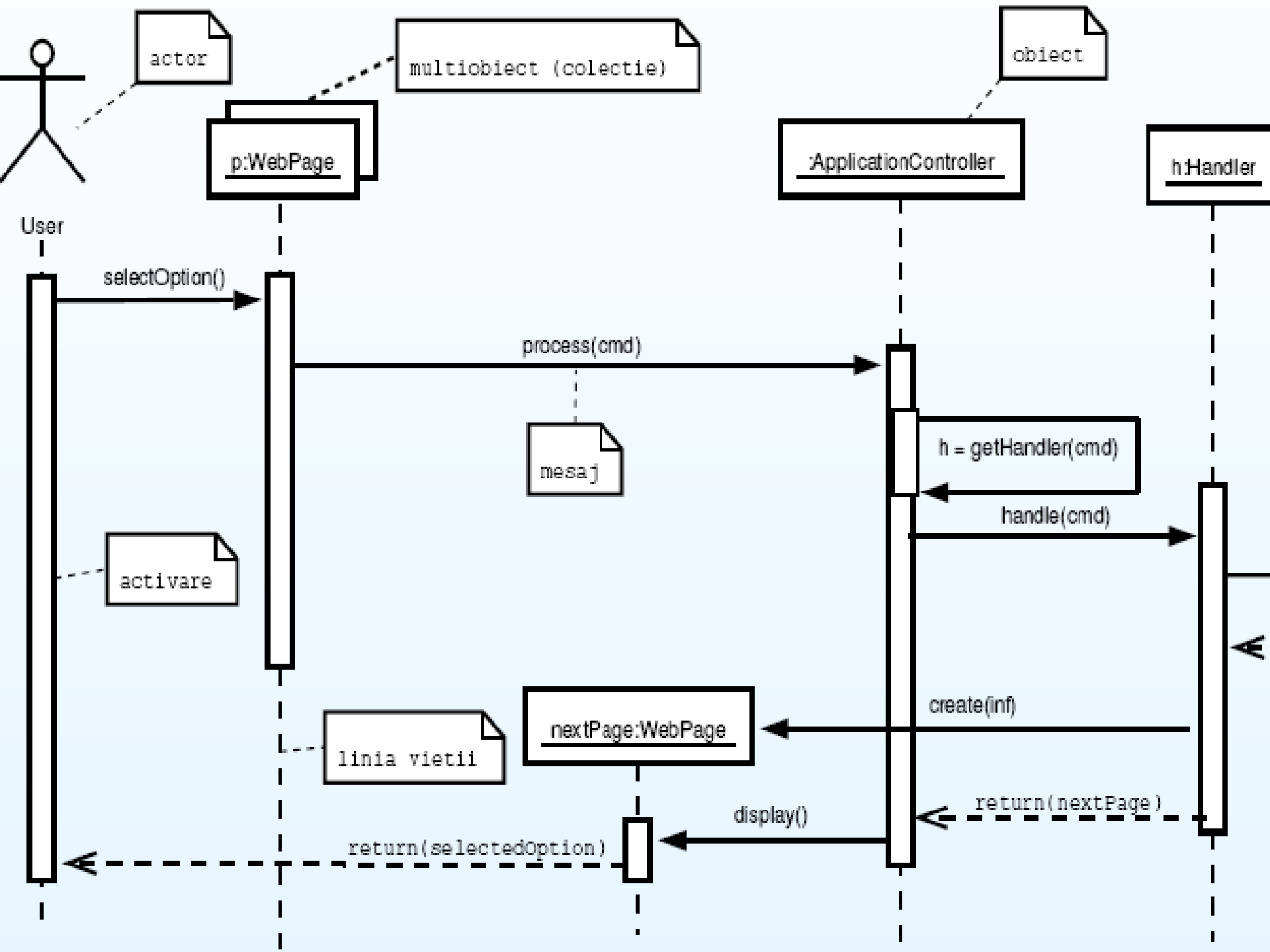
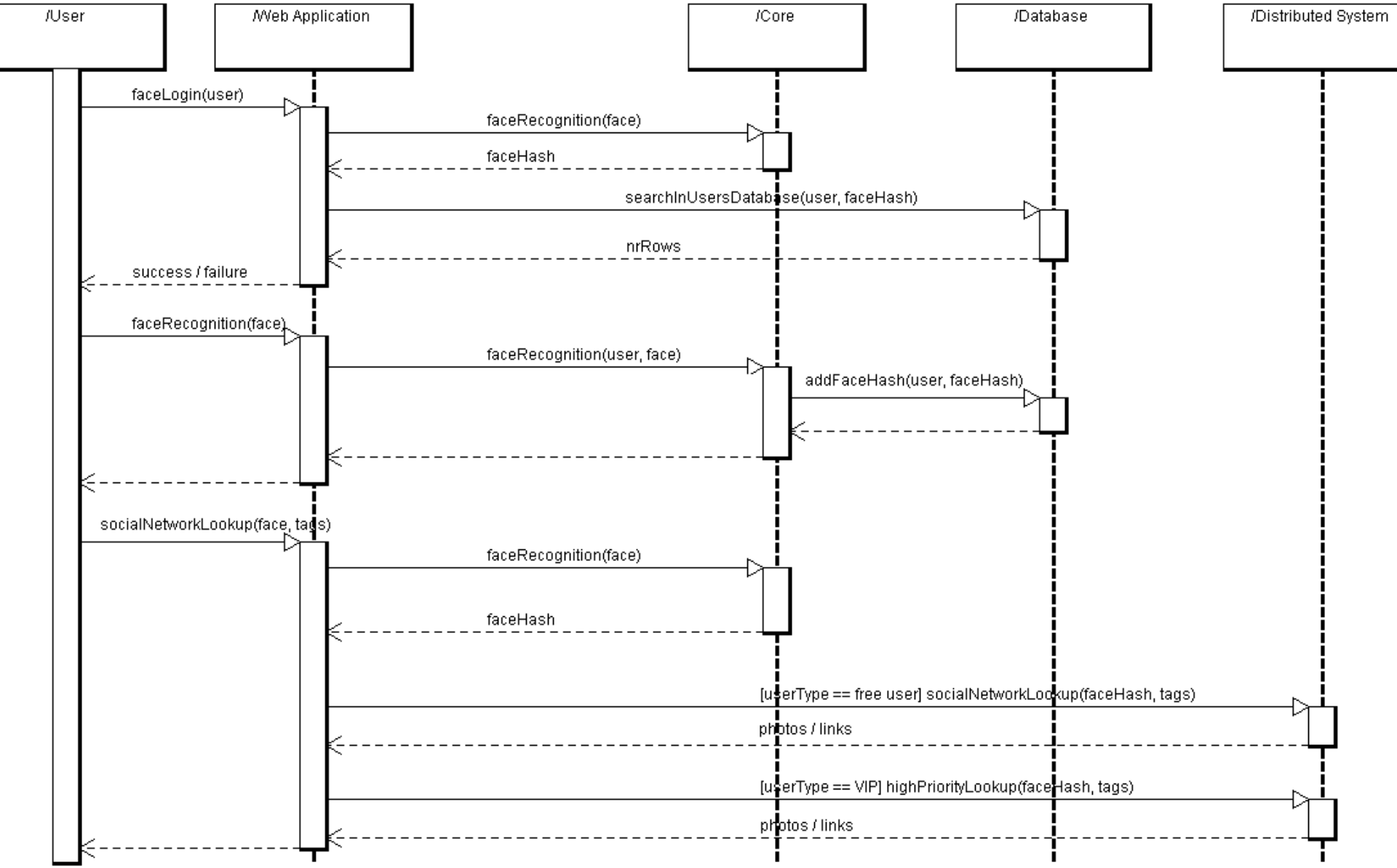


Diagrama de Secvenţă

- **Diagrama de secvenţă** curprinde secvenţa acţiunilor care au loc în sistem, invocarea metodelor fiecărui obiect ca şi ordinea în timp în care aceste invocări au loc

- O diagramă de secvenţă este bidimensională
  - Pe axa verticală se prezintă viaţa obiectului
    - linia vieţii obiectelor (grafic: linie punctată)
    - perioada de activare în care un obiect preia controlul execuţiei (grafic: dreptunghi pe linia vieţii)
  - Pe axa orizontală se arată secvenţa creării sau invocărilor
    - mesaje ordonate în timp (grafic: săgeţi)

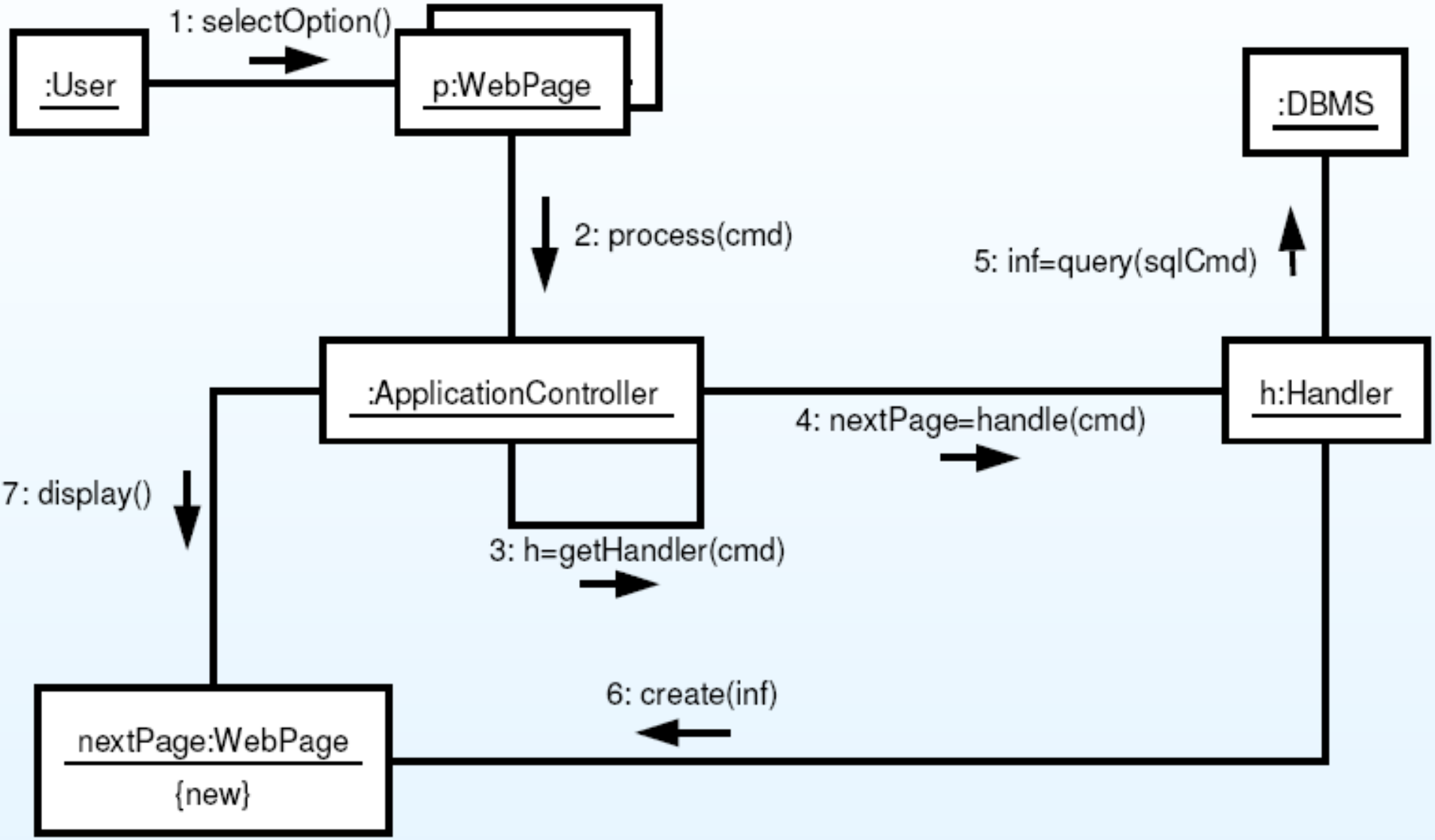

actor
multiobiect (colectie)
obiect
p:WebPage
:ApplicationController
h:Handler
User
selectOption()
process(cmd)
mesaj
h = getHandler(cmd)
handle(cmd)
activare
create(inf)
nextPage:WebPage
linia vietii
return(nextPage)
display()
return(selectedOption)


/User
/Web Application
/Core
/Database
/Distributed System
faceLogin(user)
faceRecognition(face)
faceHash
searchInUsersDatabase(user, faceHash)
nrRows
success / failure
faceRecognition(face)
faceRecognition(user, face)
addFaceHash(user, faceHash)
socialNetworkLookup(face, tags)
faceRecognition(face)
faceHash
[userType == free user] socialNetworkLookup(faceHash, tags)
photos / links
[userType == VIP] highPriorityLookup(faceHash, tags)
photos / links

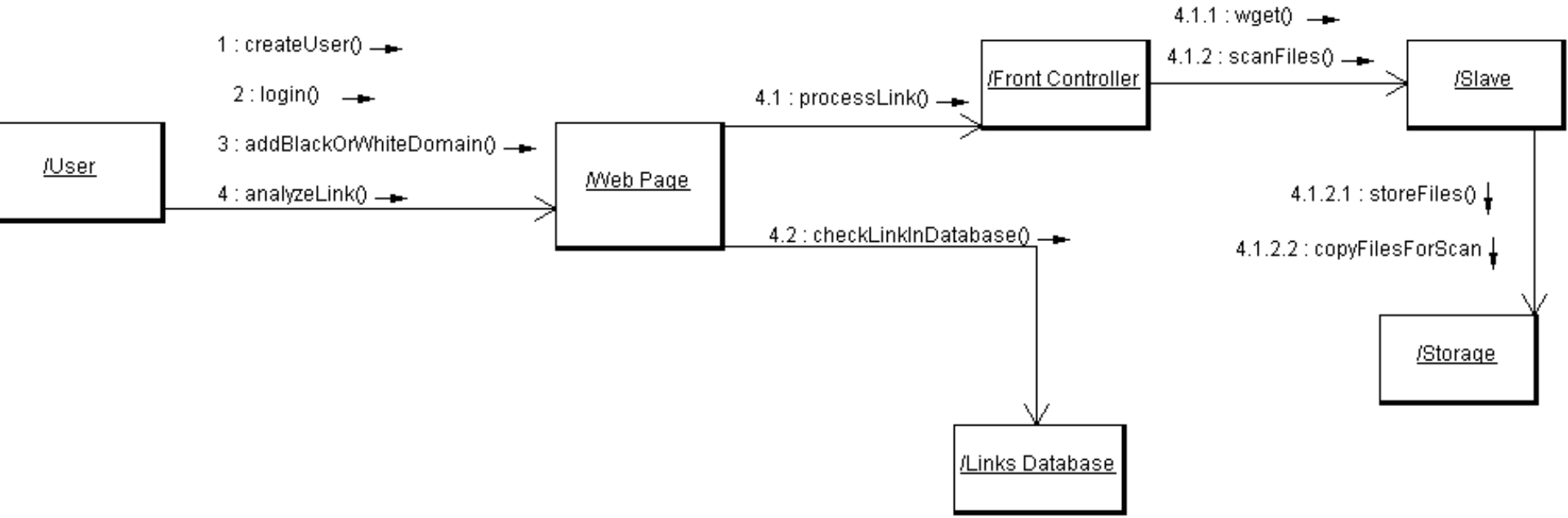
Diagramă de Colaborare

- Pune accentul pe organizarea structurală a obiectelor care participă la interacţiune
- Ilustrează mai bine ramificări complexe, iteraţii şi comportament concurent
- Poate conţine:
  - Obiecte, clase, actori
  - Legături între acestea
  - Mesaje

Exemplul 1


1: selectOption()
:User
p:WebPage
:DBMS
2: process(cmd)
5: inf=query(sqlCmd)
:ApplicationController
h:Handler
4: nextPage=handle(cmd)
7: display()
3: h=getHandler(cmd)
6: create(inf)
nextPage:WebPage
{new}

Exemplul 3

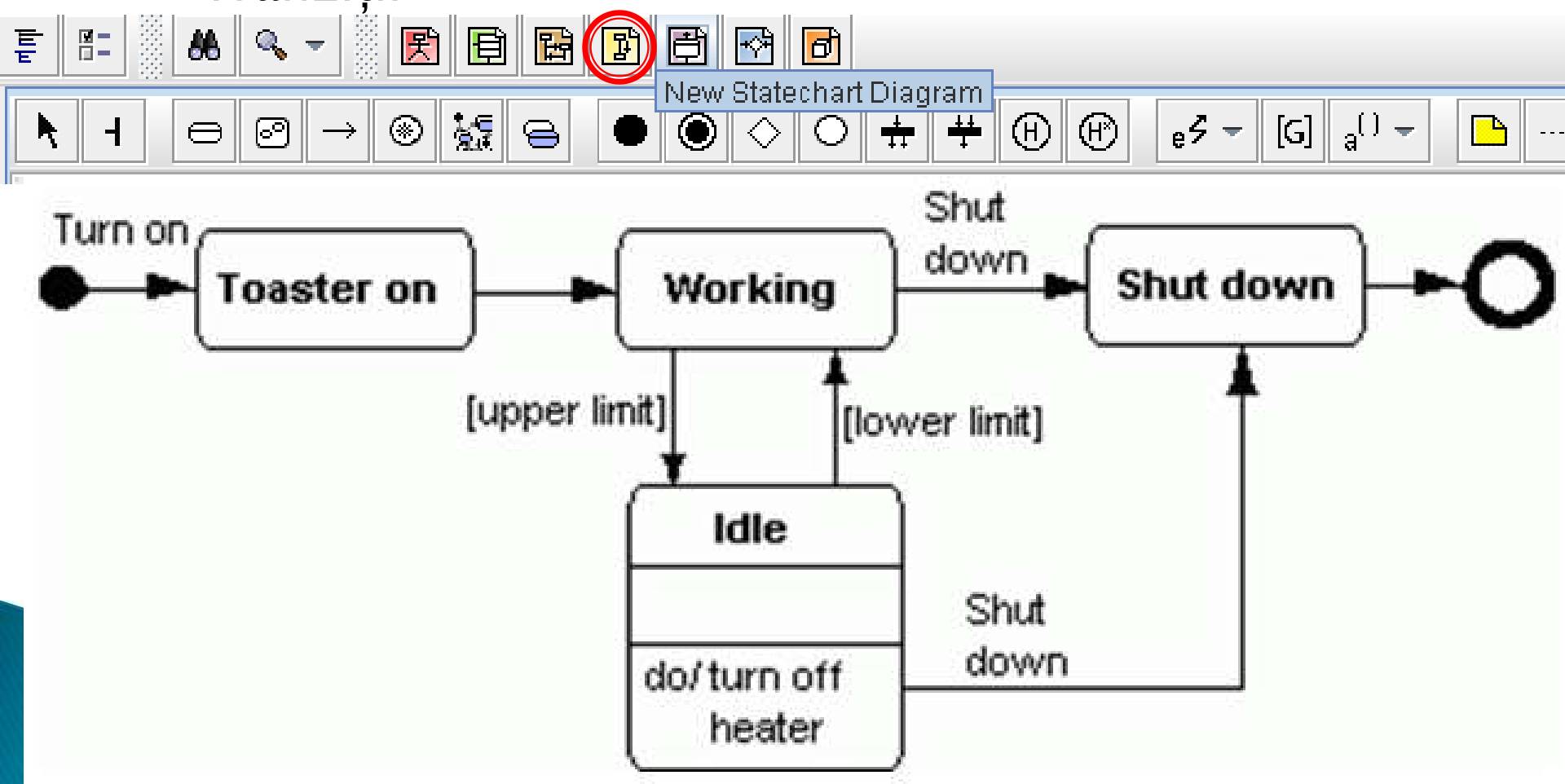

1 : createUser()
2 : login()
3 : addBlackOrWhiteDomain()
4 : analyzeLink()
/User
/Web Page
4.1 : processLink()
/Front Controller
4.1.1 : wget()
4.1.2 : scanFiles()
/Slave
4.1.2.1 : storeFiles()
4.1.2.2 : copyFilesForScan
/Storage
4.2 : checkLinkInDatabase()
/Links Database

Diagrame comportamentale

- Diagrame de stări, diagrame de activități
- Elemente de bază
  - Eveniment
  - Acțiune
  - Activitate

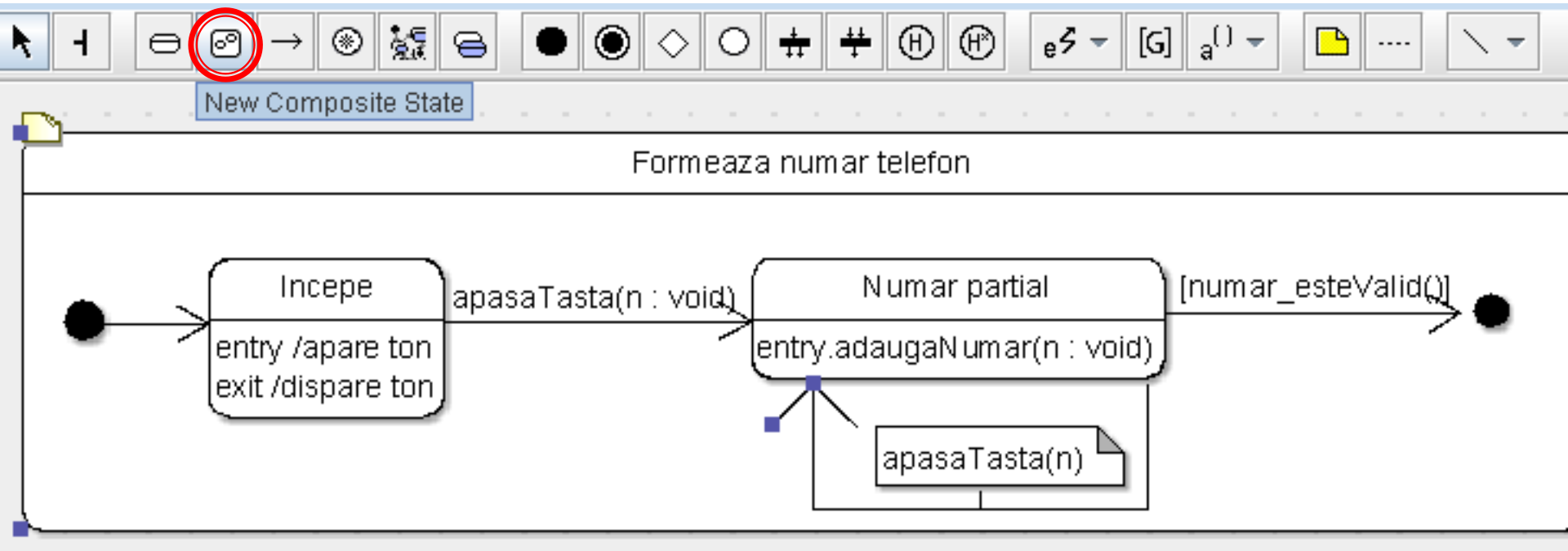
Diagramă de Stări

- Conține:
  - Stări
  - Tranziții



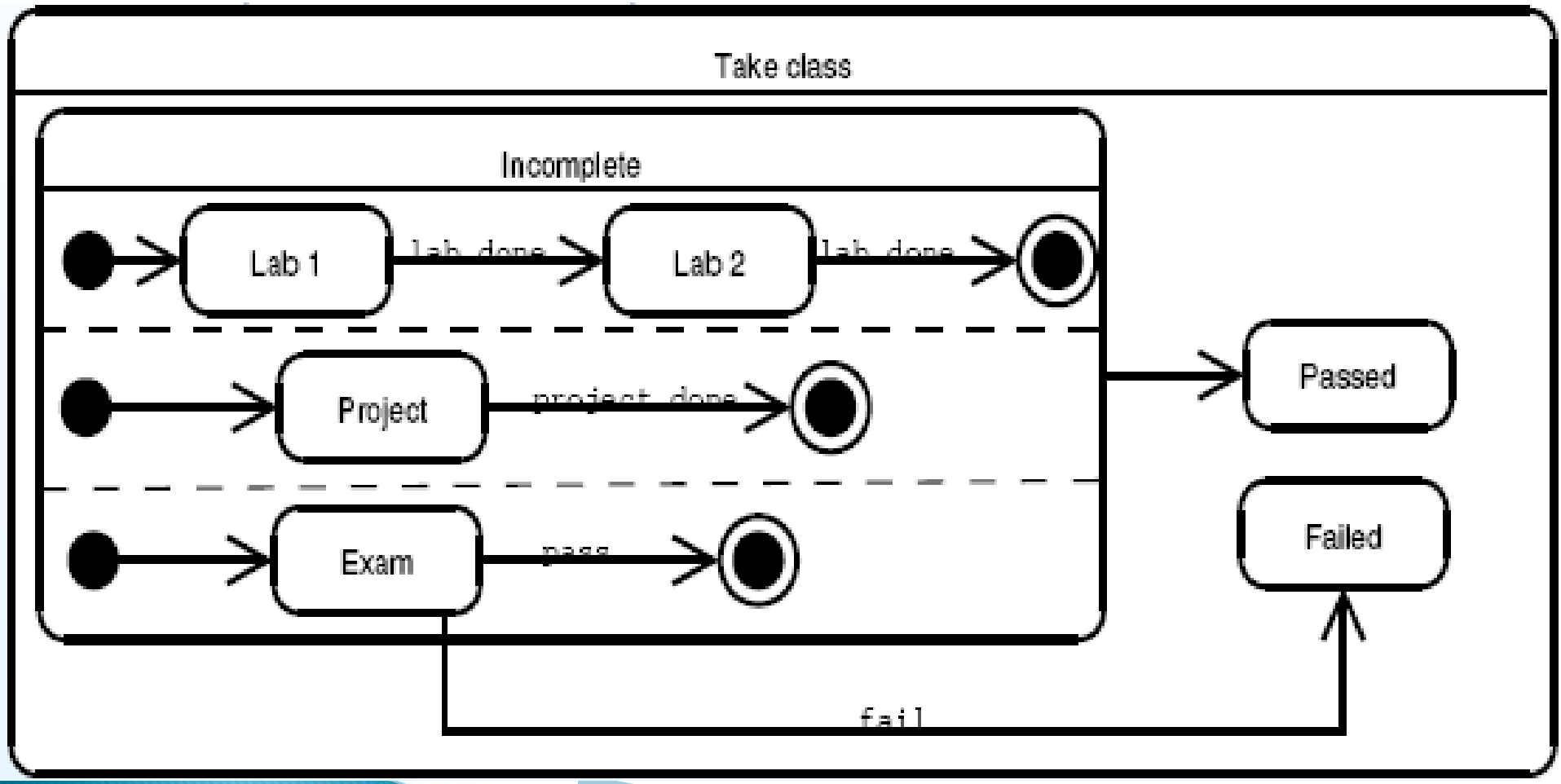
Exemplu de Stare compusă 1

- Stare compusă cu substări **secvențial** active:



Exemplu de Stare compusă 2

- Stare compusă cu substări **paralel** active:



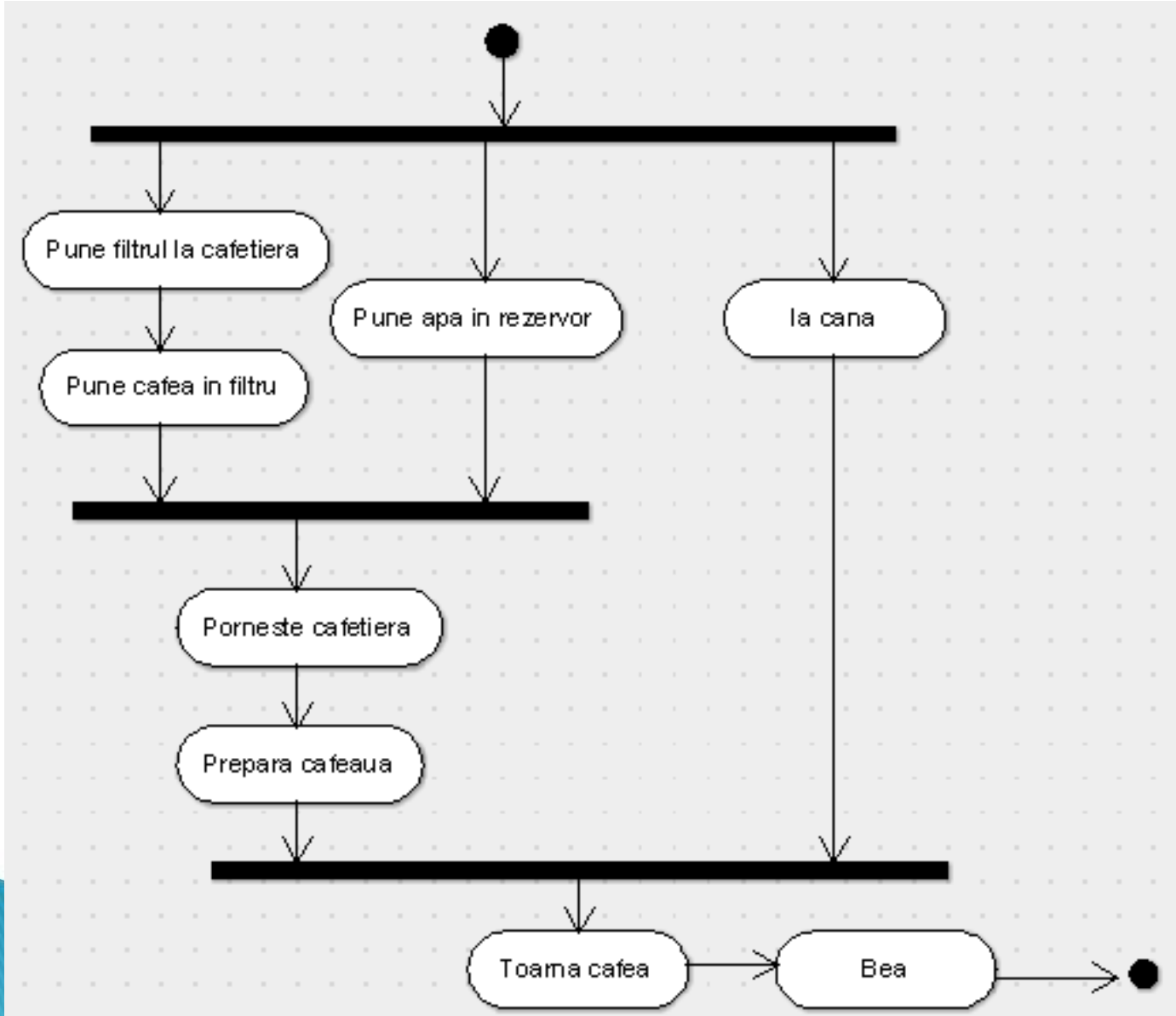
Diagramă de Activități (Activity Diagram)

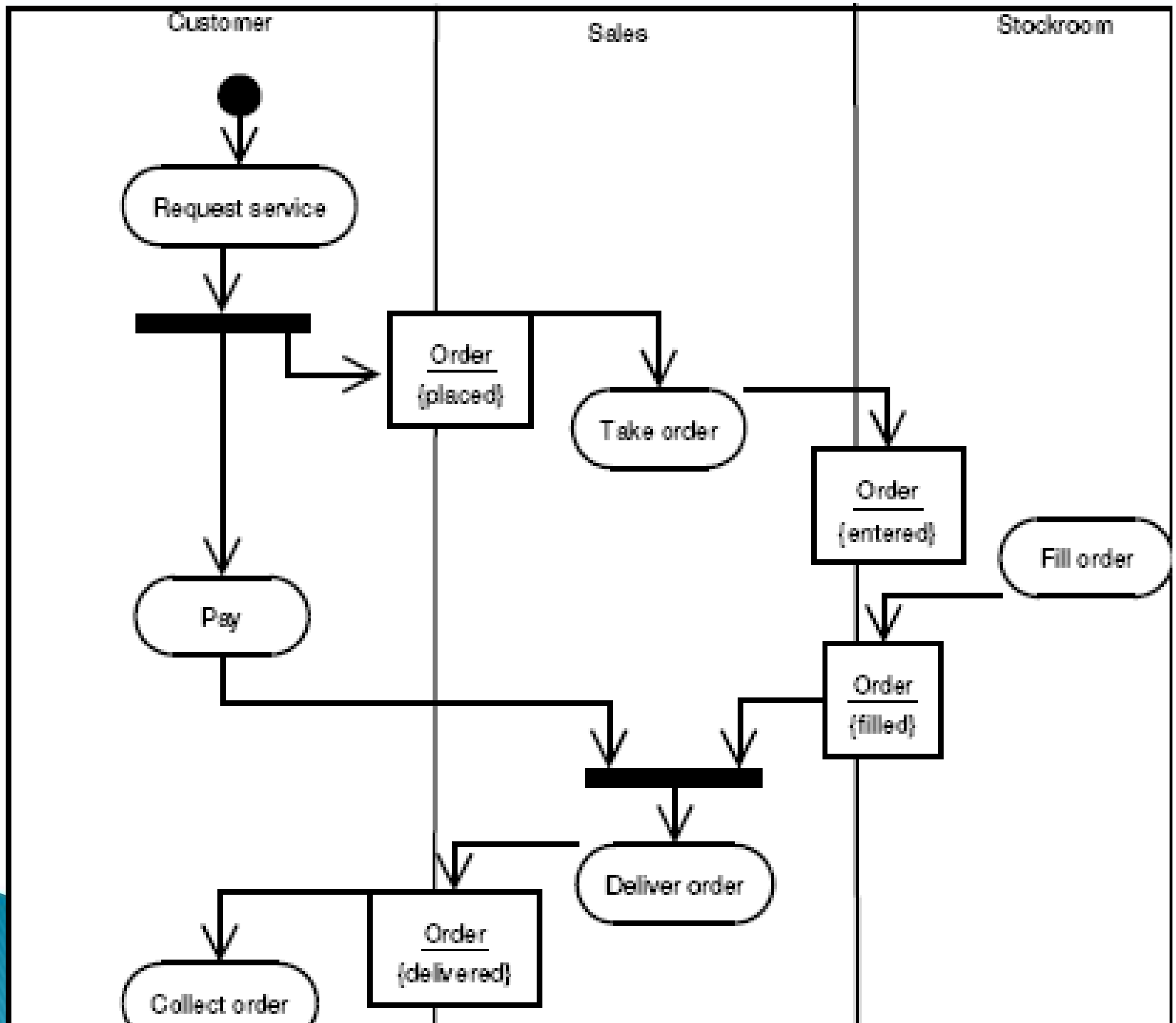
- Folosită pentru **a modela dinamica unui proces sau a unei operații**

- Evidențiază controlul execuției de la o activitate la alta

- Se atașează:
  - Unei clase (modelează un caz de utilizare)
  - Unui pachet
  - Implementării unei operații

Exemplu de DA (Sincronizare)

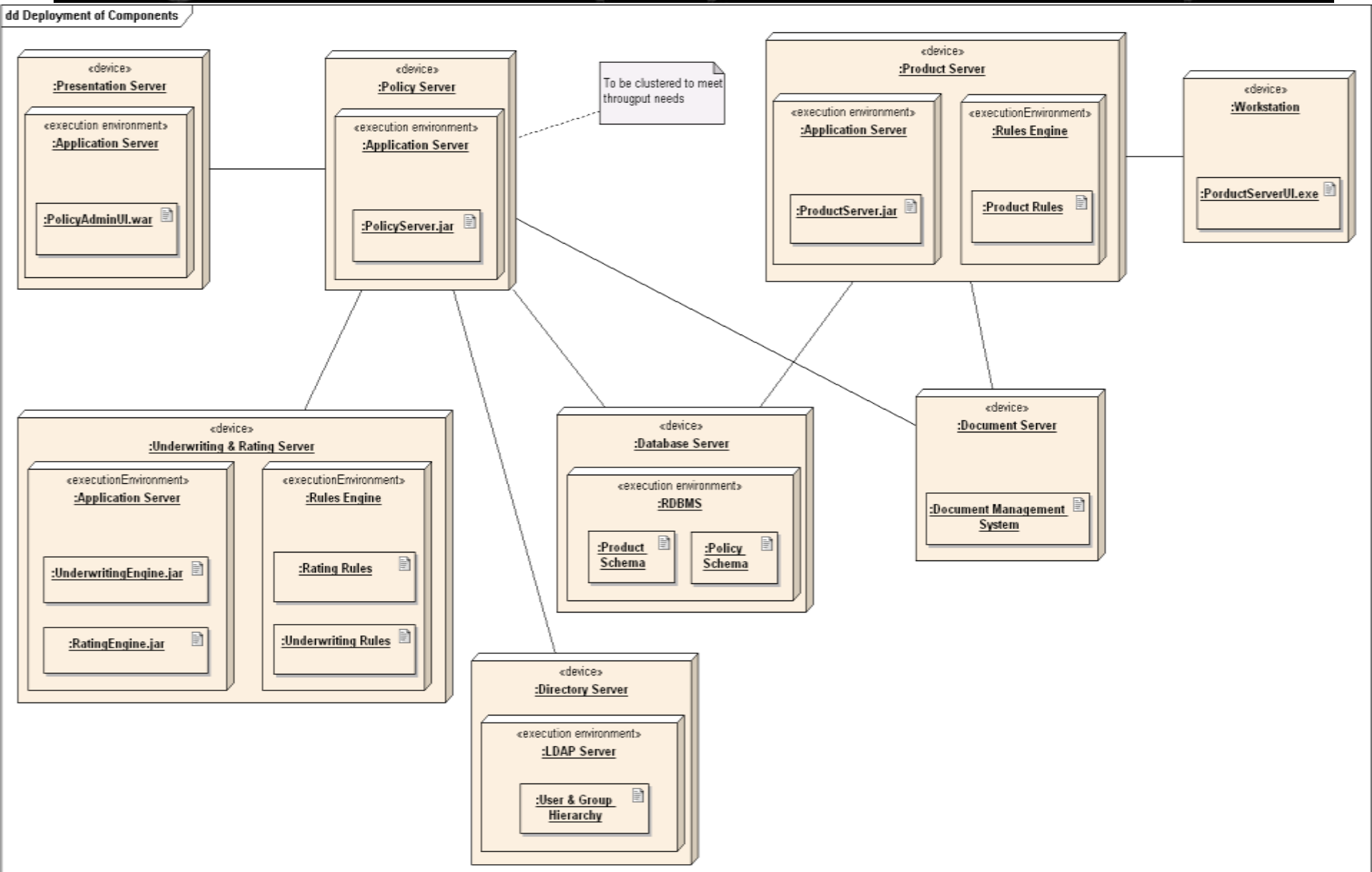



Customer
Sales
Stockroom
Request service
Order
{placed}
Take order
Order
{entered}
Fill order
Pay
Order
{filled}
Deliver order
Order
{delivered}
Collect order

Diagrame de deployment

- Modelează mediul hardware în care va funcționa proiectul
- Exemplu: pentru a descrie un site web o diagramă de deployment va conține **componentele hardware**
  - server-ul web,
  - server-ul de aplicații,
  - server-ul de baze de date
- **Componentele software** de pe fiecare din acestea
  - Aplicația web
  - Baza de date
- Modul în care acestea sunt conectate:
  - JDBC, REST, RMI

Diagramă de deployment – Exemplu 1



Diagrame de Pachete (Package Diagram)

- Pachetul:
  - Este un container logic pentru elemente între care se stabilesc legături
  - Definește un spațiu de nume
  - Toate elementele UML pot fi grupate în pachete (cel mai des pachetele sunt folosite pentru a grupa clase)
  - Un pachet poate conține subpachete => se creează o structură arborescentă (similară cu organizarea fișierele/directoarelor)

Diagrame de Pachete 2

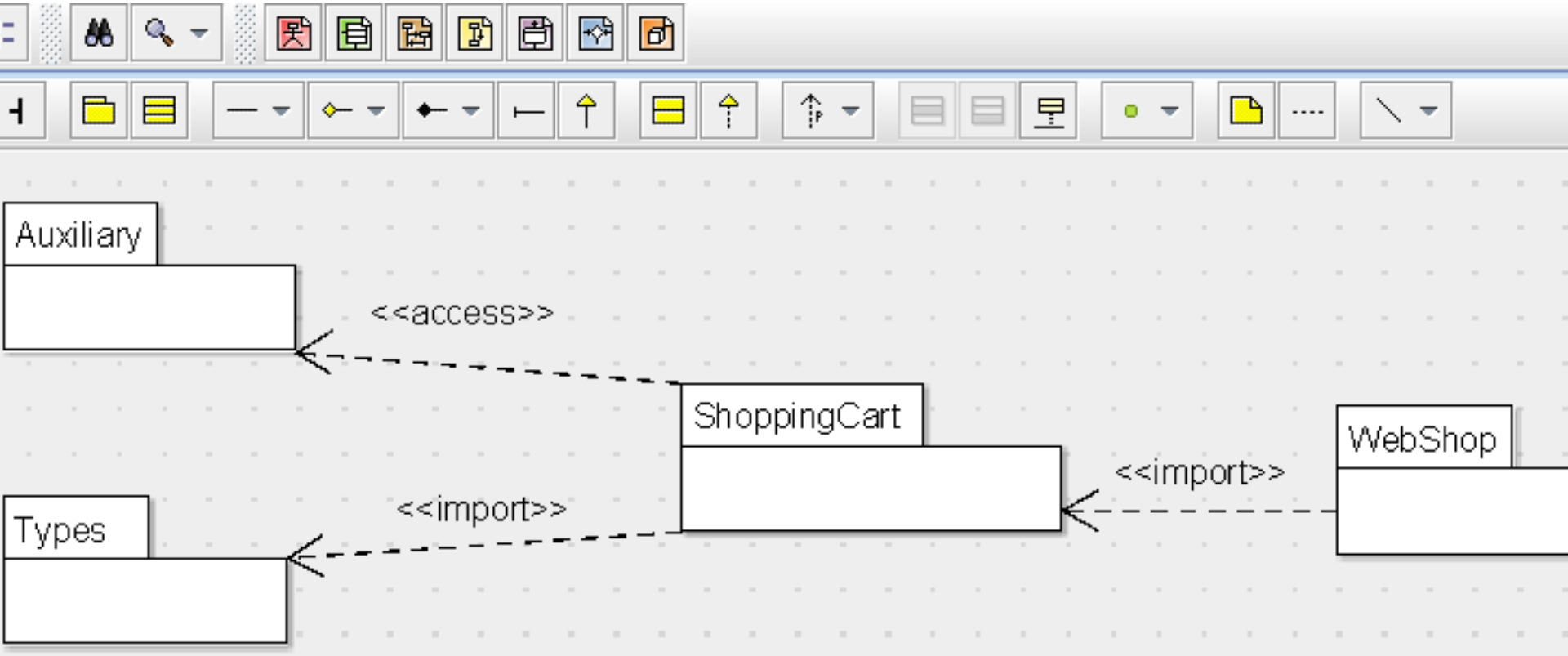
- Relații:
  - dependență <<access>> = import privat
  - dependență <<import>> = import public

- Ambele relații permit folosirea elementelor aflate în pachetul destinație de către elementele aflate în pachetul sursă fără a fi necesară calificarea numelor elementelor din pachetul destinație (similar directivei **import** din java)

- Aceste tipuri de diagrame se realizează în cadrul diagramelor de clasă

Exemplu de Diagramă de Pachete

- Elementele din Types sunt importate în ShoppingCart și apoi sunt importate mai departe de către WebShop
- Elementele din Auxiliary pot fi accesate însă doar din ShoppingCart și nu pot fi referite folosind nume necalificate din WebShop



Utilitatea diagramelor de pachete

- Împart sisteme mari în subsisteme mai mici și mai ușor de gestionat
- Permit dezvoltare paralelă iterativă
- Definirea unor interfețe clare între pachete promovează refolosirea codului (ex. pachet care oferă funcții grafice, pachet care oferă posibilitatea conectării la BD, etc...)

Recomandări în realizarea diagramelor UML

- Diagramele să nu fie nici prea complicate, dar nici prea simple: scopul este comunicarea eficientă
- Dați nume sugestive elementelor componente
- Aranjați elementele astfel încât liniile să nu se intersecteze
- Încercați să nu arătați prea multe tipuri de relații odată (evitați diagramele foarte complicate)
- Dacă este nevoie, realizați mai multe diagrame de același tip

The C4 Model for Software Architecture

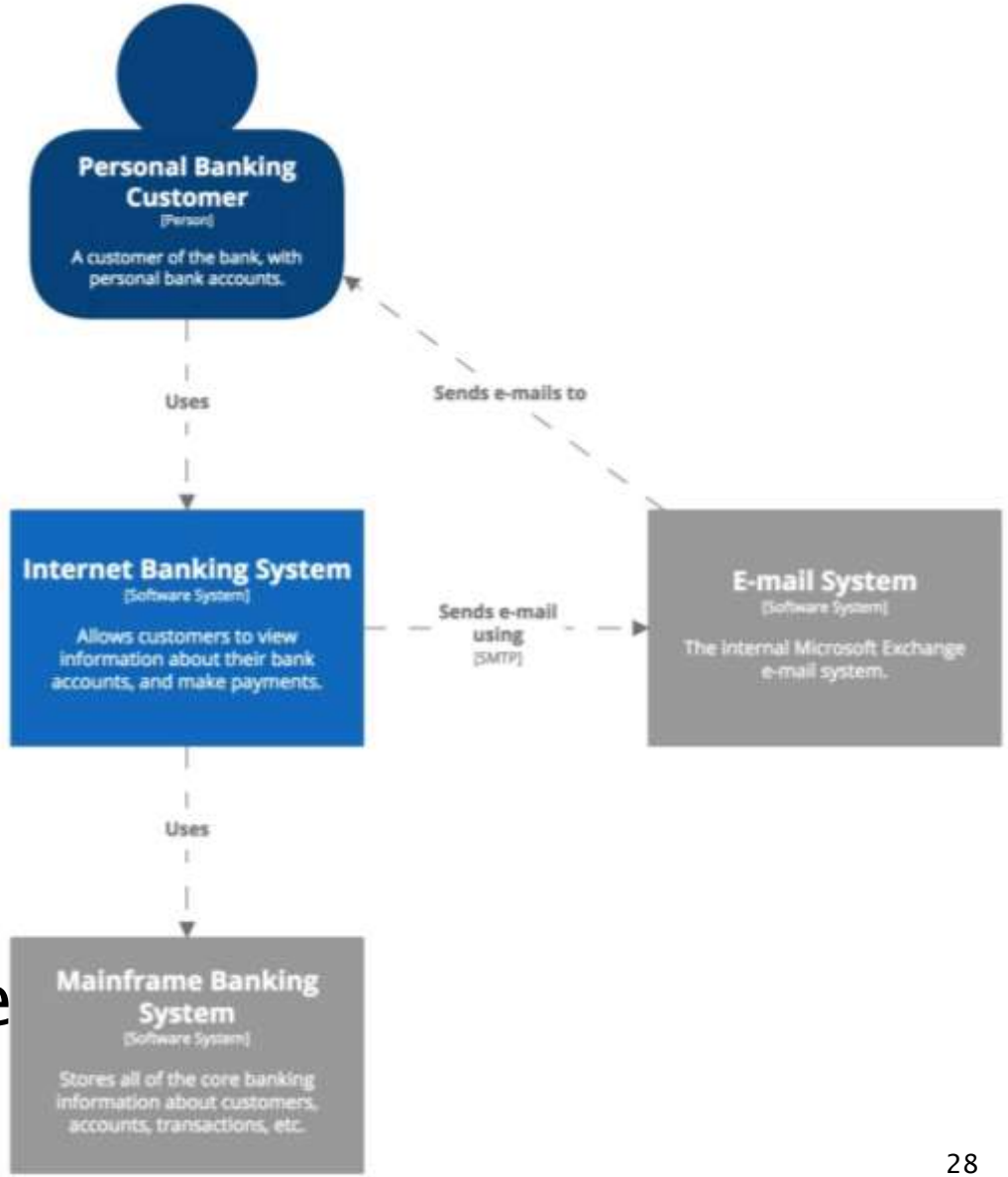
- **Context, Containers, Components** and **Code**

- **Provides different levels of abstraction**, each of which is relevant to a different audience

- Avoid ambiguity in your diagrams by **including a sufficient amount of text** as well as a key/legend for the notation you use

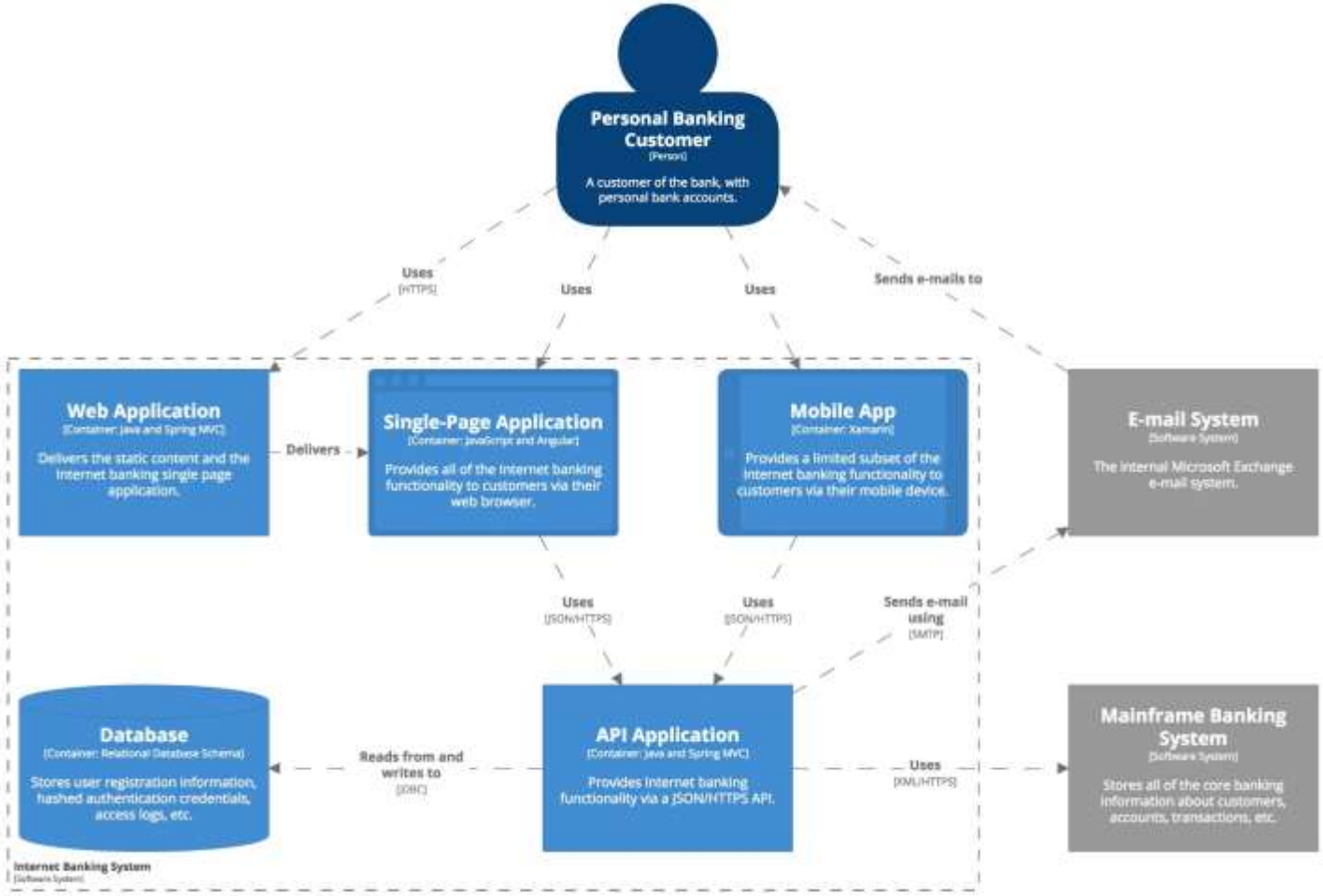
C4 Model – Level 1: System context diagram

- Shows the software system you are building
- How it fits into the world in terms of the people who use it and the other software systems it interacts with
- Colours – Systems already exist (the **grey** boxes) and those to be built (**blue**)



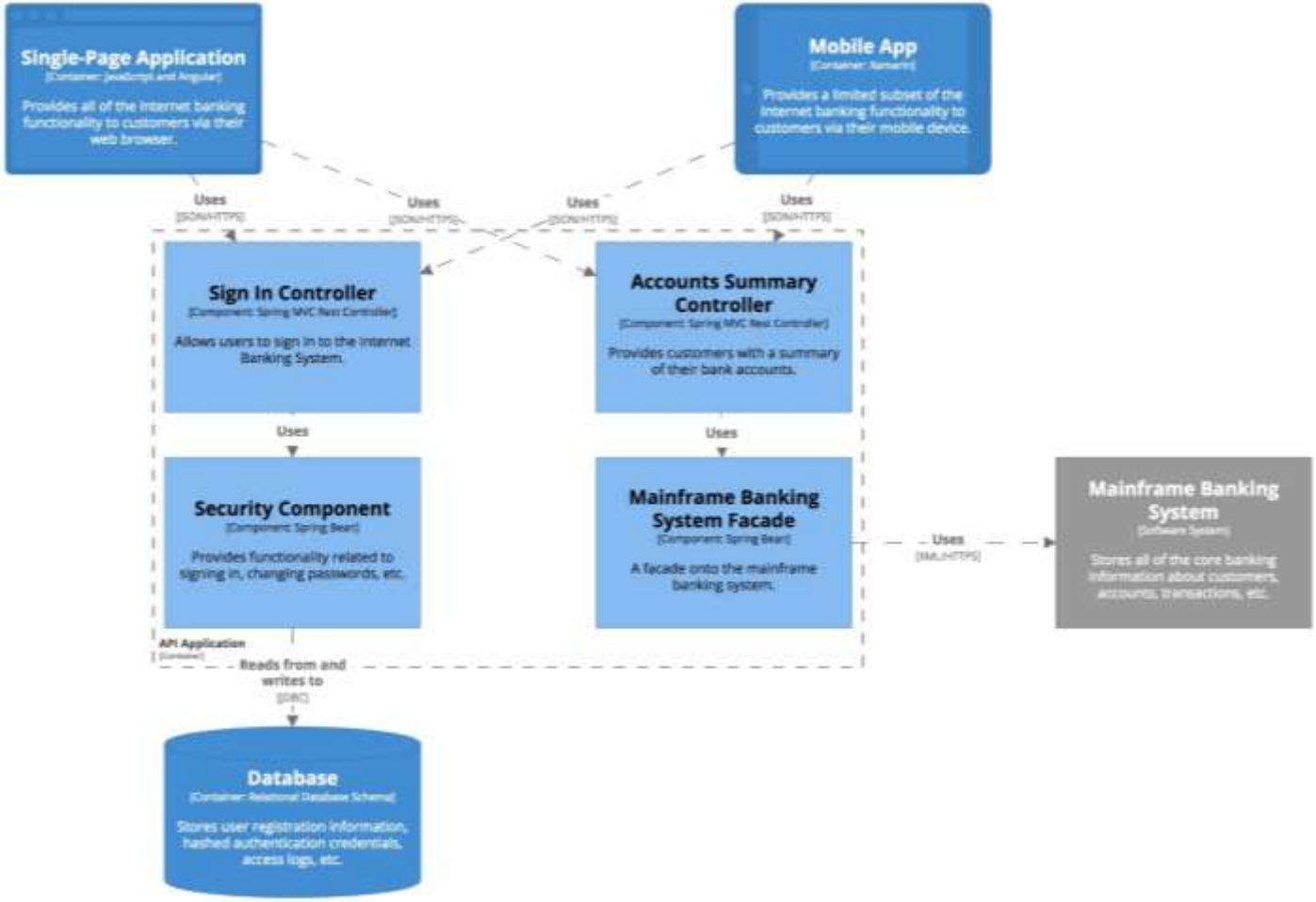
C4 Model – Level 2: Container diagram

- Zooms into the software system, and shows the containers (applications, data stores, microservices, etc.)



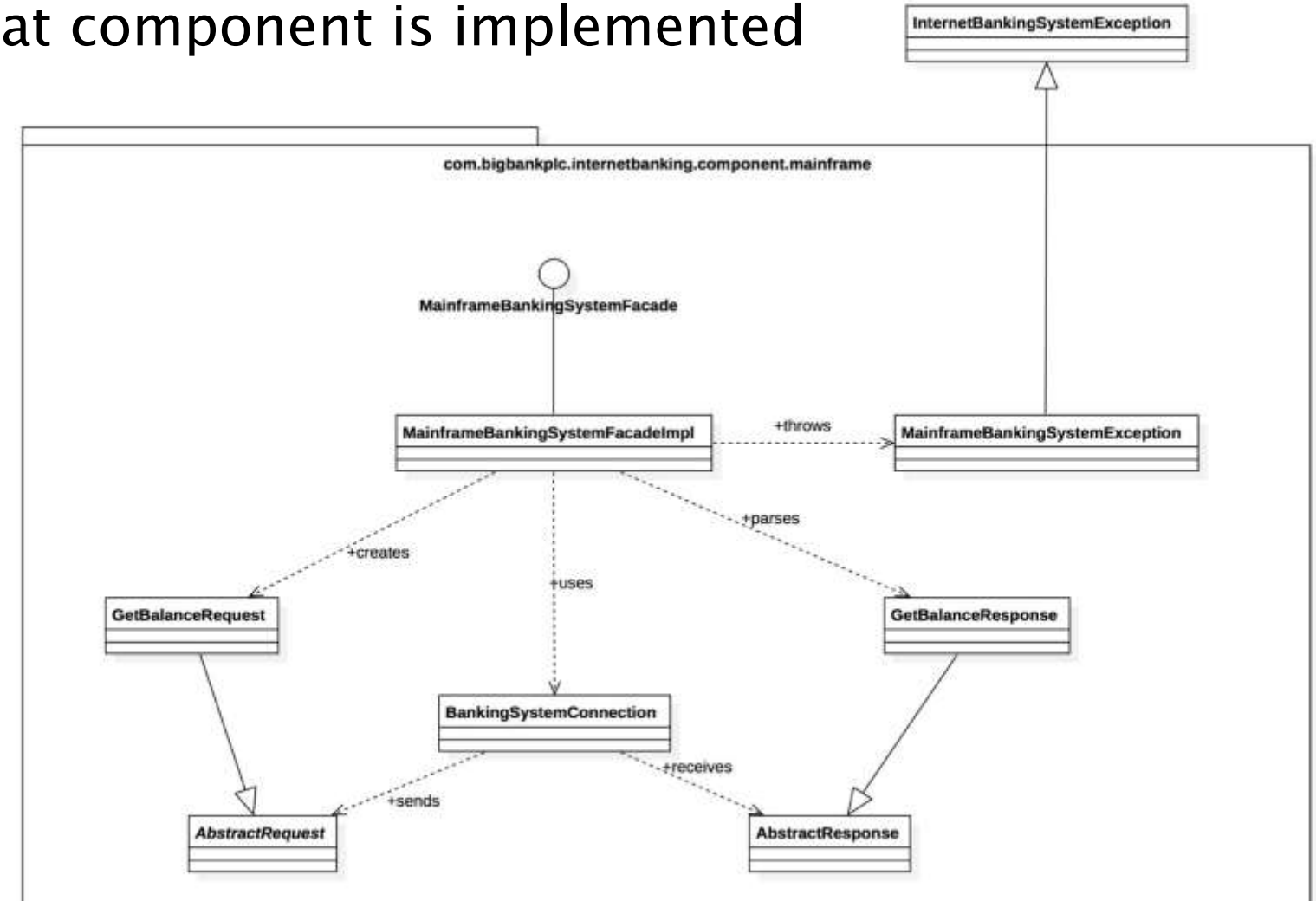
C4 Model – Level 3: Component diagram

- Zooms into an individual container to show the components inside it



C4 Model – Level 4: Code

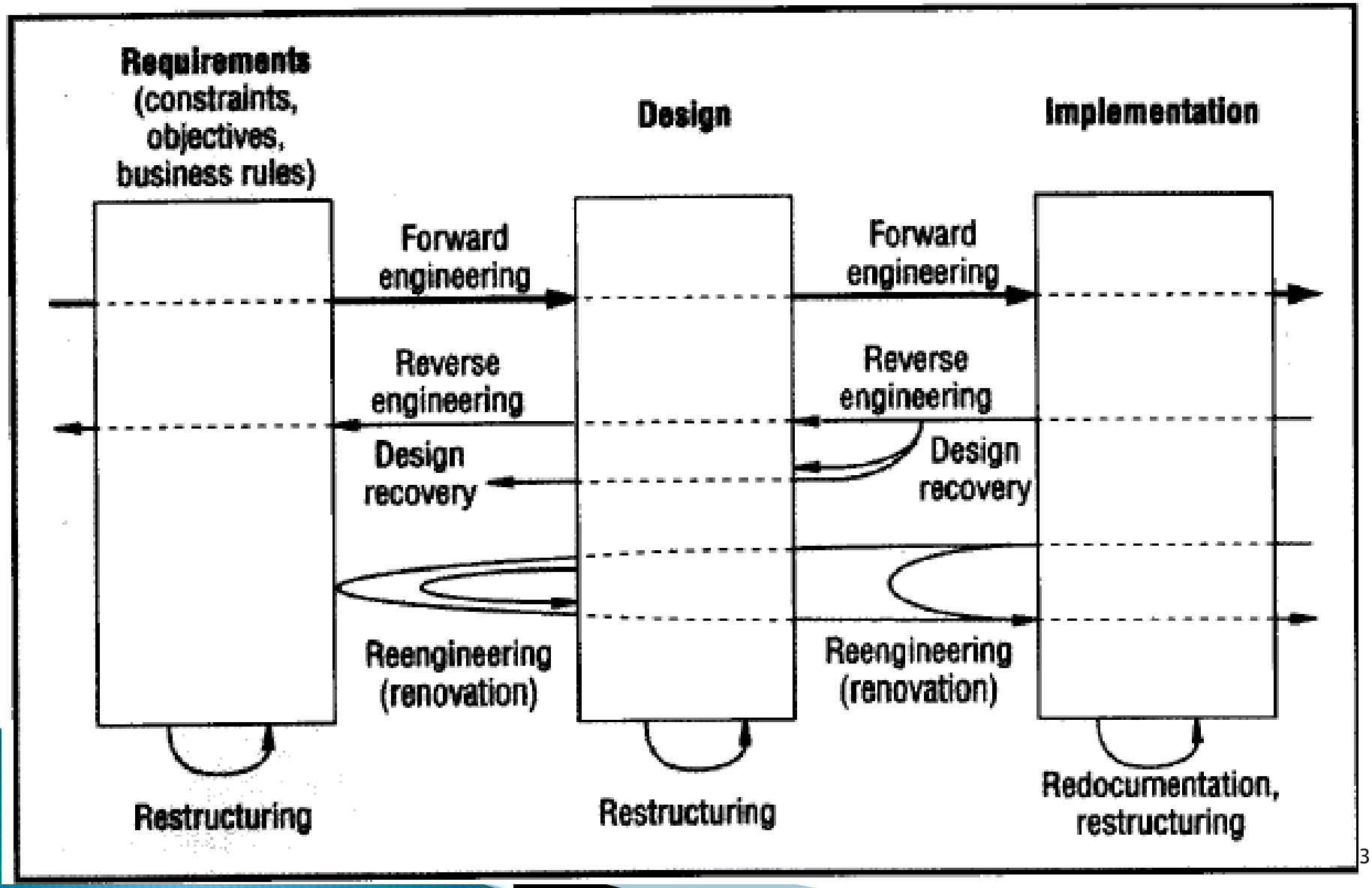
- Zoom into an individual component to show how that component is implemented



C4 Model – Links

- https://tobiashochguertel.github.io/c4-draw.io/
- https://structurizr.com/express
- https://www.infoq.com/articles/C4-architecture-model
- https://c4model.com/

Forward and Reverse Engineering



Forward Engineering

- A traditional process of moving from high-level abstractions and logical to the implementation-independent designs to the physical implementation of a system

- FE follows a sequence of going **from requirements through designing its implementation**

Reverse Engineering

- **Reverse engineering** (RE) is the process of discovering the technological principles of a device, object or system through analysis of its structure, function and operation
- *To try to make a new device or program that does the same thing without copying anything from the original*
- Reverse engineering has its origins in the analysis of hardware for commercial or military advantage

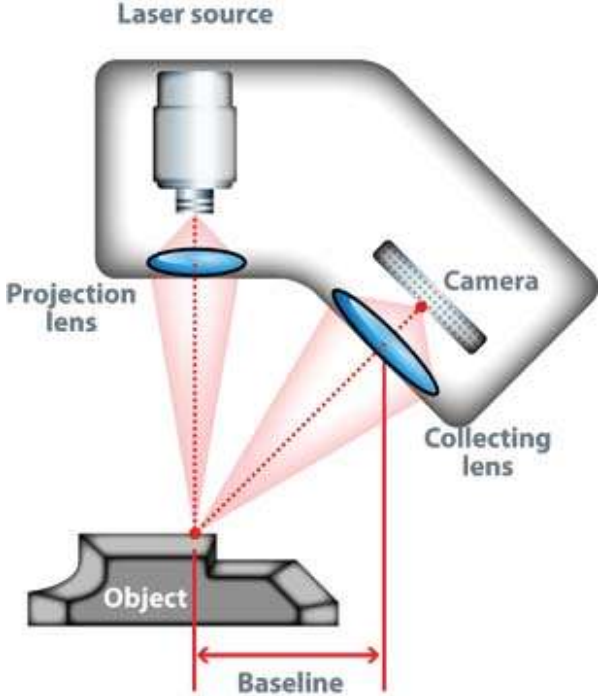
RE Motivation

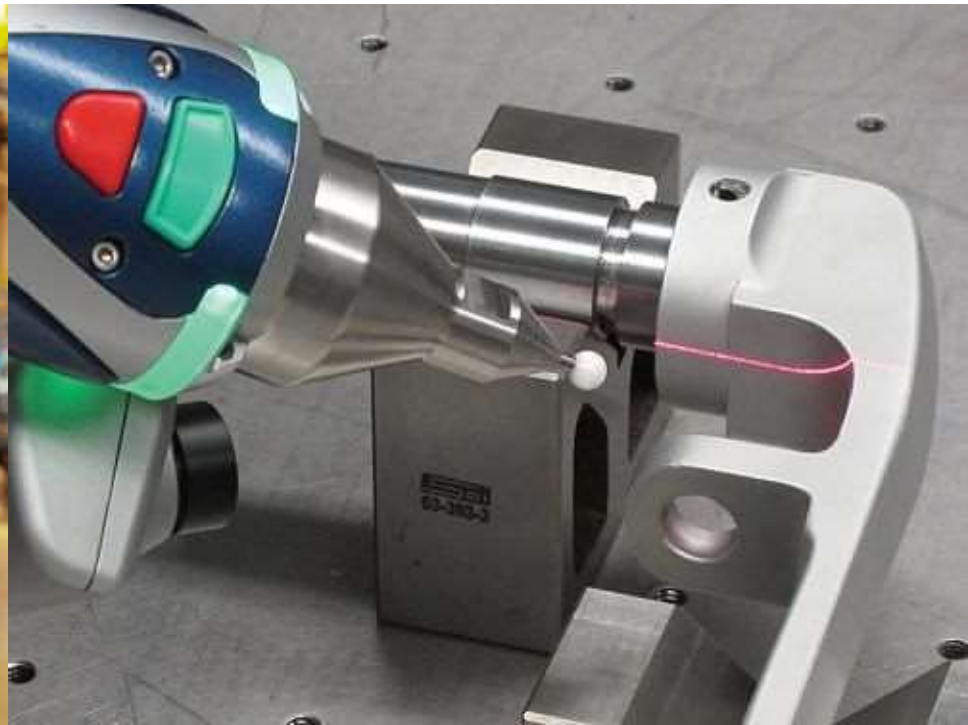
- Interoperability
- Lost documentation
- Product analysis
- Security auditing
- Removal of copy protection, circumvention of access restrictions
- Creation of unlicensed/unapproved duplicates
- Academic/learning purposes
- Curiosity
- Competitive technical intelligence (understand what your competitor is actually doing versus what they say they are doing)
- Learning: Learn from others mistakes

Types of RE

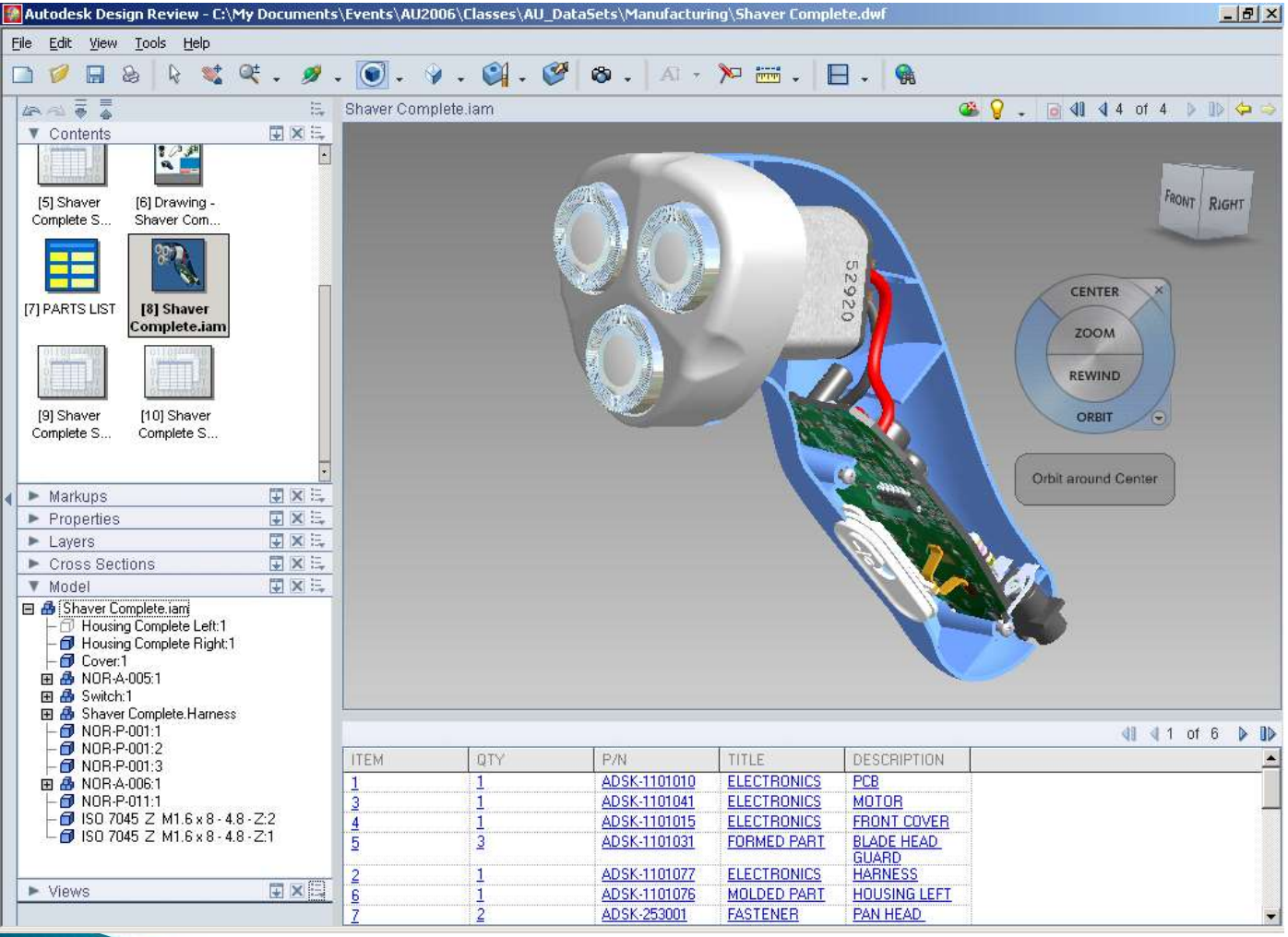
- RE1: Reverse engineering of mechanical devices
- RE2: Reverse engineering of integrated circuits/smart cards
- RE3: Reverse engineering for military applications
- RE4: Reverse engineering of software

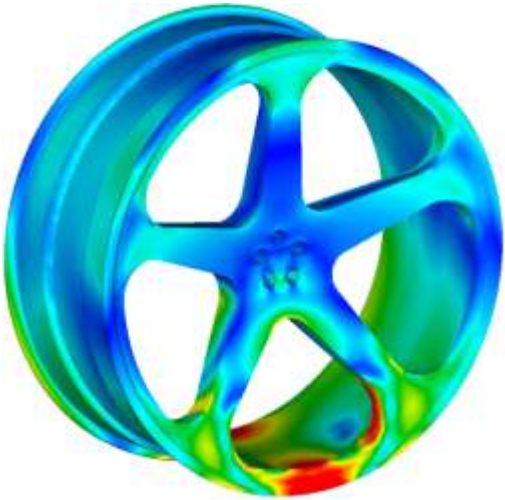
RE1: Scanere laser 3D

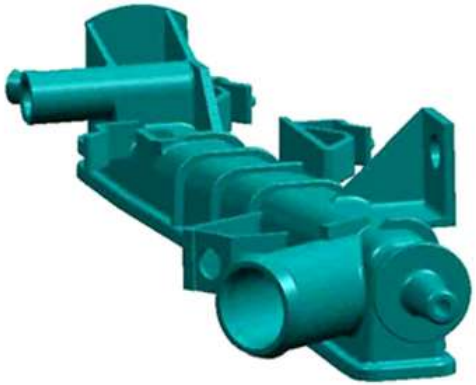




RE1: Servicii de modelare 3D CAD





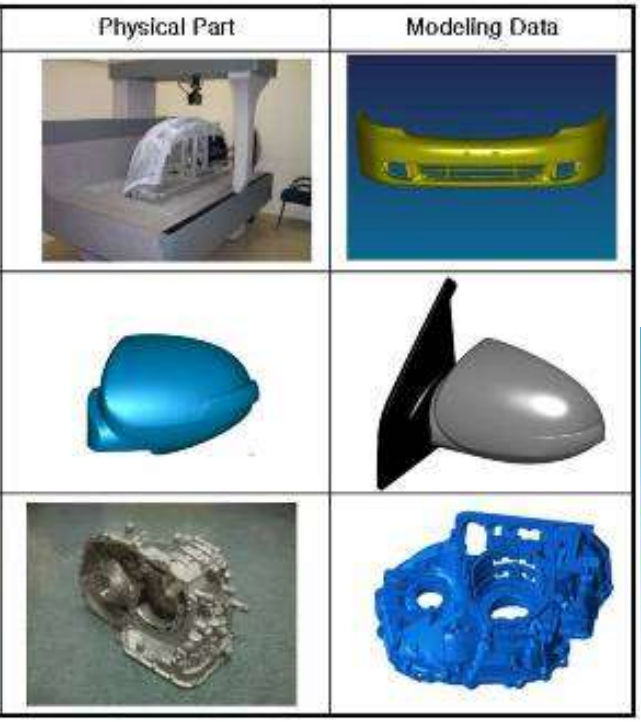


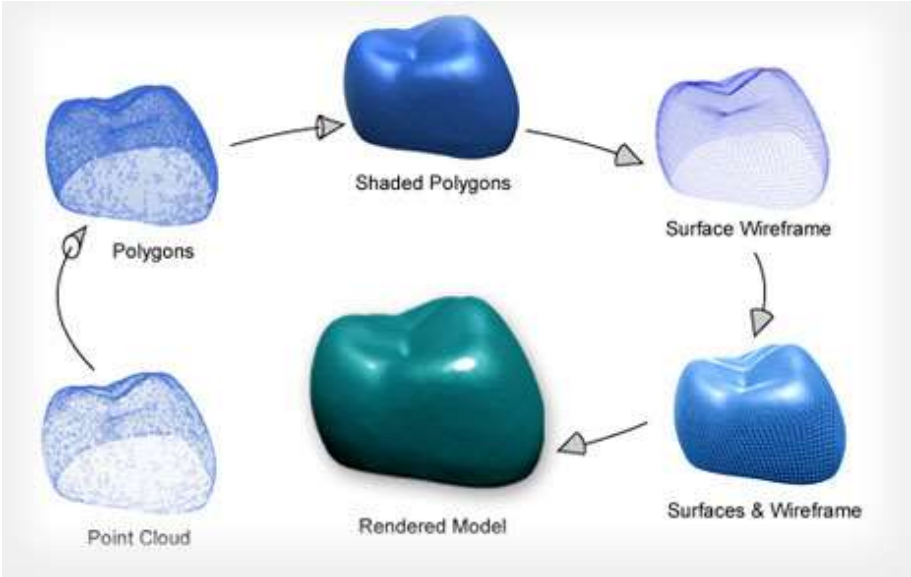
RE1: Servicii de imprimare 3D

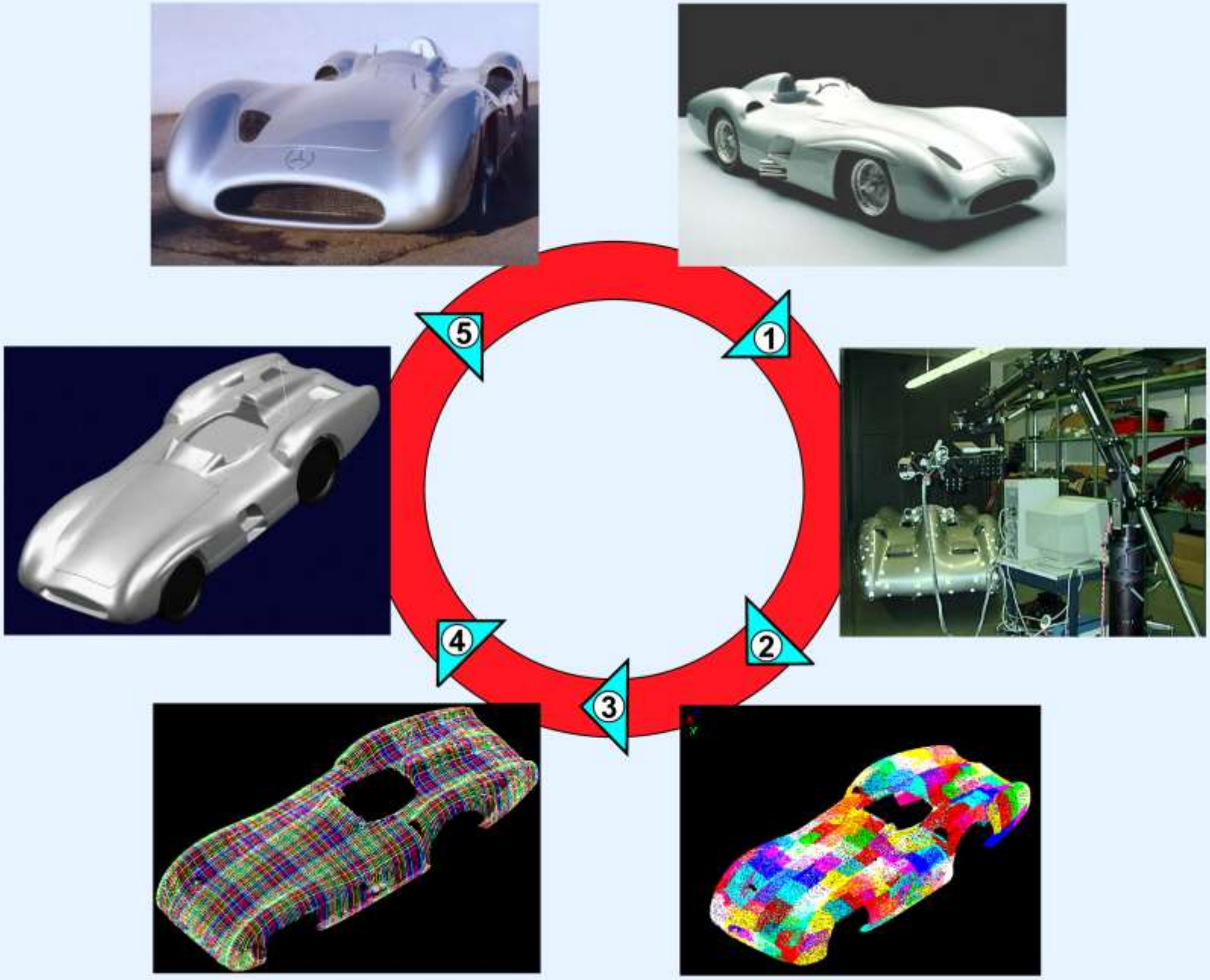
- Rapid prototyping

- FullCure materials

RE1: Domenii


Physical Part
Modeling Data


Polygons
Shaded Polygons
Surface Wireframe
Surfaces & Wireframe
Rendered Model
Point Cloud


1
2
3
4
5

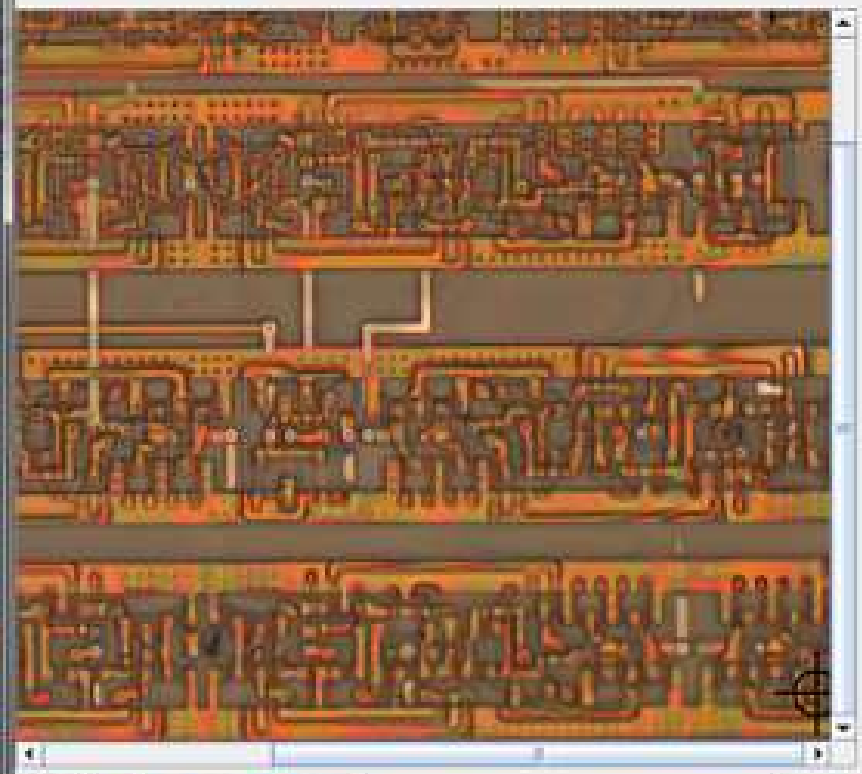
Reverse engineering of integrated circuits/smart cards

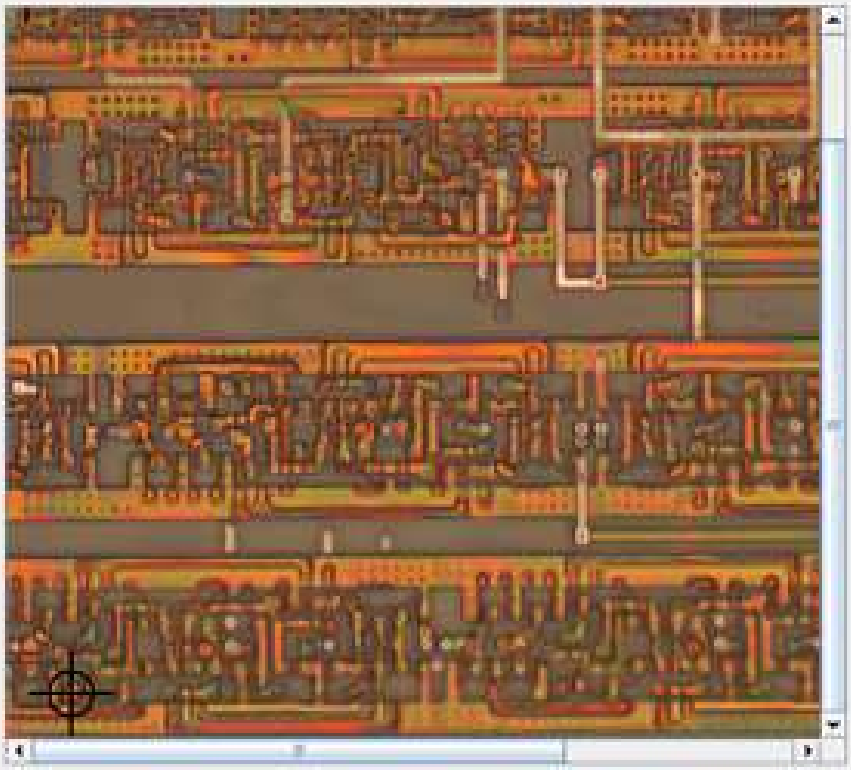
- RE is an invasive and destructive form of analyzing a smart card
- The attacker grinds away layer by layer of the smart card and takes pictures with an electron microscope
- Engineers employ sensors to detect and prevent this attack

Image Stitching Tool

3. Schritt: Setzen der Kontrollpunkte

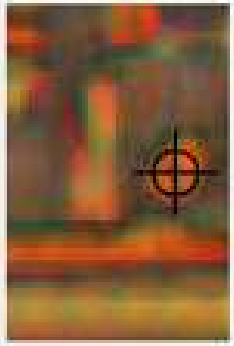
D:\... TFH DATEIEN ...\xx Praktika\CRE\aChipBilder\z...





654 - 63

680 - 484



12 - 64

38 - 486

Kontrollpunkt speichern + Weiter

degate -- [/home/martin/Development/degate3/2011-02-06_0045_degate*] [1/3]

Project View Tools Layer Logic Gate Recognition Help

Logic gates

Short Name	#	Width	Height	Fill color	Frame color	Description
01-FF	58	272	134			D-Q-FlipFlop
02-FF	11	343	134			D-Q-FlipFlop with rst
03-FF	17	343	133			D-Q-FlipFlop with rts
04-BUF	5	226	128			
05-3XOR	13	249	133			
06: 2-XNOR	12	133	133			

Edit Add Remove

Close

Edit gate

Entity Behaviour Layout

Short name: 02-FF

Description: D-Q-FlipFlop with rst

Logic Class: flipflop (generic)

Ports:

Port ID	Port Name	Port Description	In	Out
2384	!Q	!Q	☐	☑
2380	Q	Q	☐	☑
2388	clk	clk	☑	☐
2386	D	D	☑	☐
2382	rst	rst	☑	☐

Add Remove

Fill color: Reset Color

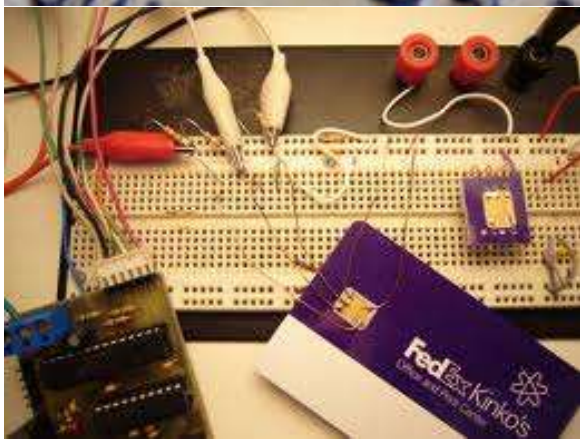
Frame color: Reset Color

Cancel OK

Autosaving project data ... done.

RE2: Smart cards

- Satellite TV
- Security card
- Phone card
- Ticket card
- Bank card



Reverse engineering for military applications

- Reverse engineering is often used by militaries in order to copy other nations' technologies, devices or information that have been obtained by regular troops in the fields or by intelligence operations

- It was often used during the Second World War and the Cold War

- Well-known examples from WWII and later include: rocket, missile, bombers, China has reversed many examples of US and Russian hardware, from fighter aircraft to missiles and HMMWV cars

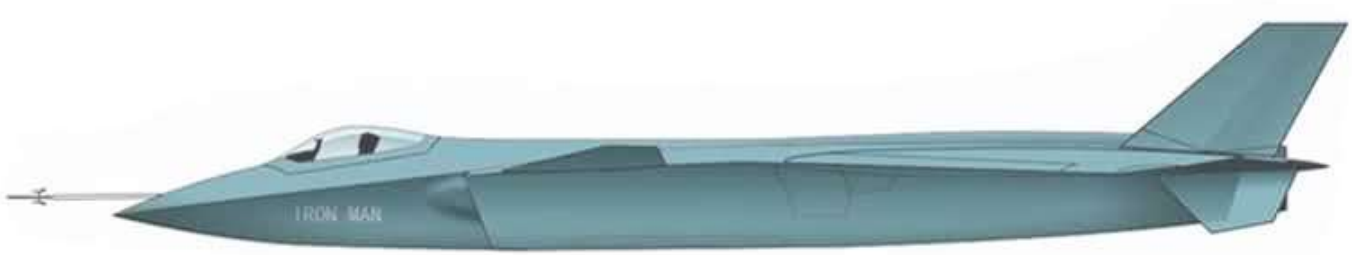
RE3: Avioane

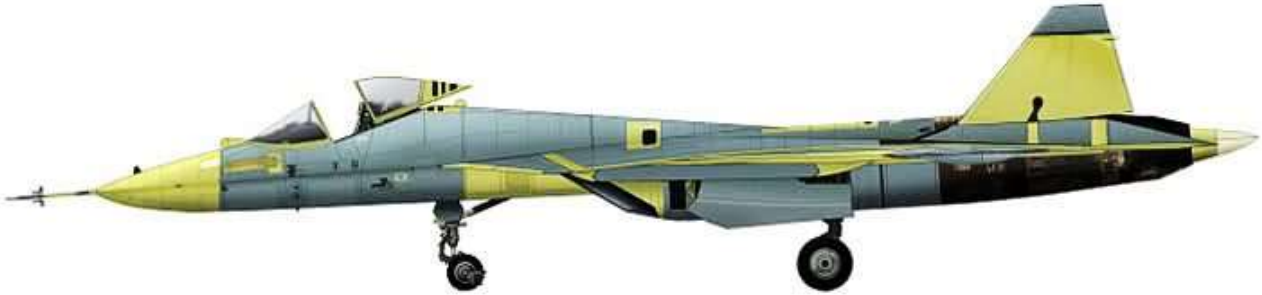
- US – B-29

URSS – Tupolev Tu-4

RE3: Avioane (2)

- Chinese J-20, Black Eagle US F-22, Russian Sukhoi T-50





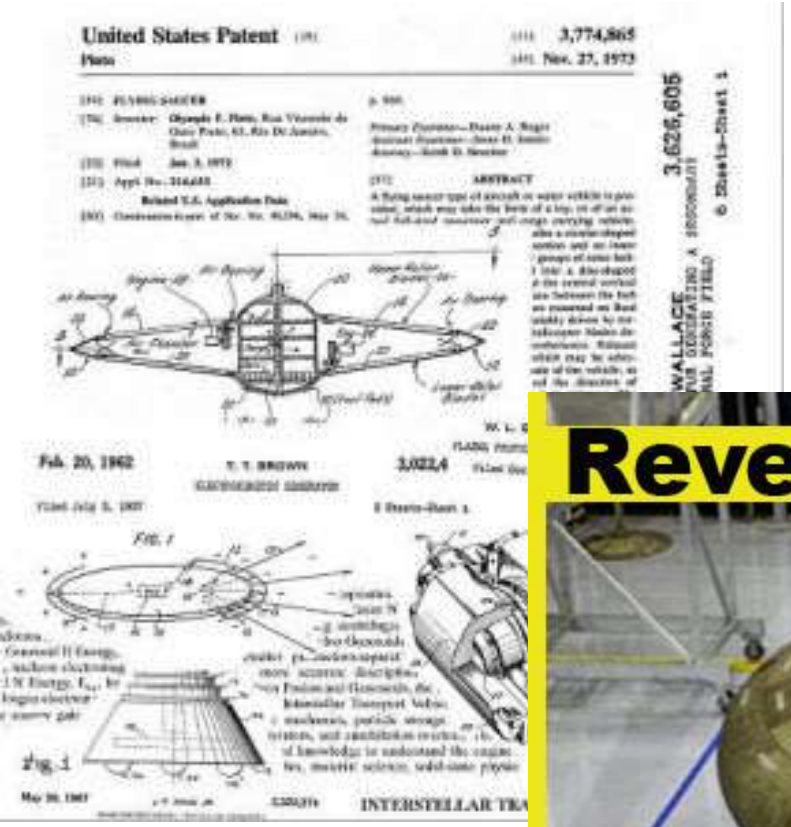
RE3: Rachete

- US –AIM-9 Sidewinder Soviet – Vympel K-13

RE3: Submar


Russia's new ballistic missile submarine
OLD
Typhoon class
16 x SLBM's
557 ft 9in
NEW
Borei class
20 x SLBM's
574 ft 2in
Yury Dolgoruky First Borei-class submarine is undergoing sea trials, two others are being built
Royal Navy
Vanguard class
16 x SLBM's
491 ft 10in

RE3: UFOs


United States Patent
3,774,865
Nov. 27, 1973
3,626,605
Feb. 20, 1962
T. T. BROWN
3,022,4
INTERSTELLAR TRA


Reverse Engineering a UFO
by Robert Klein

Reverse engineering of software

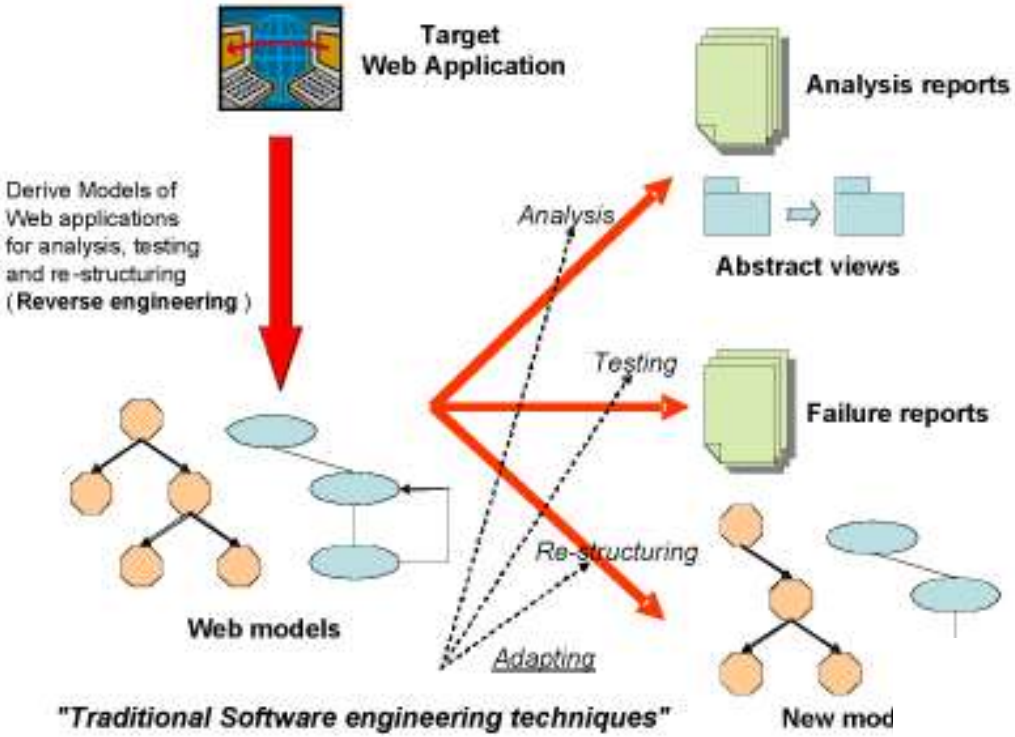
- Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction

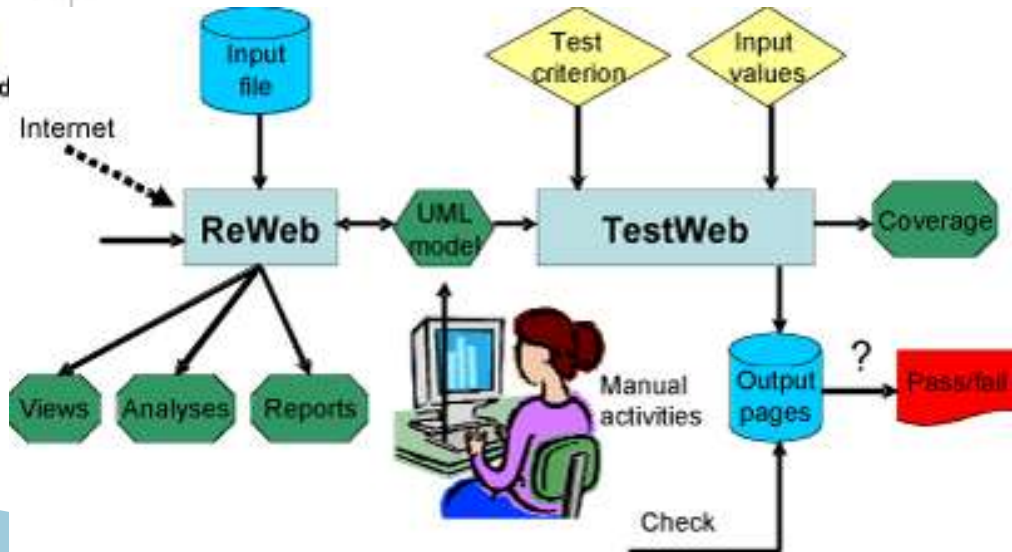
- In practice, two main types of RE emerge:
  - **Source code is available** (but it is poorly documented)
  - **There is no source code available for the software**

- **Black box testing** in software engineering has a lot in common with reverse engineering

RE4: Smart phones

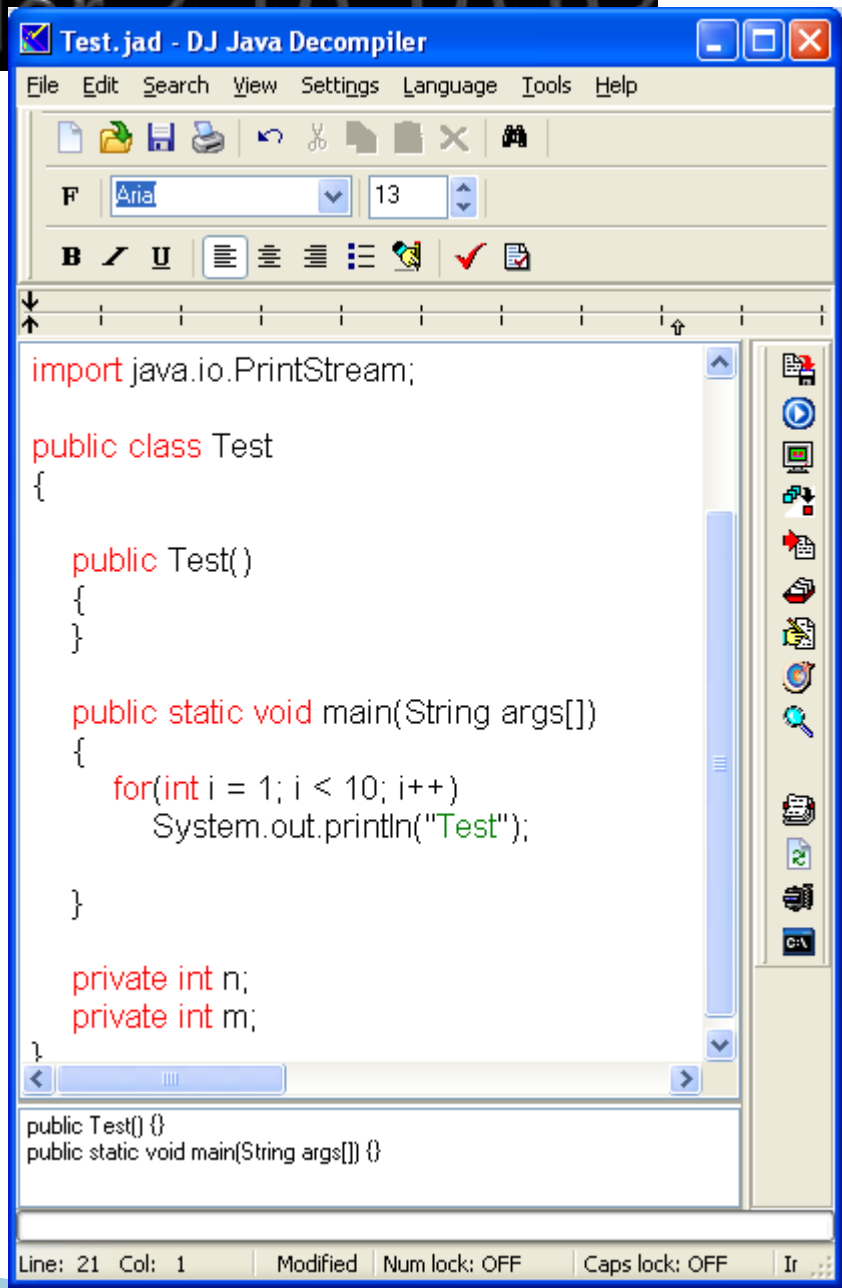
RE4 of Web Applications





RE4: DJ Java Decompiler 3.10.10.93

```
public class Test
{
   private int n;
   private int m;

   public static void main(String
   args[])
   {
         for(int i=1;i<10;i++)
         System.out.println("Test");
   }
}
```



RE4: JAD

- Link: http://www.steike.com/code/java-reverse-engineering/
- jad.exe NumeFisier.class => NumeFisier.jad

D:\Work\Ady\Scoala\2009-2010\JP\Cursuri\04\Reverse\Initial\Student.java - Notepad++

File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?

Student.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package test1;

/**
 *
 * @author Adrian
 */
public class Student extends Person{
    private int year;
    private int group;

    public Student(String name, String address,int year, int gro
    {
        super(name,address);
        this.group = group;
        this.year = year;
    }
}
```

371 chars 394 bytes 23 lines | Ln : 23 Col : 1 Sel : 0 (0 bytes) in 0 ranges | UNIX | ANSI

D:\Work\Ady\Scoala\2009-2010\JP\Cursuri\04\Reverse\Student.jad - Notepad++

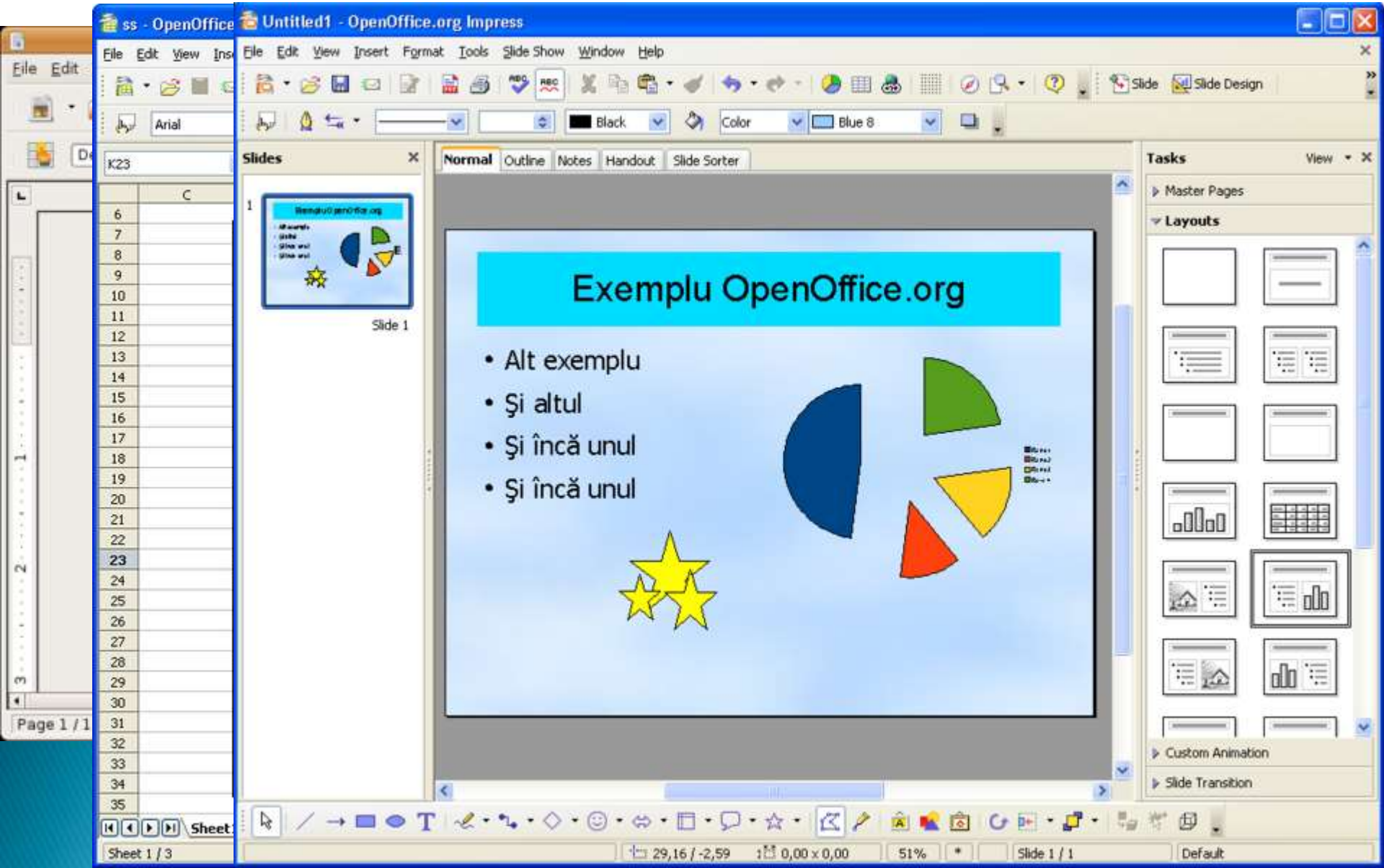
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?

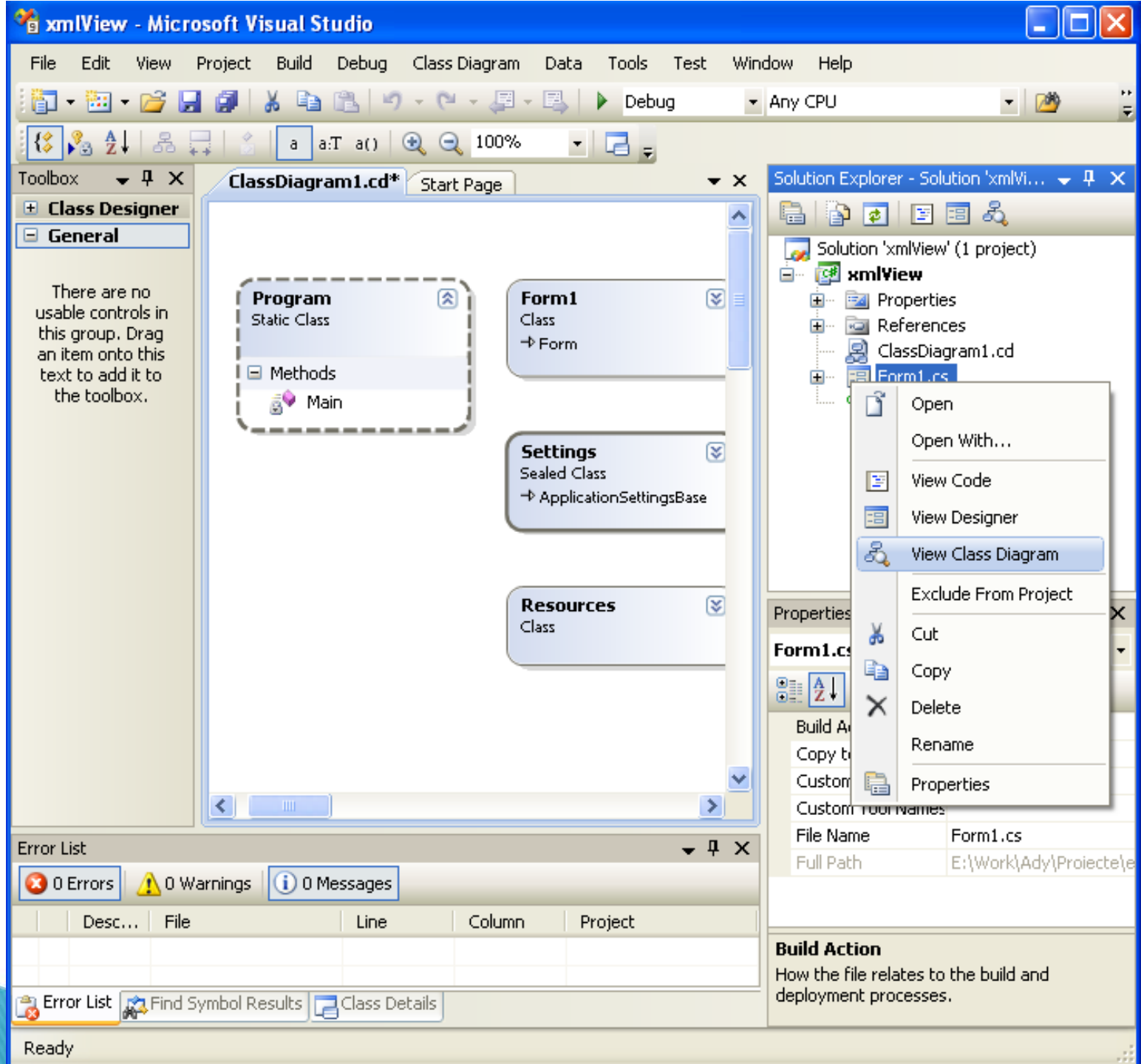
Student.jad

```java
// Decompiled by Jad v1.5.8f. Copyright 2001 Pavel Kouznetsov.
// Jad home page: http://www.kpdus.com/jad.html
// Decompiler options: packimports(3)
// Source File Name:   Student.java

package test1;


// Referenced classes of package test1:
//            Person

public class Student extends Person
{

    public Student(String name, String address, int year, int group)
    {
        super(name, address);
        this.group = group;
        this.year = year;
    }

    private int year;
    private int group;
}
```

Java : 492 chars 542 bytes 25 lines | Ln : 1 Col : 1 Sel : 0 (0 bytes) in 0 ranges | Dos\Windo | ANSI | INS

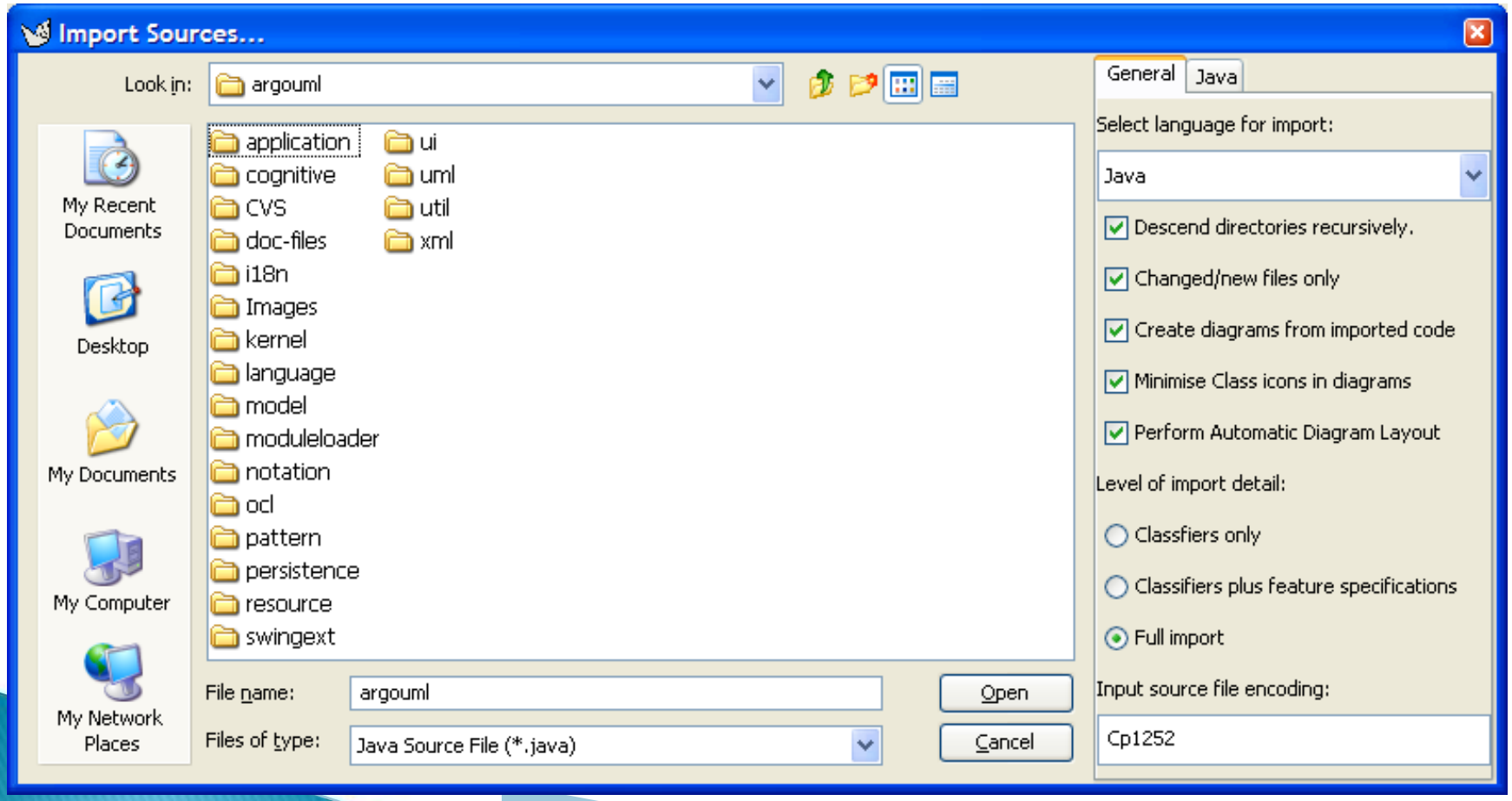
RE4: Open Office



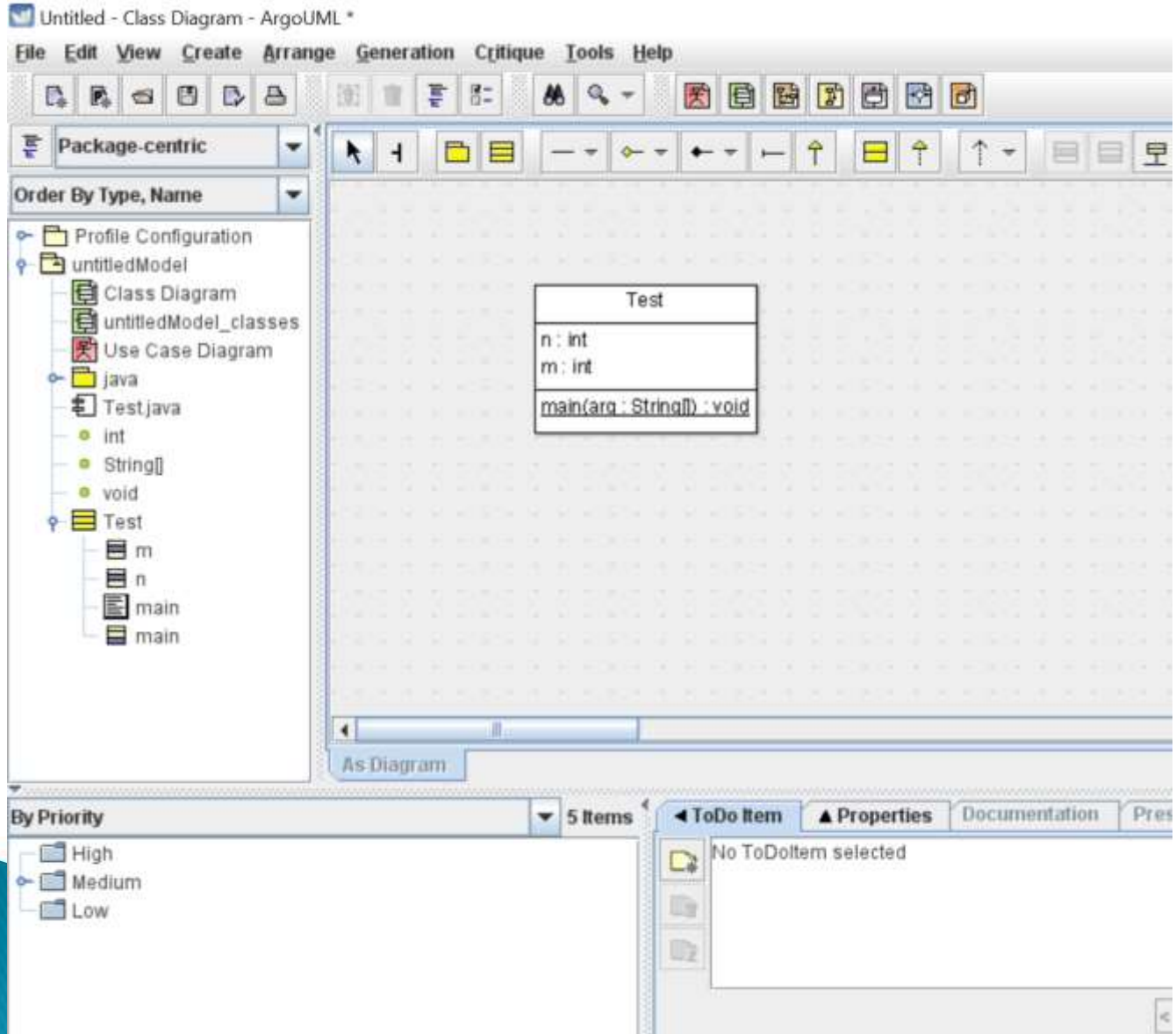

xmlView - Microsoft Visual Studio
File Edit View Project Build Debug Class Diagram Data Tools Test Window Help
Debug
Any CPU
100%
Toolbox
Class Designer
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
ClassDiagram1.cd*
Start Page
Program
Static Class
Methods
Main
Form1
Class
Form
Settings
Sealed Class
ApplicationSettingsBase
Resources
Class
Solution Explorer - Solution 'xmlVi...
Solution 'xmlView' (1 project)
xmlView
Properties
References
ClassDiagram1.cd
Form1.cs
Open
Open With...
View Code
View Designer
View Class Diagram
Exclude From Project
Cut
Copy
Delete
Rename
Properties
Form1.cs
File Name
Form1.cs
Full Path
Build Action
How the file relates to the build and deployment processes.
Error List
0 Errors
0 Warnings
0 Messages
Desc...
File
Line
Column
Project
Error List
Find Symbol Results
Class Details
Ready

RE în ArgoUML

- File –> **Import Sources...**



Pentru exemplul anterior...



Demo FE & RE

- Forward engineering:
  - Diagrame de clasă -> .java files (ArgoUML)
  - .java files -> .class files (NetBeans)

- Reverse engineering:
  - .class files -> .java files (JAD Decompiler)
  - .java files -> Diagrame de Clasă (ArgoUML)

Concluzii

- Diagrame UML:
  - Interacțiuni
  - Comportamentale
  - Structură

- C4 Model: context, container, component, code

Întrebări

- 1) Până unde trebuie să modelați un proiect la care lucrați? (un proiect cu 4-5 persoane)
- 2) Cu ce ar trebui începută modelarea unui proiect?
- 3) E etic ca o firmă (mare) să facă RE pe un produs de la concurență?
- 4) De ce nu trebuie încurajat RE? Care sunt efectele RE?

Bibliografie

- Ovidiu Gheorghieş, Curs 5 IP
- www.uml.org

- Reverse Engineering and Design Discovery: A Taxonomy, Chikofsky, E.J. and Cross, J., January, 1990

Links (RE)

- **DJ Java Decompiler 3.10.10.93:** http://www.softpedia.com/progDownload/DJ-Java-Decompiler-Download-13481.html
- **Open Office**: http://ro.wikipedia.org/wiki/OpenOffice.org
- **UML Reverse Engineering for Existing Java, C# , and Visual Basic .NET Code:** **http://www.altova.com/umodel/uml-reverse-engineering.html**
- **Reverse Engineering**: http://en.wikipedia.org/wiki/Reverse_engineering
- **PROTO 3000 3D Engineering Solutions:** http://www.proto3000.com/services.aspx
- HAR2009: http://www.degate.org/HAR2009/
- Degate: http://www.degate.org/screenshots/
- Inteligent: http://www.intelligentrd.com/
- Smartphones RE: http://www.cytraxsolutions.com/2011/01/smartphones-security-and-reverse.html