

# Ingineria Programării

Cursul 5 – 20 Martie 2017

[adiftene@info.uaic.ro](mailto:adiftene@info.uaic.ro)

# Cuprins

- ▶ Din Cursurile trecute...
- ▶ Forward and Reverse Engineering
- ▶ GRASP
  - Information Expert
  - Creator
  - Low coupling
  - High cohesion
  - Controller

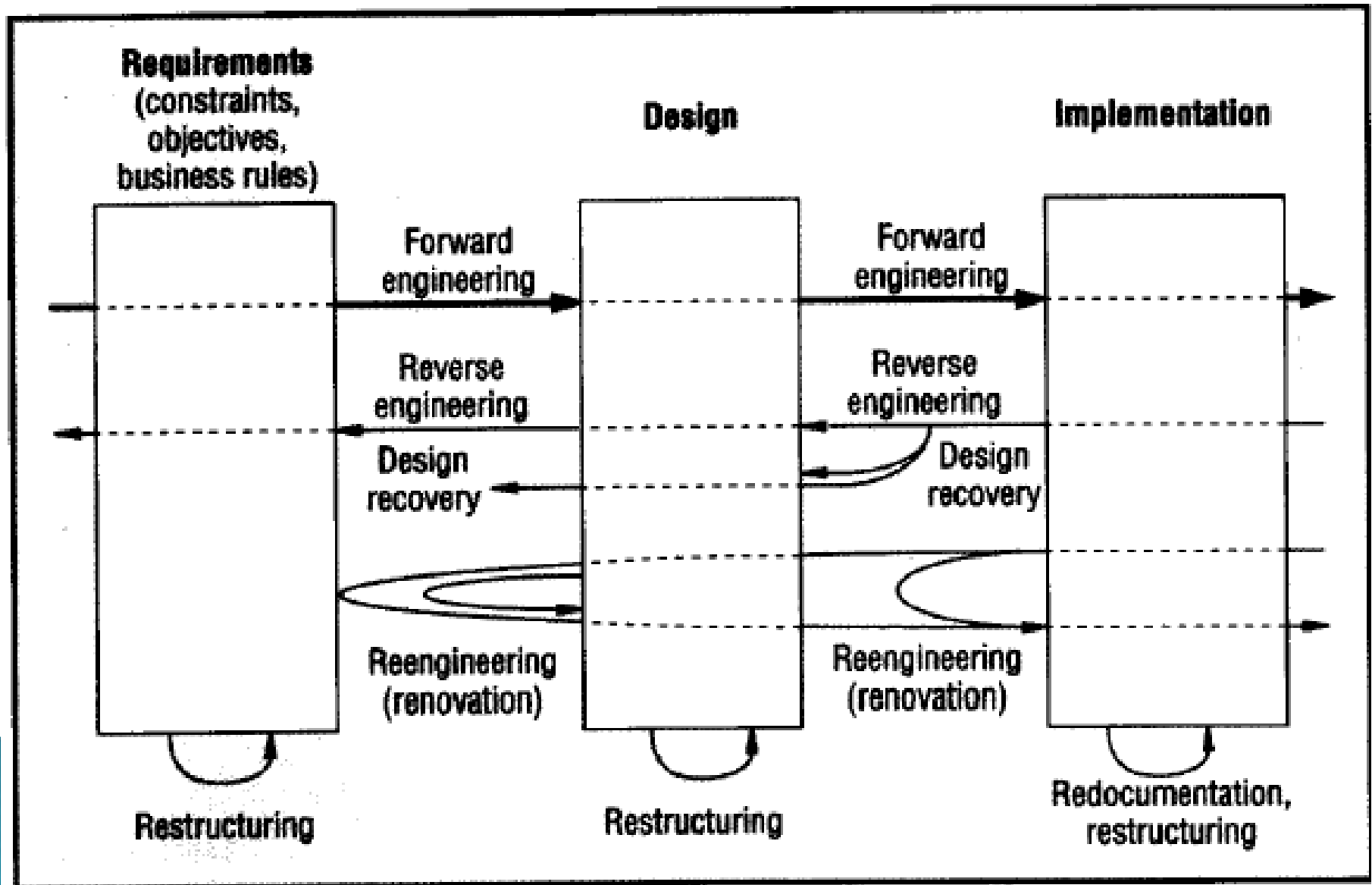
# Atenție

- ▶ Săptămâna 7-a e termenul limită pentru alegerea proiectului
- ▶ După care urmează: documentare, înțelegere, knowledge transfer, diagrame use case, diagrame de clasă, implementare, unit testing, etc.
- ▶ Săptămâna a 7-a începe efectiv lucrul la proiect, iar evaluarea se încheie în săptămâna a 14-a
- ▶ În săptămâna a 8-a *nu se fac ore...*

# RE

- ▶ De ce avem nevoie de modelare?
- ▶ Cum putem modela un proiect?
- ▶ SCRUM – roles, values, artifacts, events, rules

# 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

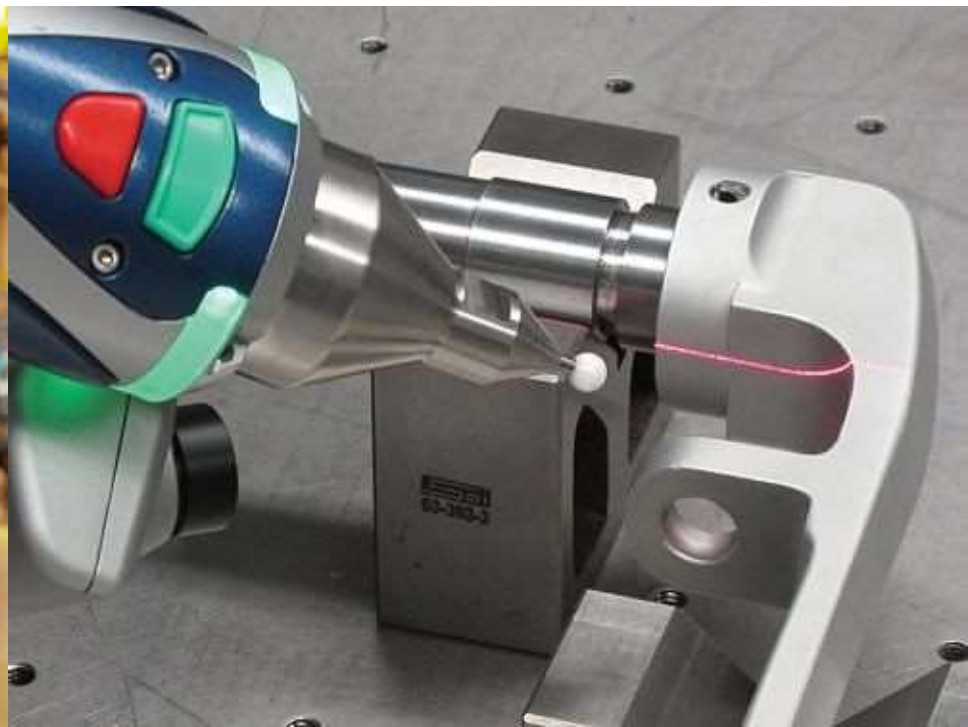
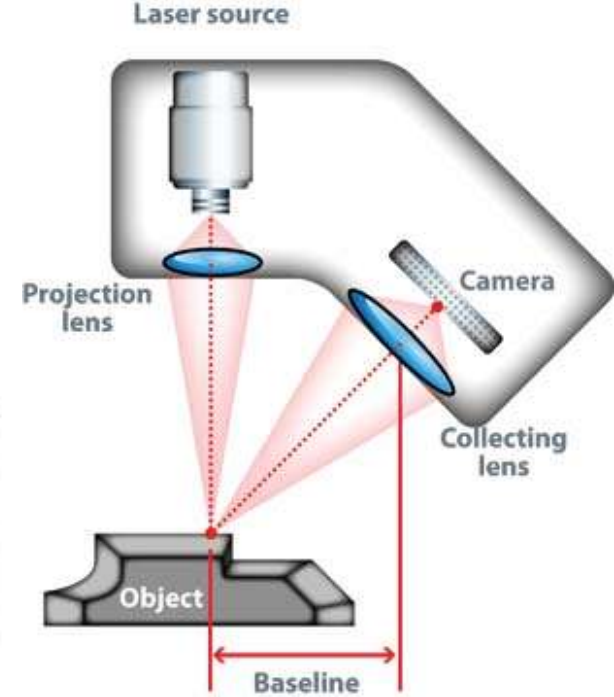
- ▶ Interoperability
- ▶ Lost documentation
- ▶ Product analysis
- ▶ Security auditing
- ▶ Removal of copy protection, circumvention of access restrictions
- ▶ Creation of unlicensed/unapproved duplicates
- ▶ Academic/learning purposes
- ▶ Curiosity
- ▶ Competitive technical intelligence (understand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



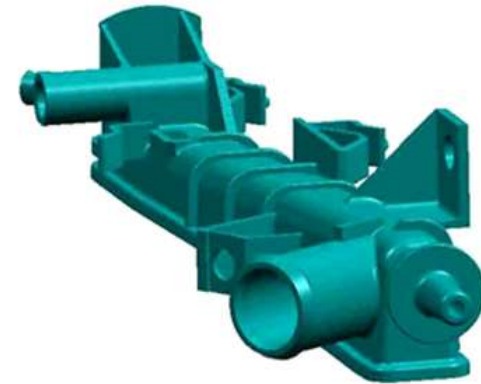
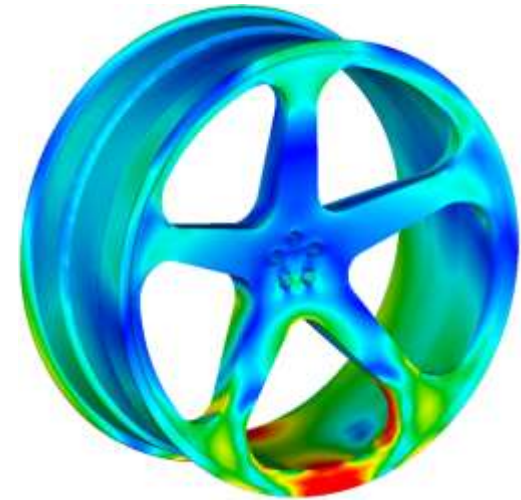
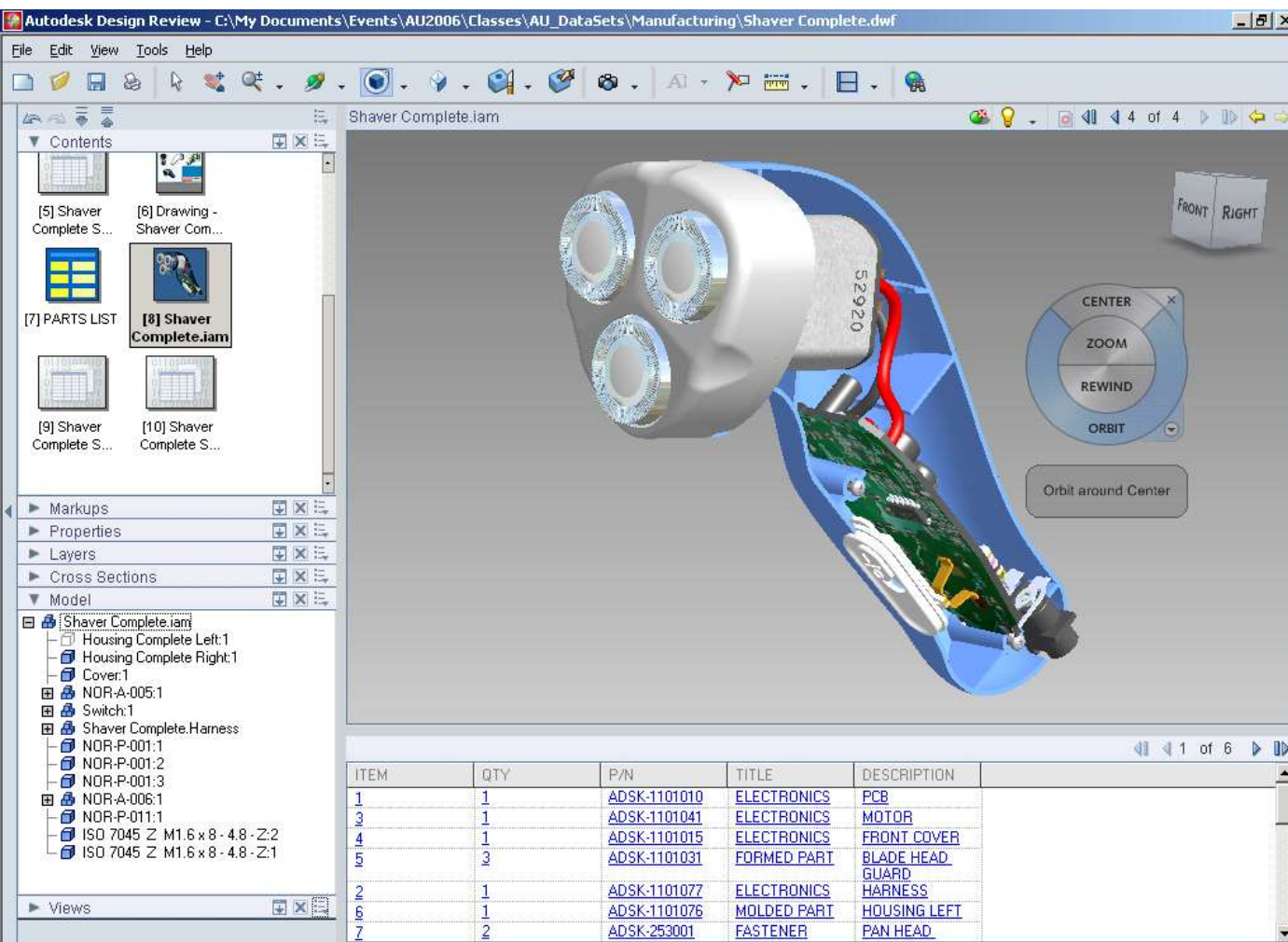
# Types of RE

- ▶ 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

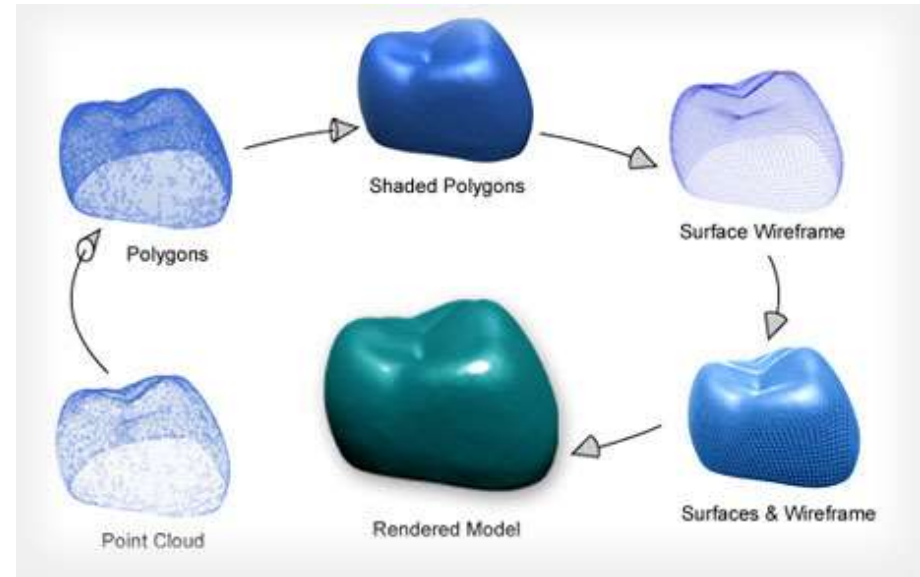


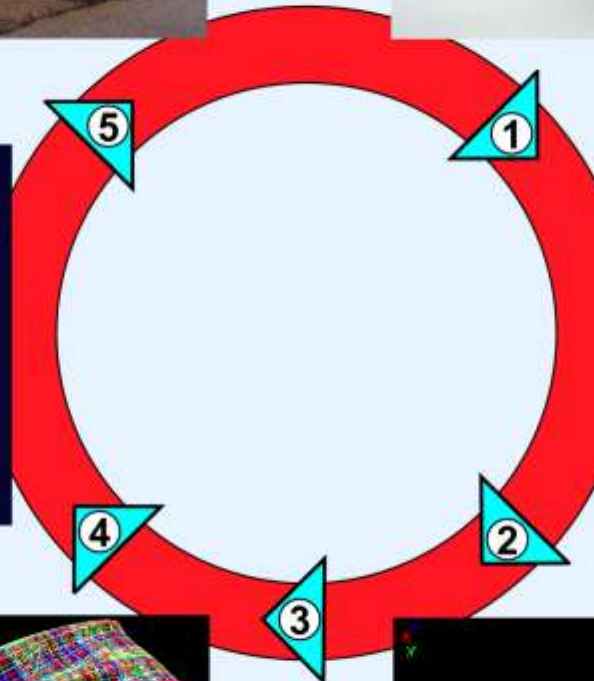
- ▶ FullCure materials



# RE1: Domenii

Physical Part	Modeling Data
	
	
	





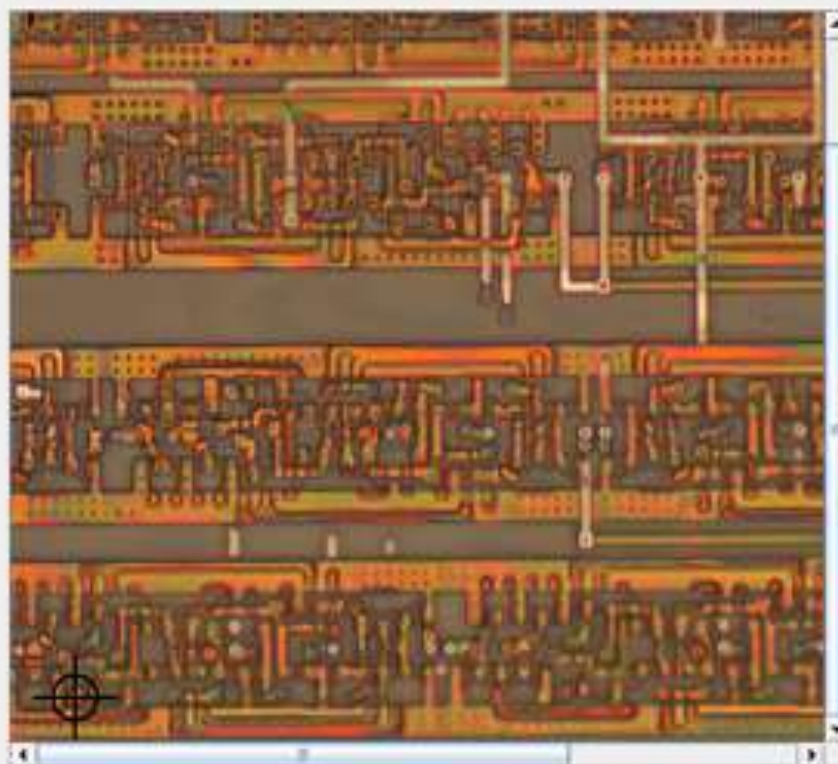
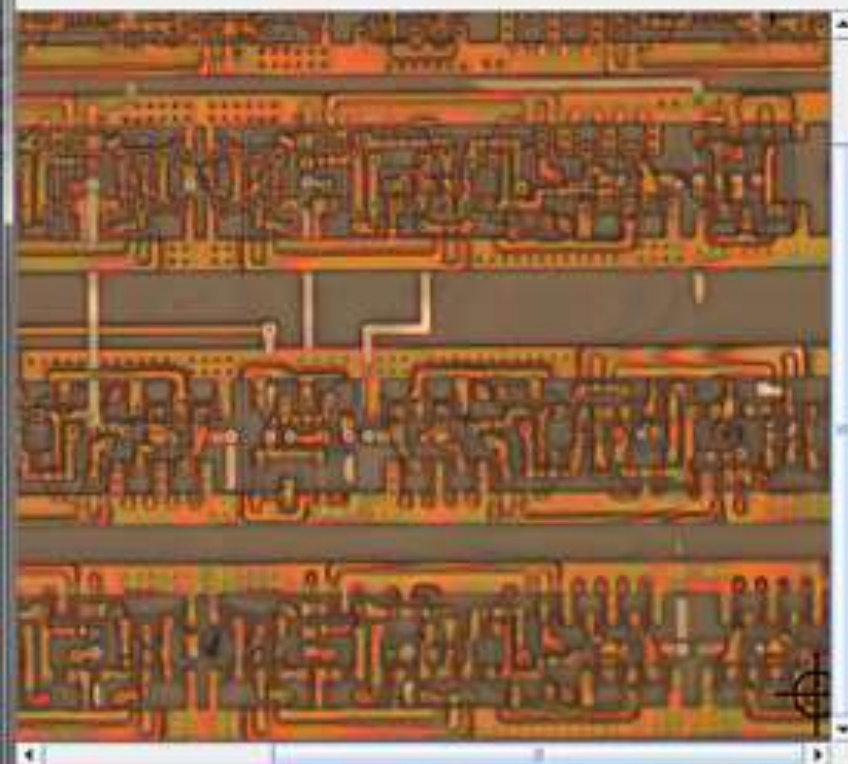
# Reverse engineering of integrated circuits/smart cards

- ▶ 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



### 3. Schritt: Setzen der Kontrollpunkte

☐ D:\TFH DATEIEN \---\praktika\CREla\ChipBilder\iz...



654 - 63

580 - 484



12 - 64

38 - 486





The screenshot displays the degate software interface. The background is a grayscale circuit layout with various logic gates and interconnects. A ruler at the top shows coordinates from 5300 to 6600. Two dialog boxes are open in the foreground.

**Logic gates dialog:**

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			

Buttons: Edit, Add, Remove, Close

**Edit gate dialog:**

Entity Behaviour Layout

Short name: 02-FF  
Description: D-Q-FlipFlop with rst  
Logic Class: flipflop (generic)

Port ID	Port Name	Port Description	In	Out
2384	!Q	!Q	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2380	Q	Q	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2388	clk	clk	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2386	D	D	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2382	rst	rst	<input checked="" type="checkbox"/>	<input type="checkbox"/>

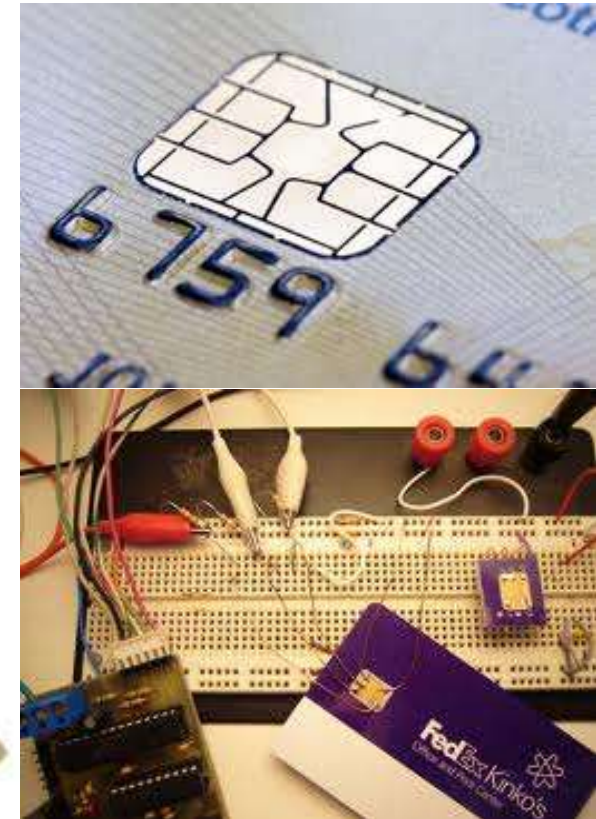
Buttons: Add, Remove

Fill color: [Color Picker] Reset Color  
Frame color: [Color Picker] Reset Color

Buttons: Cancel, OK

# 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

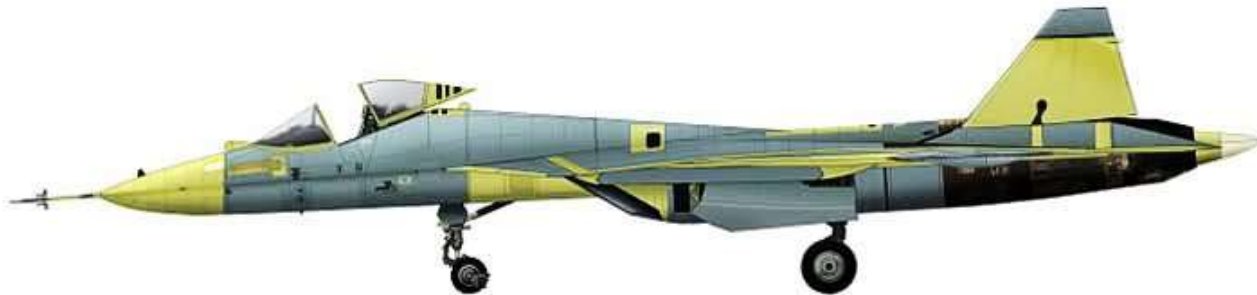
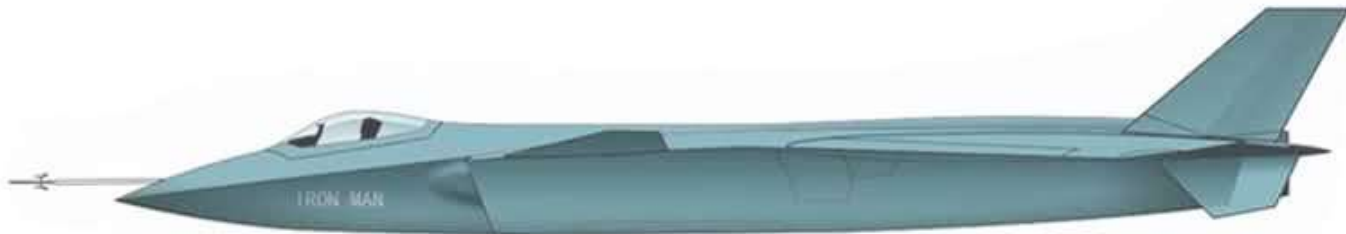


URSS – Tupolev Tu-4



# 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



# Russia's new ballistic missile submarine

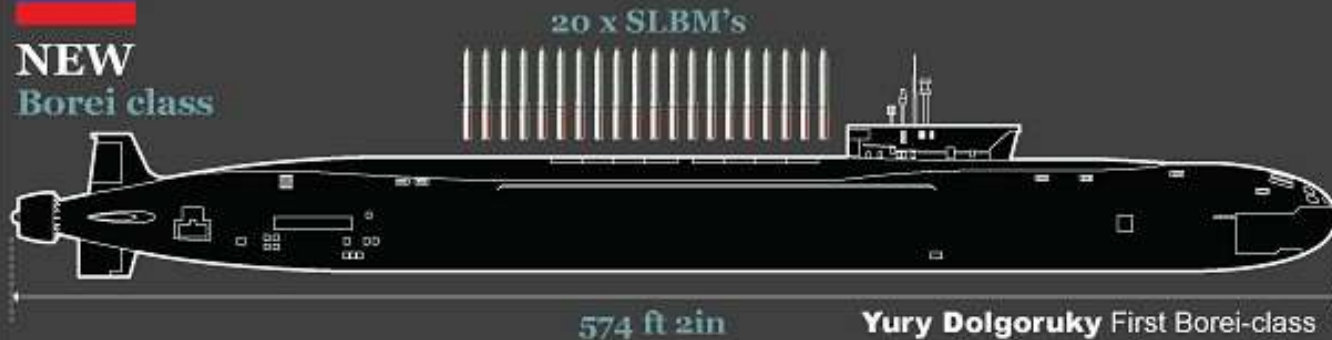
RE3: Submar



**OLD**  
Typhoon class



**NEW**  
Borei class



**Yury Dolgoruky** First Borei-class submarine is undergoing sea trials, two others are being built



**Royal Navy**  
Vanguard class





# RE3: UFOs

United States Patent (19) 3,774,865  
Patent

(11) 3,774,865  
(12) Nov. 27, 1973

(21) 3,774,865

(22) Inventor: Olynth E. Platt, Rue Vienne de  
Olynth E. Platt, Rue Vienne de  
Olynth E. Platt, Rue Vienne de

(23) Filed Jan. 3, 1972

(24) Appl. No. 3,774,865

Related U.S. Application Data

(30) Continuation of Ser. No. 8,124, Mar. 26,

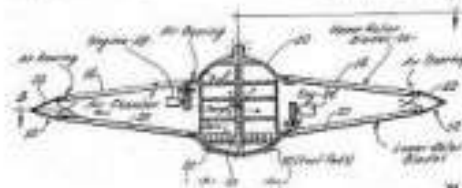
p. 905

Primary Examiner—Dante A. Ragni  
Assistant Examiner—James H. Smith  
Attorney—Ruth D. Newman

(57) ABSTRACT

A flying saucer type of aircraft or water vehicle is provided, which may take the form of a top or of an actual hull and may carry out wings covering sections also a water-tight section and an inner group of sections back 1 into a disc-shaped or the central vertical are between the hull are provided on the water-tight section by the water-tight section. The hull which may be adapted to the vehicle, as and the direction of

3,774,865  
WALLACE  
FOR ORIGINATING A SECONDARY  
WALLACE FIELD  
© Sheets-Sheet 1



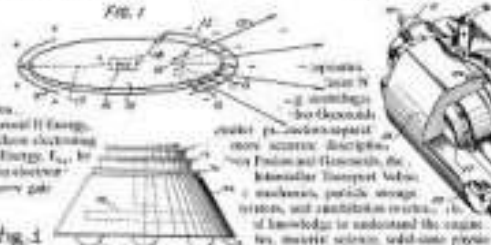
Feb. 20, 1962

T. T. BRIDGE  
GARDEN CITY, N.Y.

3,022,4

Filed July 5, 1961

2 Sheets-Sheet 1



May 26, 1967

3,022,4

INTERSTELLAR TRAVEL

## Reverse Engineering a UFO



by Robert Klein



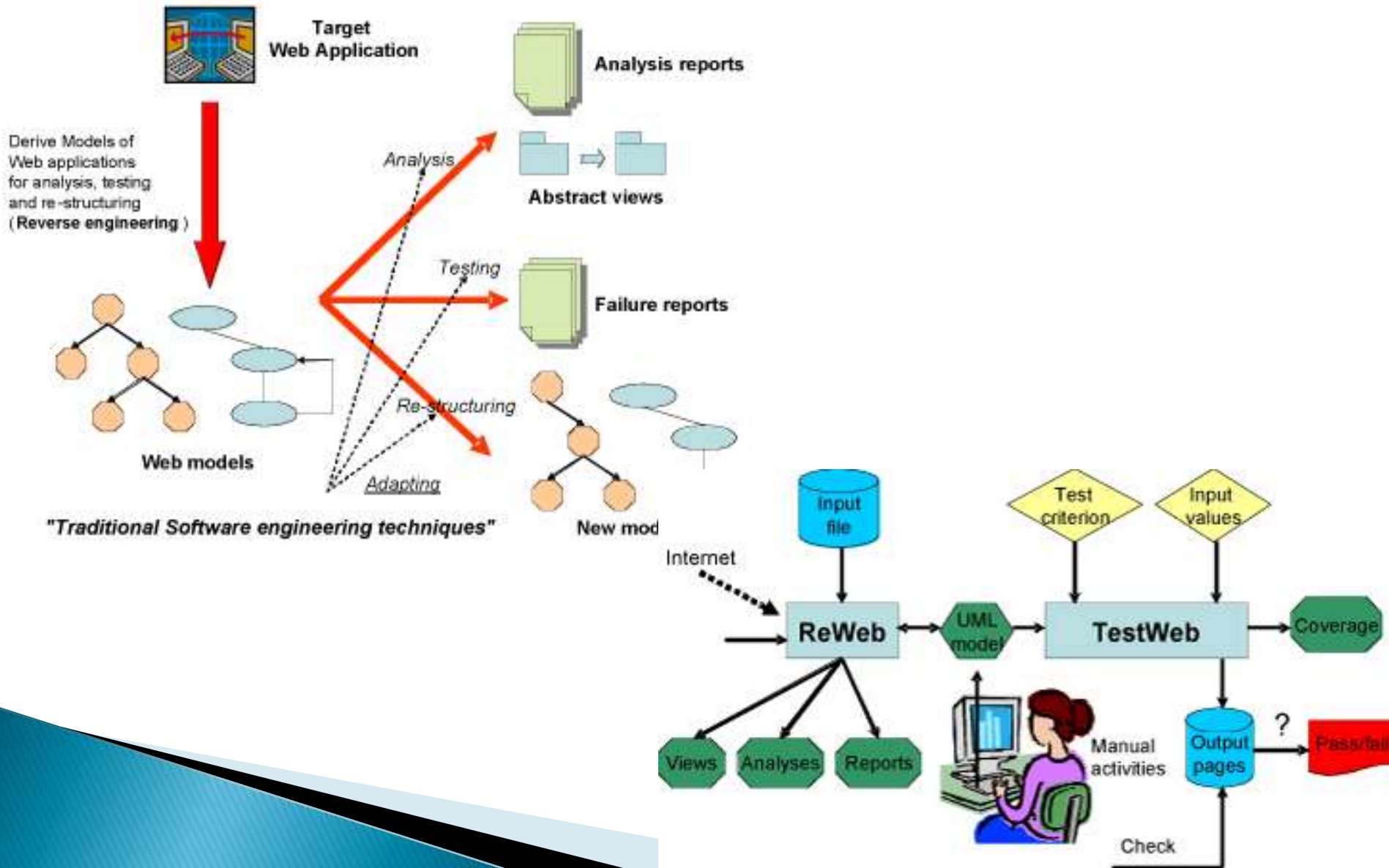
# Reverse engineering of software

- ▶ 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



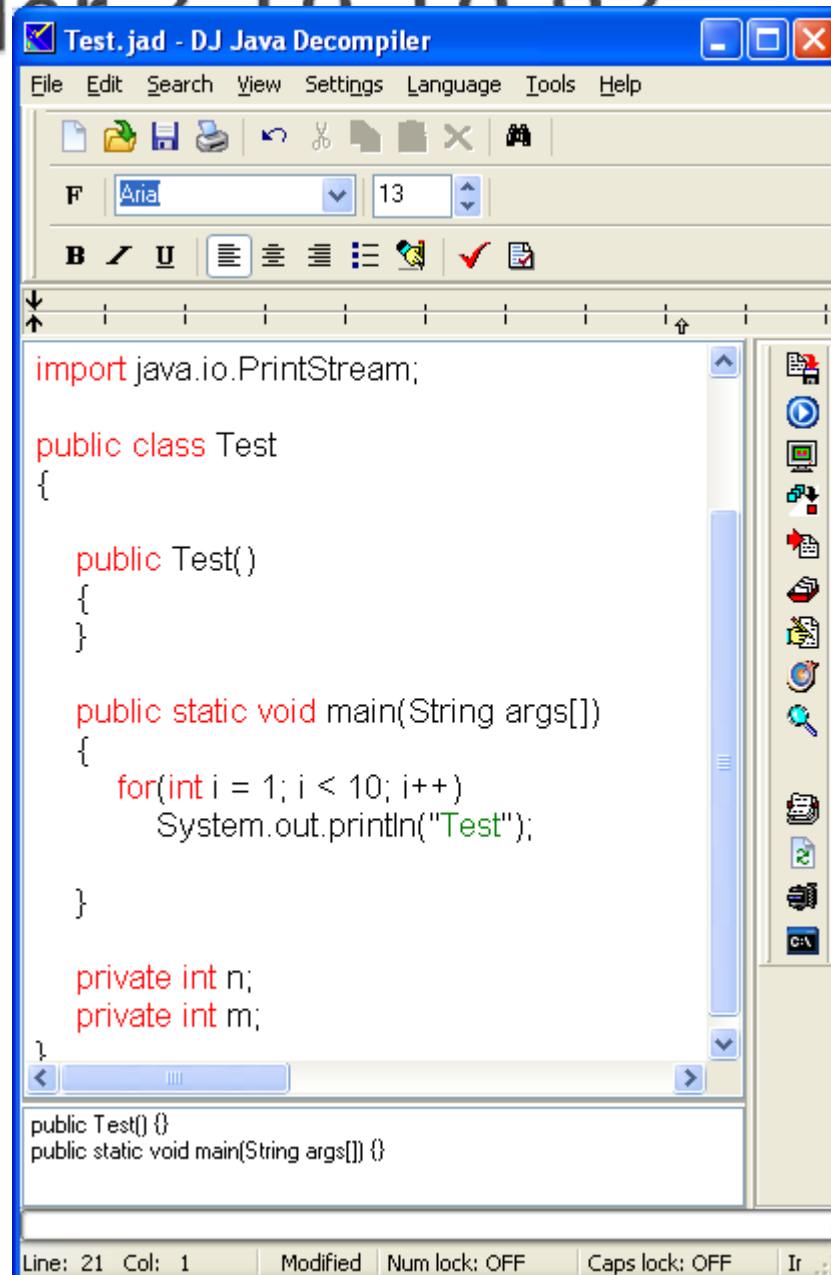
# RE4 of Web Applications



# RE4: DJ Java Decompiler 2.10.10.02

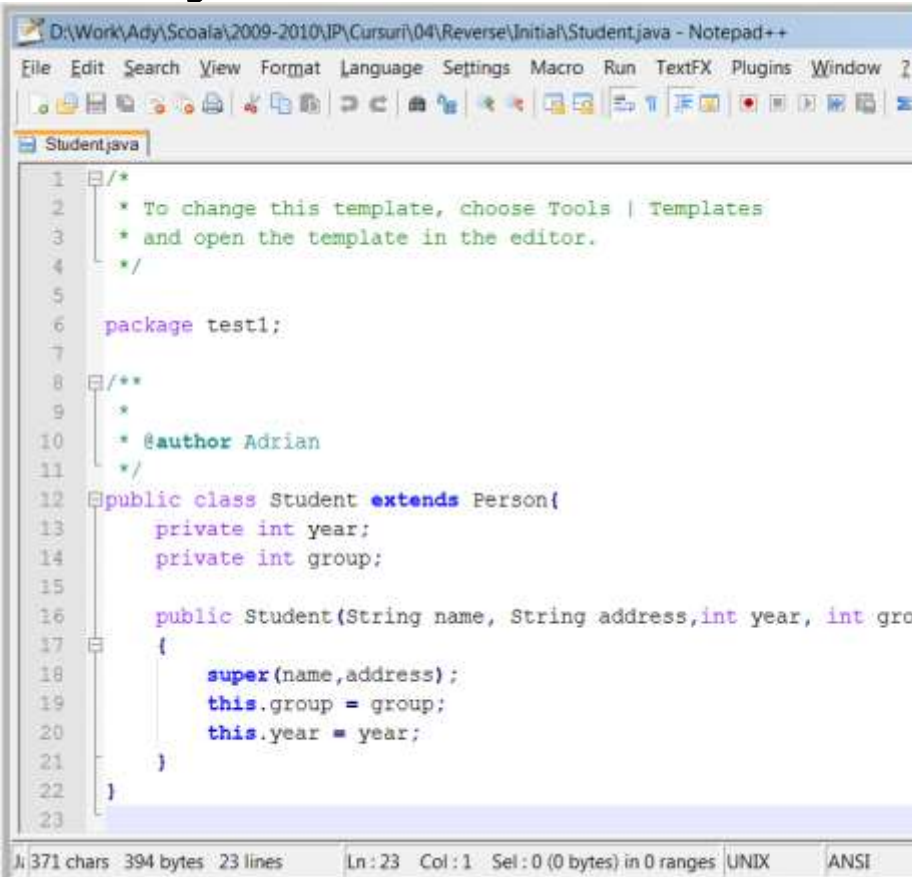
```
public class Test
{
    private int n;
    private int m;

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



# RE4: JAD

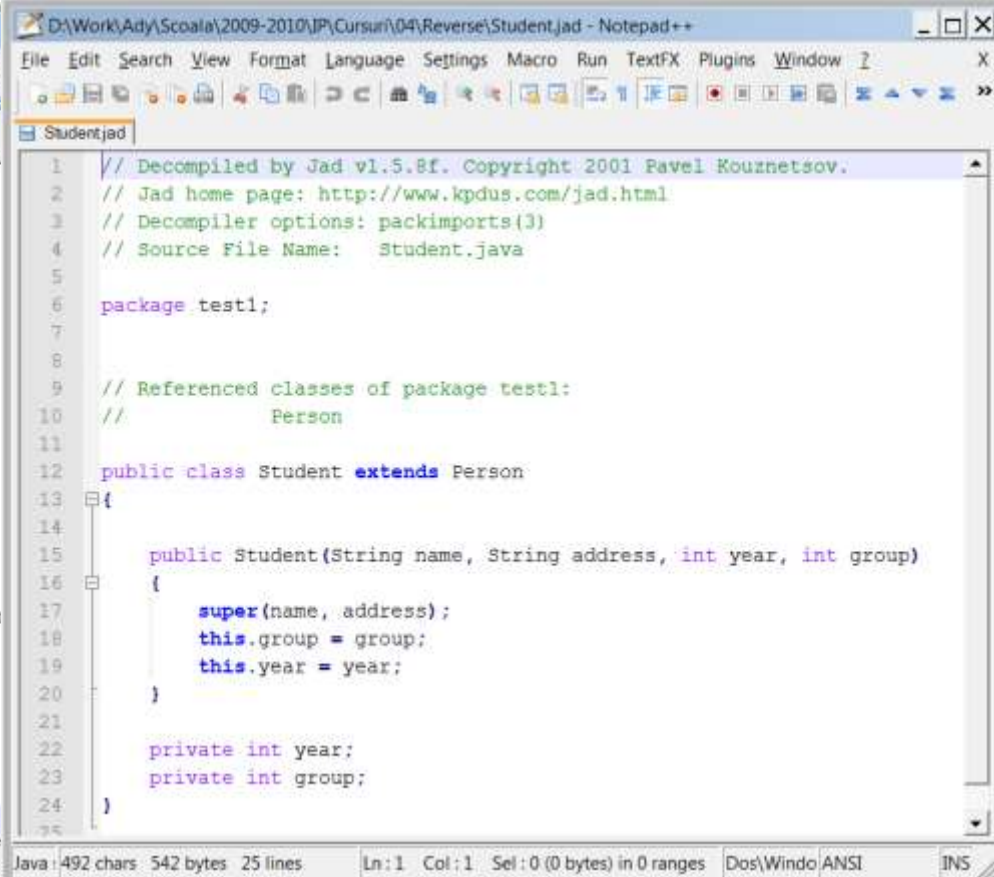
- ▶ Link: <http://www.steike.com/code/java-reverse-engineering/>
- ▶ `jad.exe NumeFisier.class => NumeFisier.jad`



The screenshot shows a Notepad++ window titled "D:\Work\Ady\Scoala\2009-2010\IP\Cursuri\04\Reverse\Initial\Student.java - Notepad++". The file "Student.java" contains the following code:

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package test1;
7
8  /**
9   *
10  * @author Adrian
11  */
12  public class Student extends Person{
13      private int year;
14      private int group;
15
16      public Student(String name, String address,int year, int group)
17      {
18          super(name,address);
19          this.group = group;
20          this.year = year;
21      }
22  }
```

The status bar at the bottom indicates: 371 chars 394 bytes 23 lines Ln: 23 Col: 1 Sel: 0 (0 bytes) in 0 ranges UNIX ANSI.



The screenshot shows a Notepad++ window titled "D:\Work\Ady\Scoala\2009-2010\IP\Cursuri\04\Reverse\Student.jad - Notepad++". The file "Student.jad" contains the following decompiled code:

```
1  // Decompiled by Jad vl.5.8f. Copyright 2001 Pavel Kouznetsov.
2  // Jad home page: http://www.kpdus.com/jad.html
3  // Decompiler options: packimports(3)
4  // Source File Name:   Student.java
5
6  package test1;
7
8
9  // Referenced classes of package test1:
10 //     Person
11
12  public class Student extends Person
13  {
14
15      public Student(String name, String address, int year, int group)
16      {
17          super(name, address);
18          this.group = group;
19          this.year = year;
20      }
21
22      private int year;
23      private int group;
24  }
```

The status bar at the bottom indicates: Java 492 chars 542 bytes 25 lines Ln: 1 Col: 1 Sel: 0 (0 bytes) in 0 ranges Dos\Windows ANSI.



# RE4: Open Office

ss - OpenOffice.org Impress

File Edit View Insert Format Tools Slide Show Window Help

Normal Outline Notes Handout Slide Sorter

Slide 1

## Exemplu OpenOffice.org

- Alt exemplu
- Și altul
- Și încă unul
- Și încă unul

Tasks

Master Pages

Layouts

Custom Animation

Slide Transition

Page 1 / 1

Sheet 1 / 3

29,16 / -2,59 0,00 x 0,00 51% Slide 1 / 1 Default

# C#

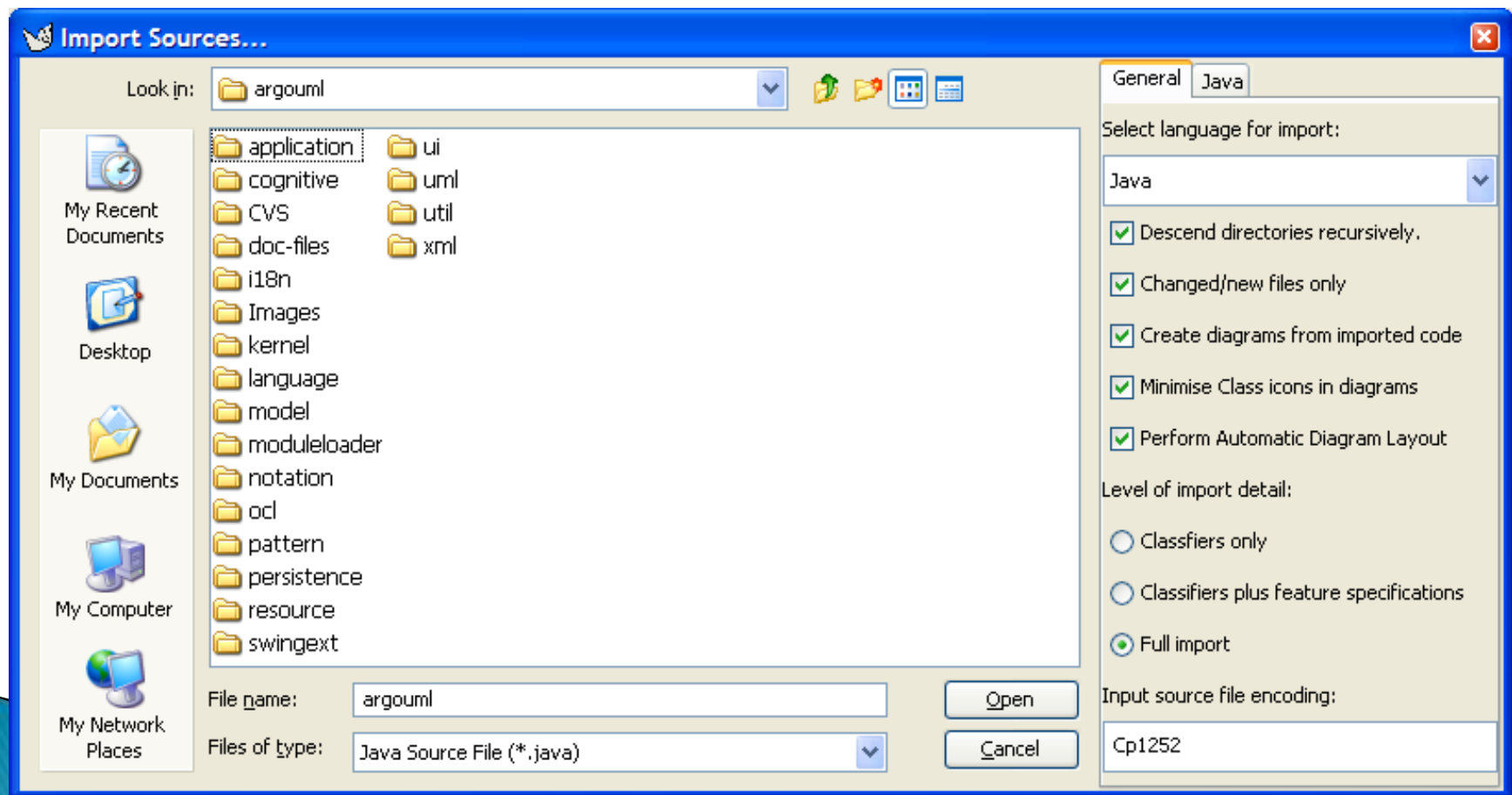
The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Class Designer (ClassDiagram1.cd\*):** Shows a class hierarchy with four classes: **Program** (Static Class) containing a **Main** method; **Form1** (Class) inheriting from **Form**; **Settings** (Sealed Class) inheriting from **ApplicationSettingsBase**; and **Resources** (Class).
- Solution Explorer:** Shows the project structure for 'xmlView' (1 project), including **Properties**, **References**, **ClassDiagram1.cd**, and **Form1.cs**. A context menu is open over **Form1.cs**, listing actions such as **Open**, **Open With...**, **View Code**, **View Designer**, **View Class Diagram** (highlighted), **Exclude From Project**, **Cut**, **Copy**, **Delete**, **Rename**, and **Properties**.
- Error List:** Shows 0 Errors, 0 Warnings, and 0 Messages.
- Build Action:** A section at the bottom right explaining how the file relates to the build and deployment processes.

Ready

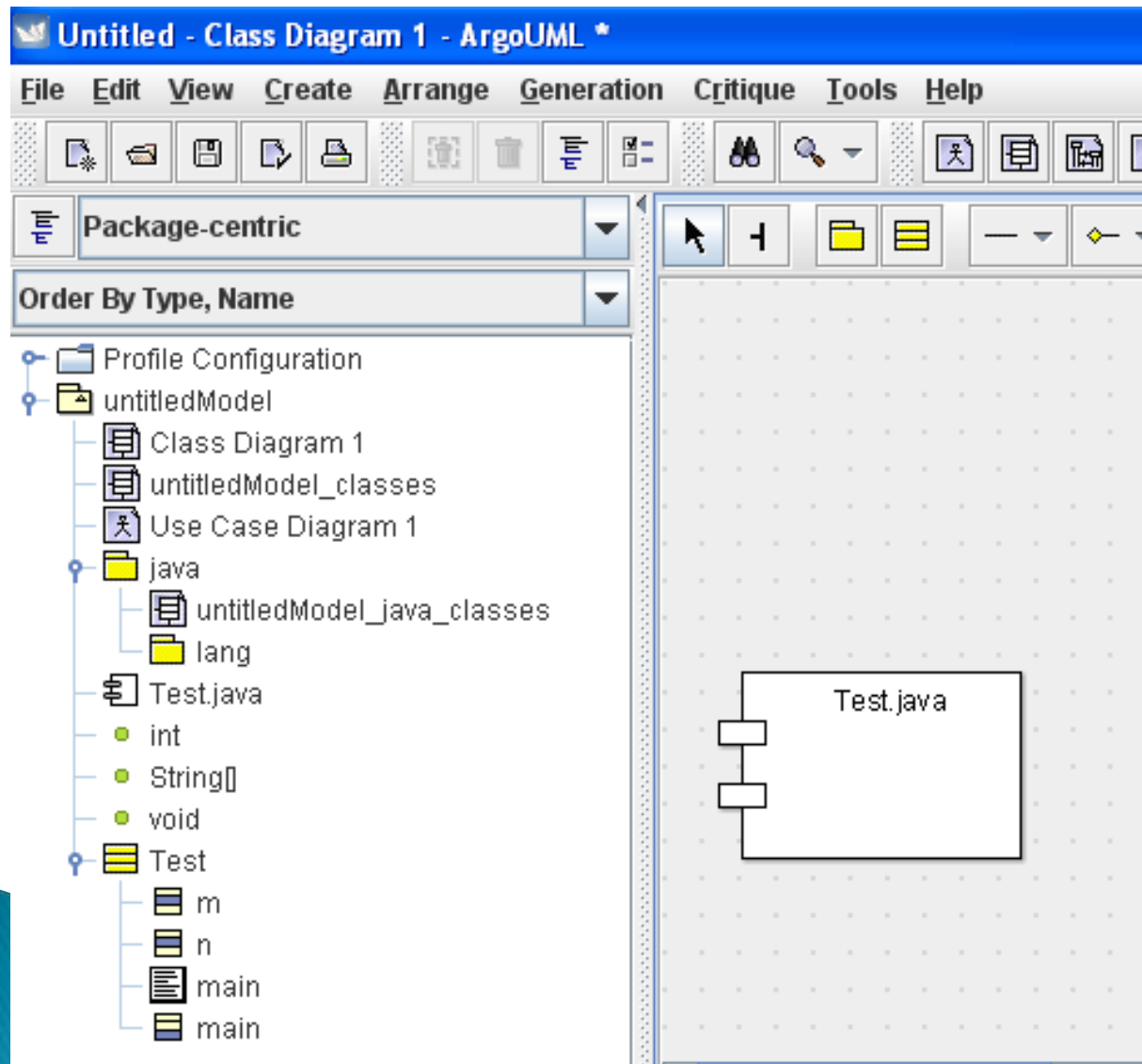
# RE în ArgoUML

## ► File -> Import Sources...





# Pentru exemplul anterior...



# GRASP

- ▶ GRASP = General Responsibility Assignment Software Patterns (Principles)
- ▶ Descrise de Craig Larman în cartea *Applying UML and Patterns. An Introduction to Object Oriented Analysis and Design*
- ▶ Ne ajută să alocăm responsabilități claselor și obiectelor în cel mai elegant mod posibil
- ▶ Exemple de principii folosite în GRASP: *Information Expert* (sau *Expert*), *Creator*, *High Cohesion*, *Low Coupling*, *Controller*, *Polymorphism*, *Pure Fabrication*, *Indirection*, *Protected Variations*

# Ce responsabilități?

## ▶ Să facă:

- Să facă ceva el însuși, precum crearea unui obiect sau să facă un calcul
- Inițializarea unei acțiuni în alte obiecte
- Controlarea și coordonarea activităților altor obiecte

## ▶ Să cunoască:

- Atributele private
- Obiectele proprii
- Lucrurile pe care le poate face sau le poate apela

# Pattern

- ▶ Traducere: șablon, model
- ▶ Este o soluție generală la o problemă comună
- ▶ Fiecare pattern are un nume sugestiv și ușor de reținut (ex. composite, observer, iterator, singleton, etc.)

# Information Expert 1

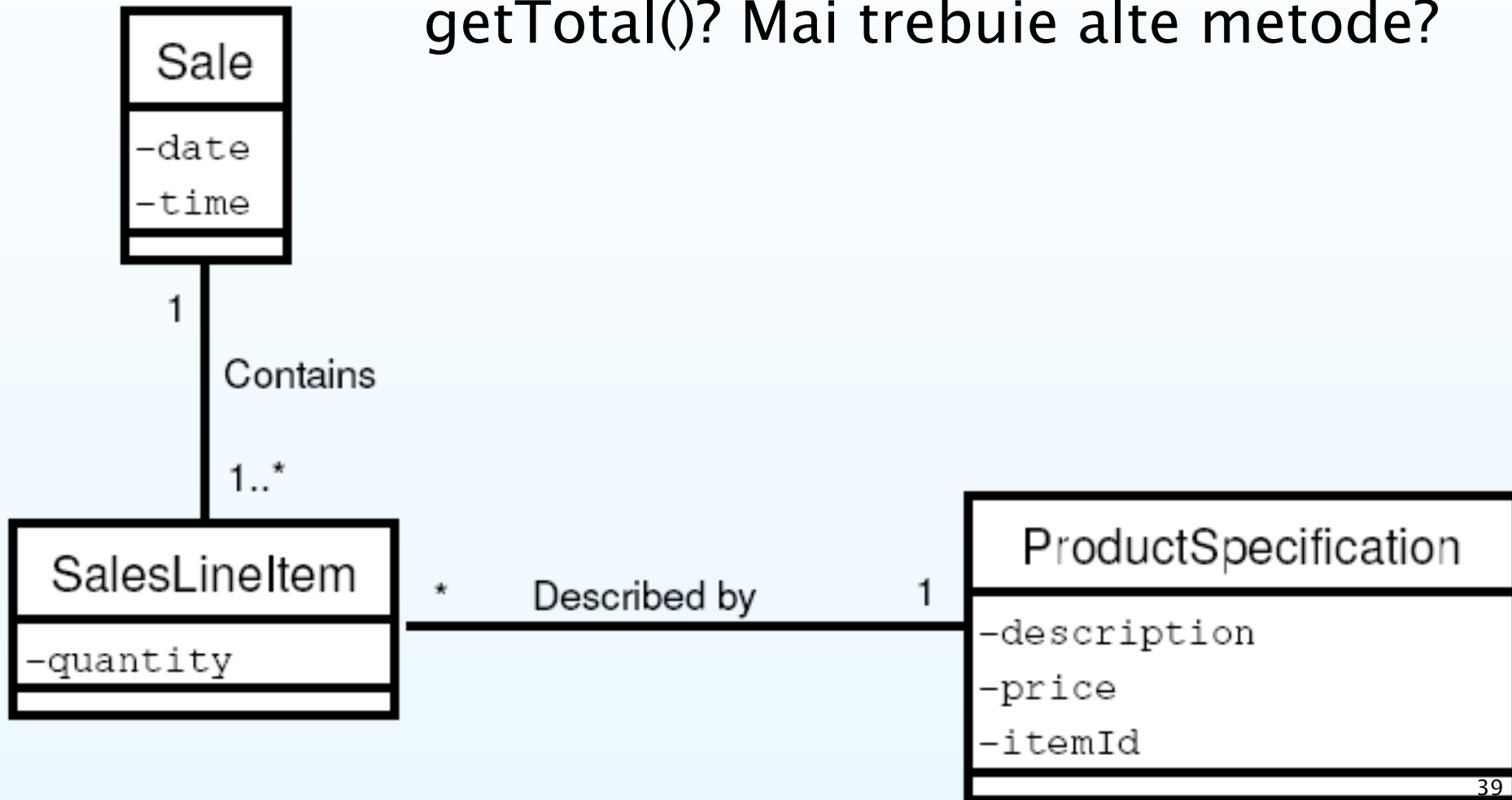
- ▶ **Problemă:** dat un anumit comportament (operație), cărei clase trebuie să-i fie atribuit?
- ▶ O alocare bună a operațiilor conduce la sisteme care sunt:
  - Ușor de înțeles
  - Mai ușor de extins
  - Refolosibile
  - Mai robuste

# Information Expert 2

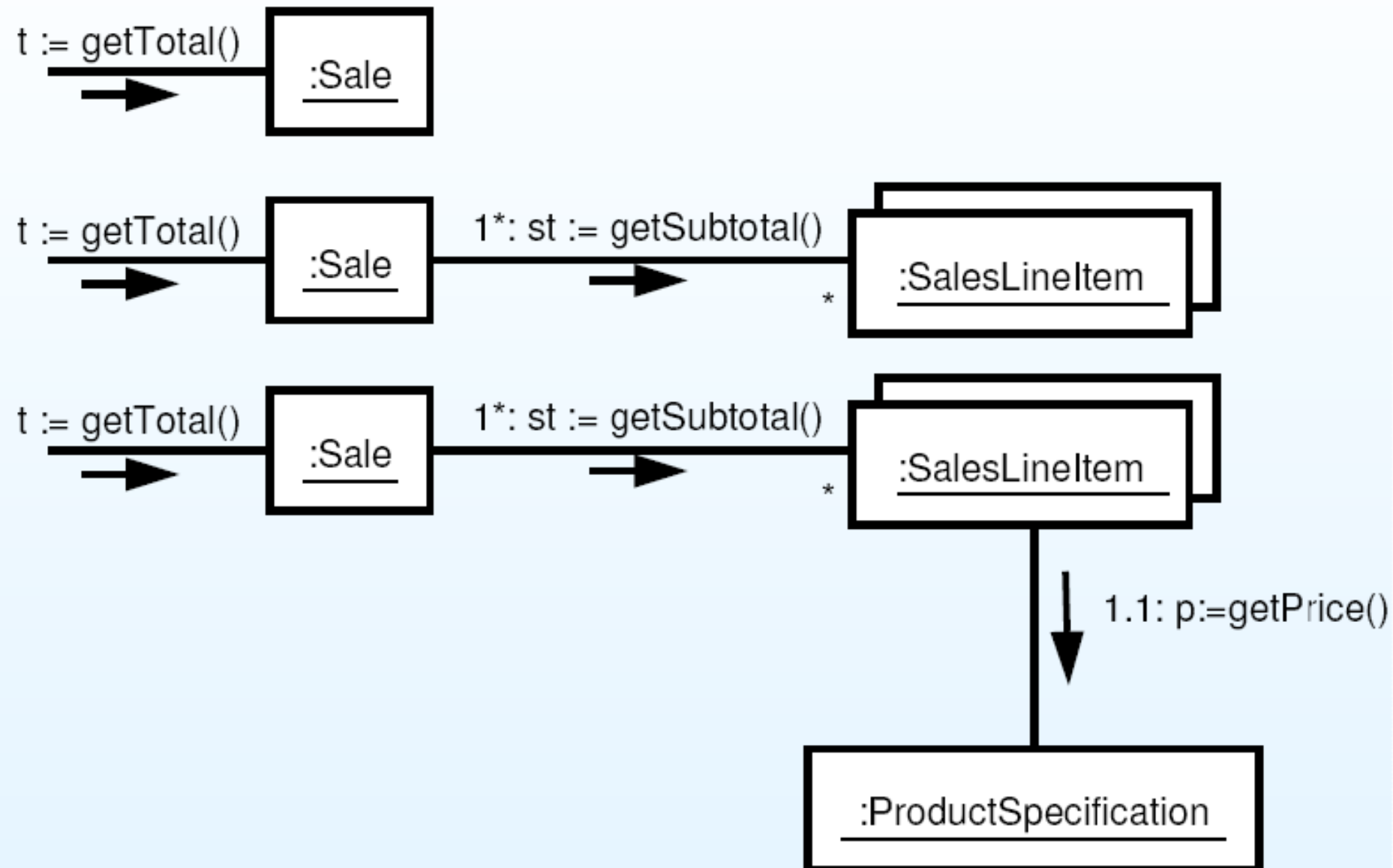
- ▶ **Soluție:**
- ▶ asignez o responsabilitate clasei care are *informațiile necesare* pentru îndeplinirea acelei responsabilități
- ▶ **Recomandare:**
- ▶ începeți asignarea responsabilităților evidențiind clar care sunt responsabilitățile

# Exemplul 1

- ▶ Carei clase trebuie sa-i fie asignată metoda getTotal()? Mai trebuie alte metode?



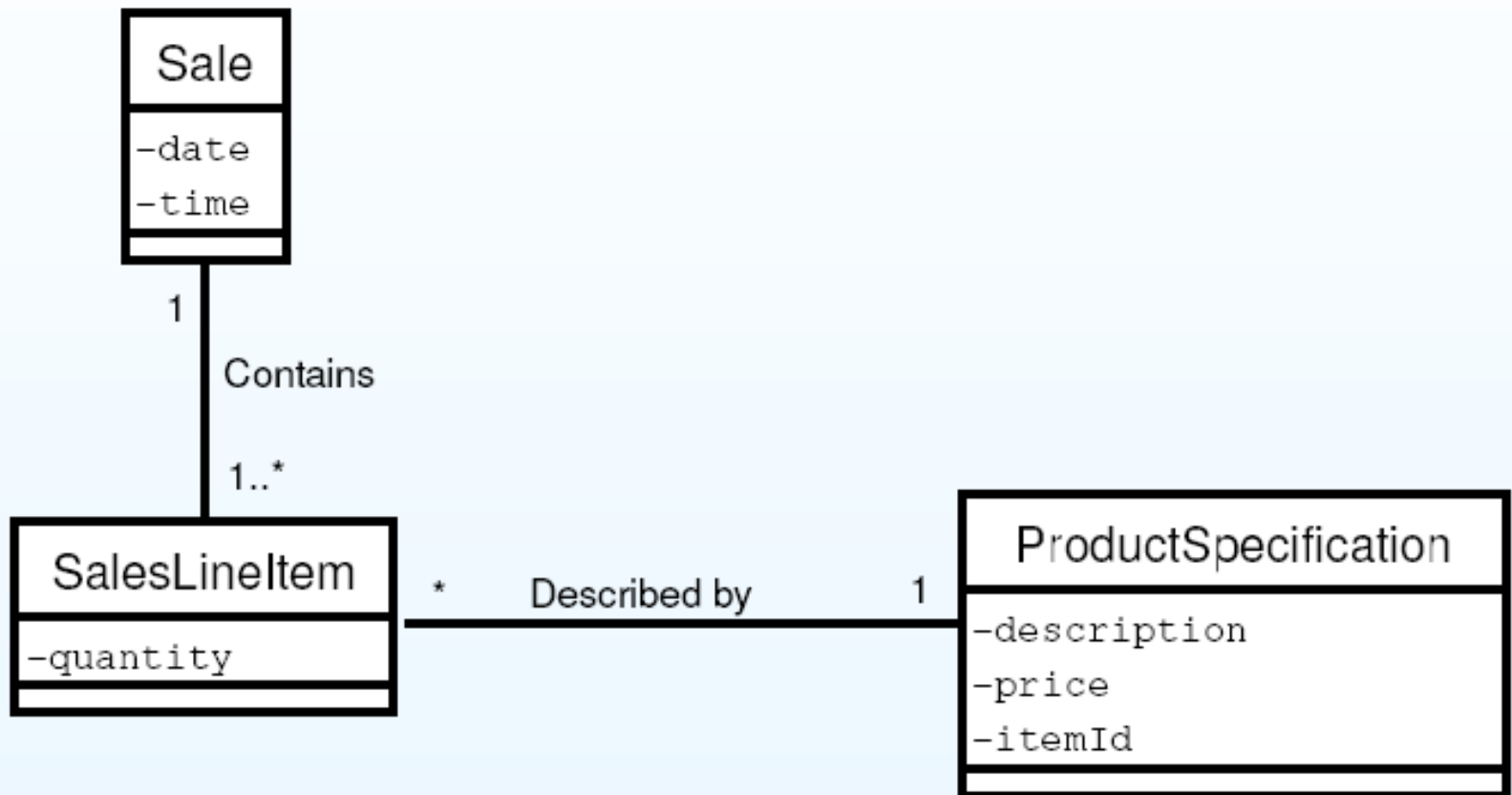
# Exemplul 2



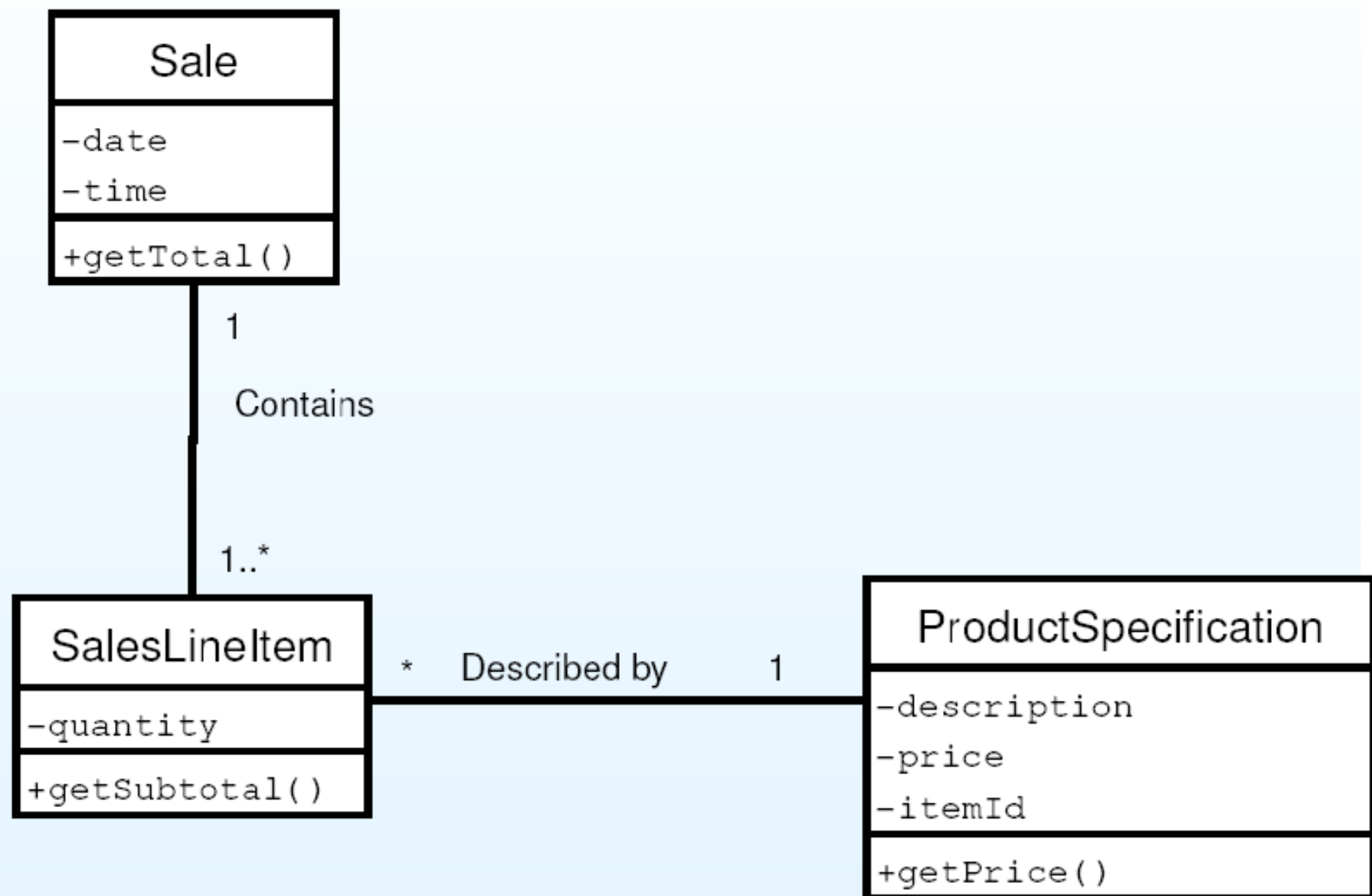


# Soluție posibilă 1

Clasă	Responsabilități
Sale	să cunoască valoarea totală a cumpărăturilor
SalesLineItem	să cunoască subtotalul pentru un produs
ProductSpecification	să cunoască prețul produsului



# Soluție posibilă 2

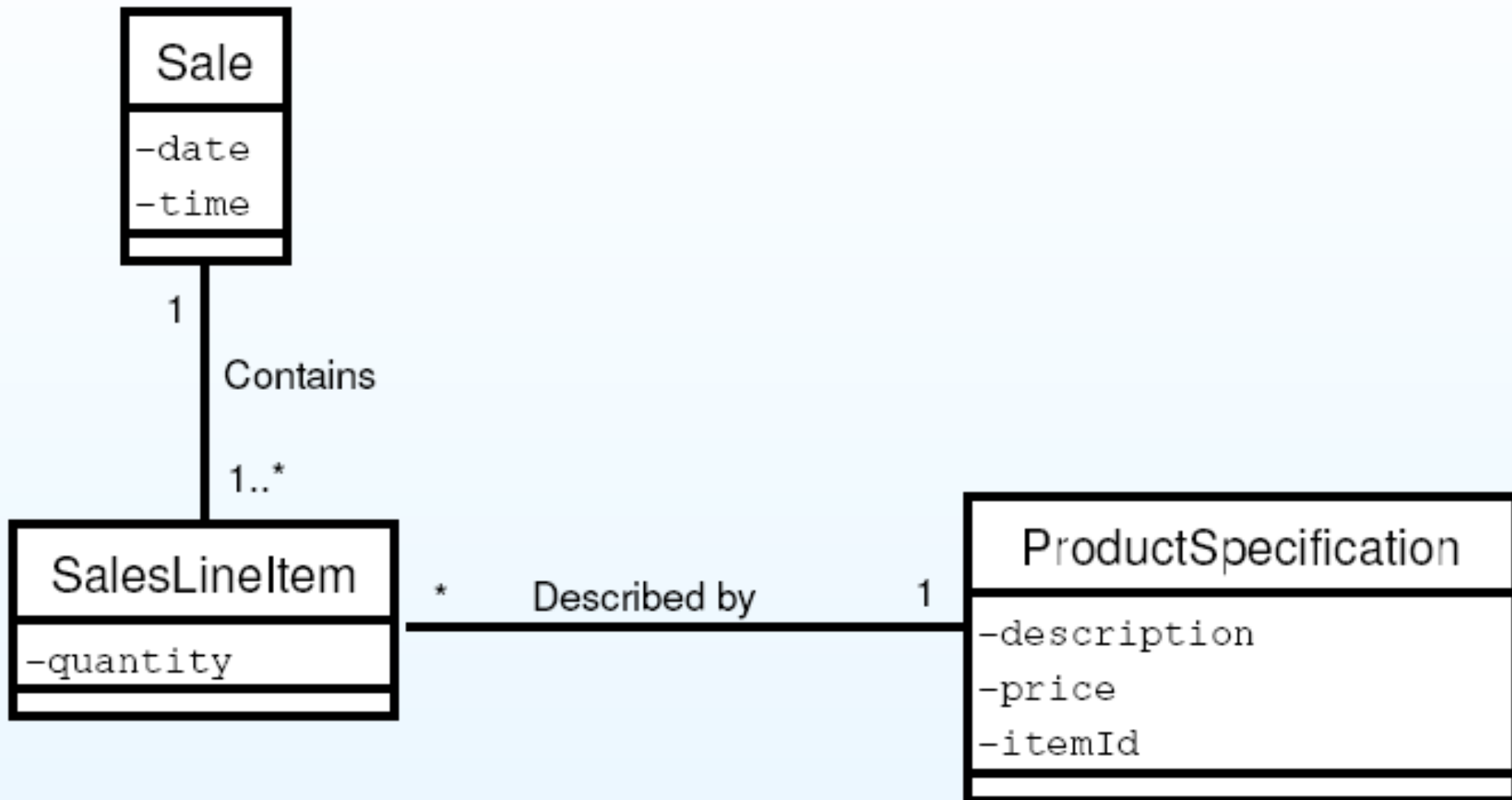


# Creator 1

- ▶ **Problemă:** cine trebuie să fie responsabil cu crearea unei instanțe a unei clase?
- ▶ **Soluție:** Asignați clasei B responsabilitatea de a crea instanțe ale clasei A doar dacă cel puțin una dintre următoarele afirmații este adevărată:
  - B *agregă* obiecte de tip A
  - B *conține* obiecte de tip A
  - B *folosește* obiecte de tip A
  - B *are datele de inițializare care trebuie transmise la* instanțierea unui obiect de tip A (B este deci un Expert în ceea ce privește crearea obiectelor de tip A)
- ▶ *Factory pattern* este o variantă mai complexă

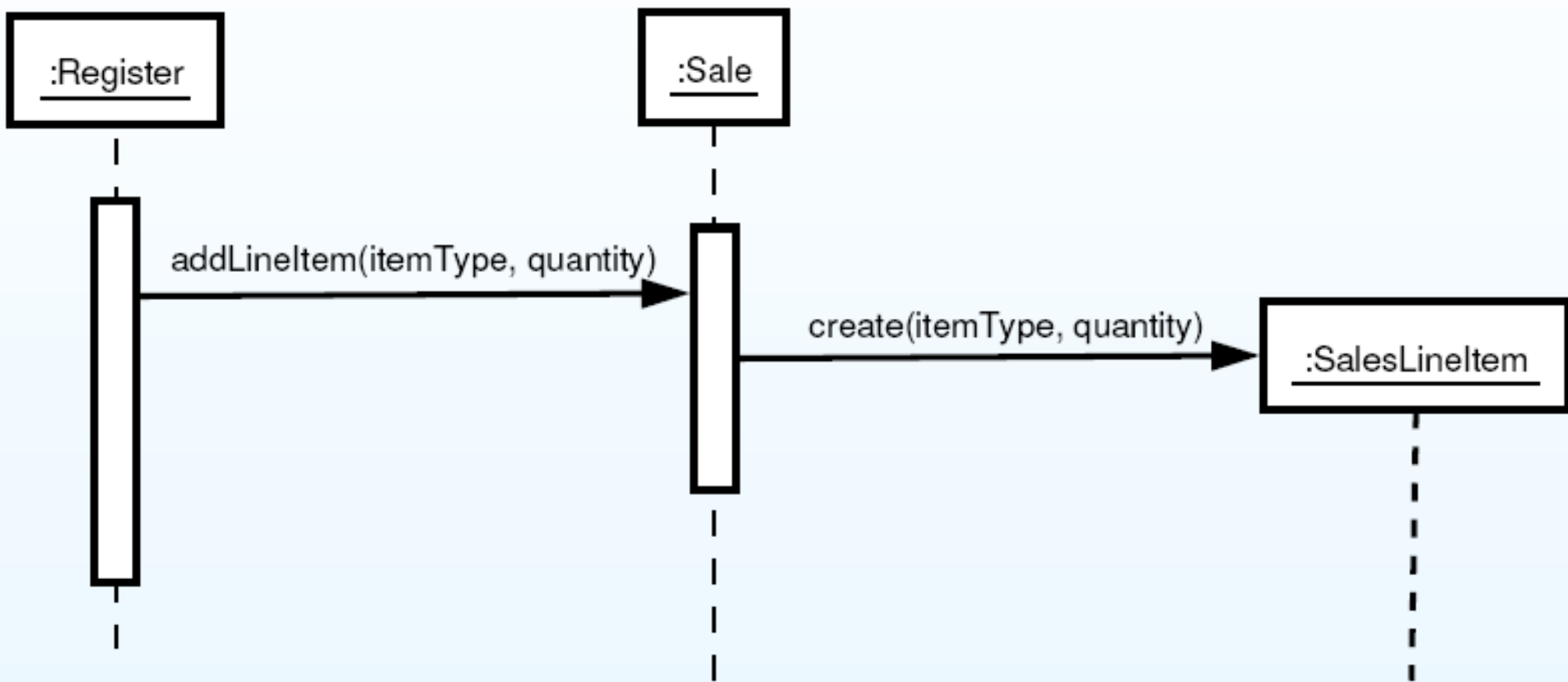
# Creator 2

- ▶ Cine este responsabil cu crearea unei instanțe a clasei SalesLineItem?



# Creator 3

- ▶ Deoarece Sale conține (agregă) instanțe de tip SalesLineItem, Sale este un bun candidat pentru a i se atribui responsabilitatea creării acestor instanțe



# Low coupling (cuplaj redus)

- ▶ Cuplajul este o măsură a gradului de dependență a unei clase de alte clase
- ▶ Tipuri de Dependență:
  - este conectată cu
  - are cunoștințe despre
  - se bazează pe
- ▶ **O clasă care are cuplaj mic (redus) nu depinde de “multe” alte clase; unde “multe” este dependent de contex**
- ▶ O clasă care are cuplaj mare depinde de multe alte clase



# Cuplaj 2

- ▶ Probleme cauzate de cuplaj:
  - schimbări în clasele relaționate forțează schimbări locale
  - clase **greu de înțeles** în izolare (scoase din context)
  - clase **greu de refolosit** deoarece folosirea lor presupune și prezența claselor de care depind

# Cuplaj 3

- ▶ Forme comune de cuplaj de la clasa A la clasa B sunt:
  - A are un atribut de tip B
  - O instanță a clasei A apelează un serviciu oferit de un obiect de tip B
  - A are o metodă care referențiază B (parametru, obiect local, obiect returnat)
  - A este subclasă (direct sau indirect) a lui B
  - B este o interfață, iar A implementează această interfață

# Legea lui Demeter

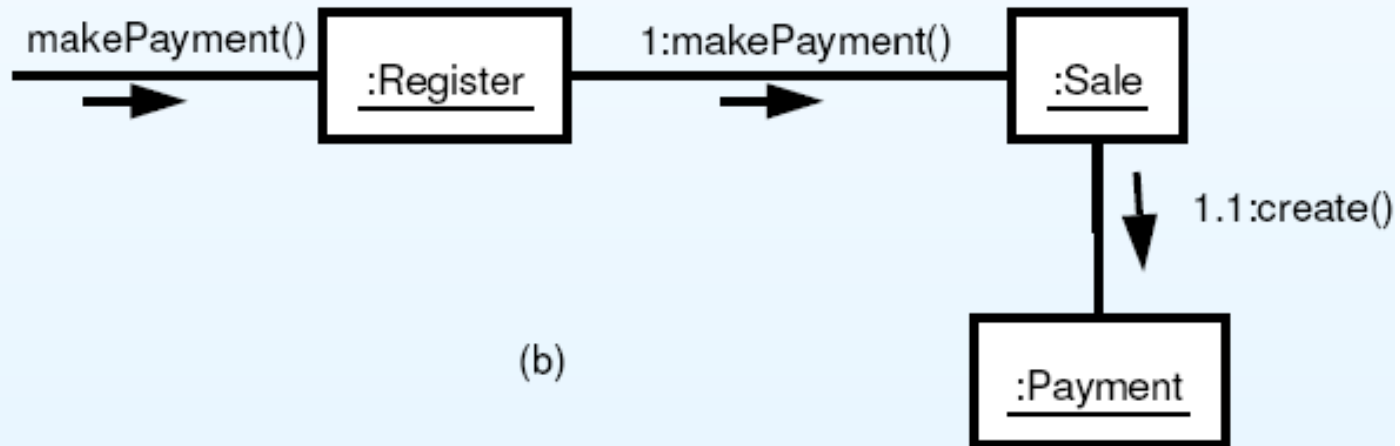
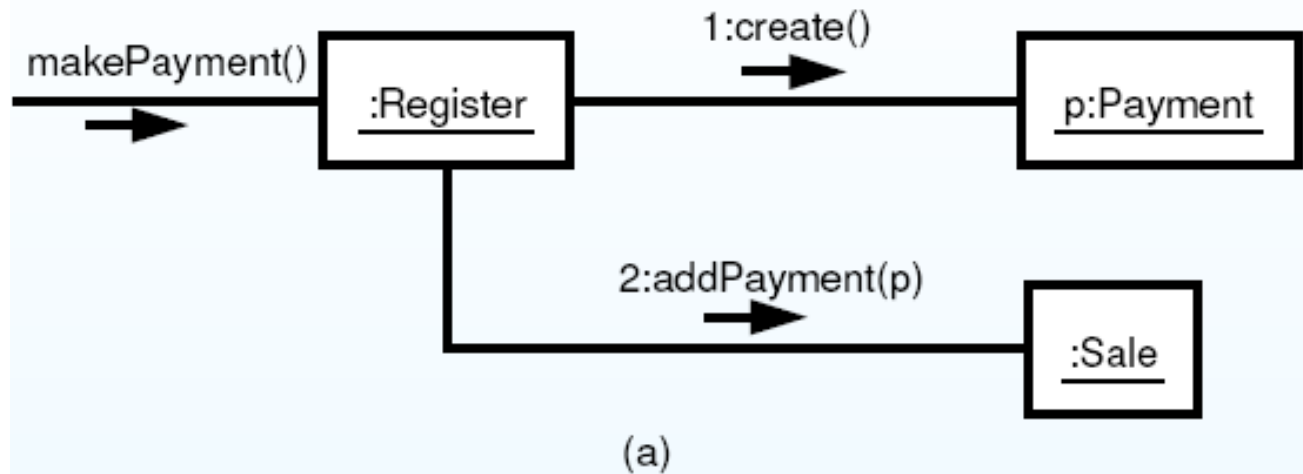
- ▶ *Don't talk to strangers*
- ▶ Orice metodă a unui obiect trebuie să apeleze doar metode aparținând
  - lui însuși
  - oricărui parametru al metodei
  - oricărui obiect pe care l-a creat
  - oricăror obiecte pe care le conține

# Vizualizarea Cuplajelor

- ▶ Diagrama de clase
- ▶ Diagrama de colaborare

# Exemplul 1

- ▶ Exista legături între toate clasele
- ▶ Elimină cuplajul dintre Register și Payment



# High Cohesion

- ▶ **Coeziunea** este o măsură a cât de puternic sunt focalizate responsabilitățile unei clase
- ▶ O clasă ale cărei responsabilități sunt foarte strâns legate și care nu face foarte multe lucruri are o **coeziune mare**
- ▶ O clasă care face multe lucruri care nu sunt relaționate sau face prea multe lucruri are o **coeziune mică (slabă)**



# Coeziune

- ▶ Probleme cauzate de o slabă coeziune:
  - greu de înțeles
  - greu de refolosit
  - greu de menținut
  - delicate; astfel de clase sunt mereu supuse la schimbări

# Coeziune și Cuplaj

- ▶ Sunt principii vechi în design-ul software
- ▶ Promovează un design modular
- ▶ Modularitatea este proprietatea unui sistem care a fost descompus într-o mulțime de module coezive și slab cuplate

# Controller 1

- ▶ **Problemă:** Cine este responsabil cu tratarea unui eveniment generat de un actor?
- ▶ Aceste evenimente sunt asociate cu operații ale sistemului
- ▶ Un **Controller** este un obiect care nu ține de interfața grafică și care este responsabil cu recepționarea sau gestionarea unui eveniment
- ▶ Un Controller definește o metodă corespunzătoare operației sistemului

# Controller 2

- ▶ **Soluție:** asignează responsabilitatea pentru recepționarea sau gestionarea unui eveniment unei clase care reprezintă una dintre următoarele alegeri:
  - Reprezintă întregul sistem sau subsistem (fațadă controller)
  - Reprezintă un scenariu de utilizare în care apare evenimentul;

# Controller 3

- ▶ În mod normal, un controller ar trebui să delege altor obiecte munca care trebuie făcută;
- ▶ **Controller-ul coordonează sau controlează activitatea, însă nu face prea multe lucruri el însuși**
- ▶ O greșeală comună în design-ul unui controller este să i se atribuie prea multe responsabilități (fațade controller)



# Concluzii

- ▶ Forward & Reverse Engineering
- ▶ GRASP
  - Information Expert
  - Creator
  - Low coupling
  - High cohesion
  - Controller

# Bibliografie

- ▶ Reverse Engineering and Design Discovery: A Taxonomy, Chikofsky, E.J. and Cross, J., January, 1990
- ▶ Craig Larman. *Applying UML and Patterns. An Introduction to Object Oriented Analysis and Design*
- ▶ Ovidiu Gheorghies, Curs 6 IP

# 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](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/>
- ▶ **Intelligent:** <http://www.intelligenttrd.com/>
- ▶ **Smartphones RE:** <http://www.cytraxsolutions.com/2011/01/smartphones-security-and-reverse.html>

# Links (GRASP)

- ▶ WebProjectManager: <http://profs.info.uaic.ro/~adrianaa/uml/>
- ▶ Diagrame de Stare și de Activitate:  
[http://software.ucv.ro/~soimu\\_anca/itpm/Diagrame%20de%20Stare%20si%20Activitate.doc](http://software.ucv.ro/~soimu_anca/itpm/Diagrame%20de%20Stare%20si%20Activitate.doc)
- ▶ Deployment Diagram:  
[http://en.wikipedia.org/wiki/Deployment\\_diagram](http://en.wikipedia.org/wiki/Deployment_diagram)  
<http://www.agilemodeling.com/artifacts/deploymentDiagram.htm>
- ▶ GRASP:  
[http://en.wikipedia.org/wiki/GRASP\\_\(Object\\_Oriented\\_Design\)](http://en.wikipedia.org/wiki/GRASP_(Object_Oriented_Design))
- ▶ <http://web.cs.wpi.edu/~gpollice/cs4233-a05/CourseNotes/maps/class4/GRASPpatterns.html>
- ▶ Introduction to GRASP Patterns:  
[http://faculty.inverhills.edu/dlevitt/CS%202000%20\(FP\)/GRASP%20Patterns.pdf](http://faculty.inverhills.edu/dlevitt/CS%202000%20(FP)/GRASP%20Patterns.pdf)