What is UML?

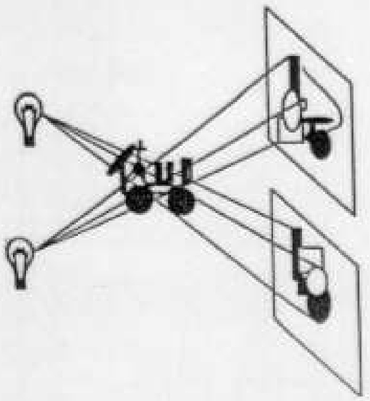
1997 - 99

- UML is an Object-Oriented modeling language.
- UML originates from OMT, Booch, and OOSE developed by Rumbaugh, Booch, and Jacobsen respectively.
- UML is maintained and further developed by the OMG.
- Telelogic is a full member and is also co-chairing the RTAD (real-time analysis design) group.
- We are defining the extensions (SDL) to UML that will include support for real-time modelling.

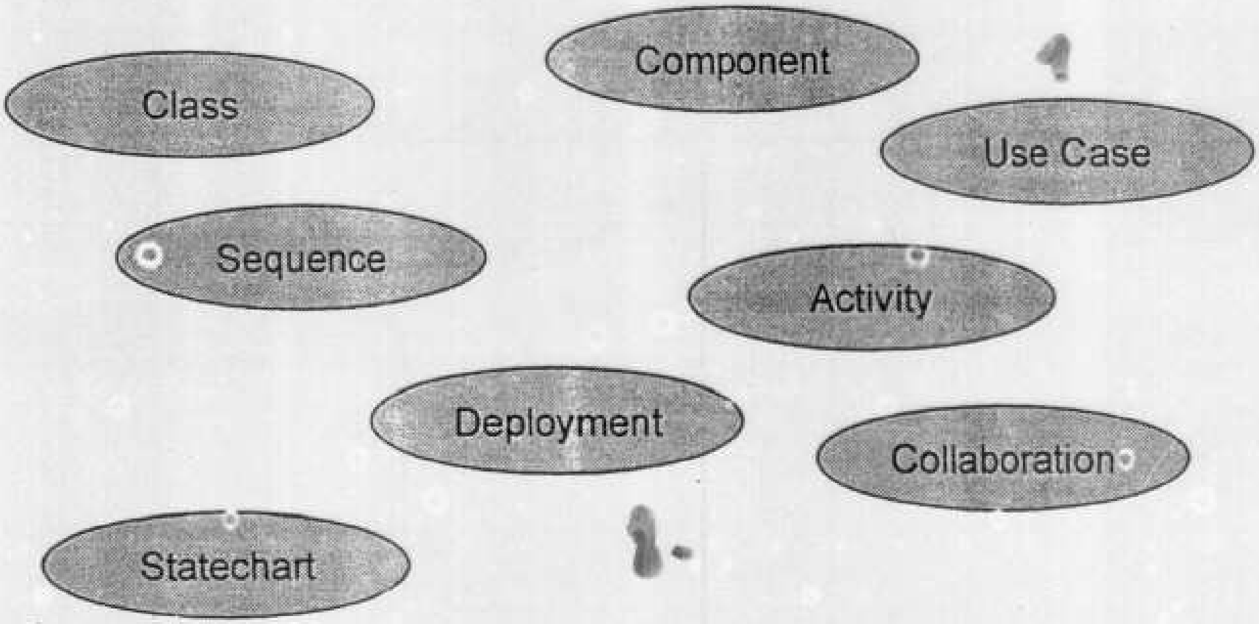
It is a notation, not a method!

The notation

Most systems being modeled are so complex that a single kind of diagram cannot clearly and completely describe the system. Because of this, UML provides different diagram types, each type offering a different view or perspective on the system.



UML diagrams

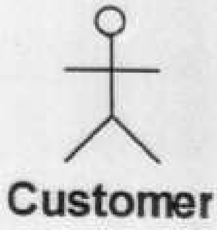


Building Requirements Model with Use Case Diagrams

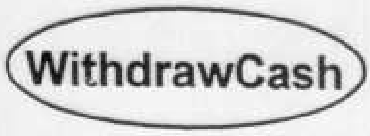
- Purpose: discover, describe, agree to the functionality required of the system
- Main audience:
  - Users and management, who need to plan and fund the project
  - Analysts, who need to understand what is being requested
- Remember to exclude: the ***form*** of the system. This is about the ***function*** of the system
- Describes what an end user expects from the system

Actors

- A type of external entity that interacts with the system.
  - Users
  - Other systems or devices
- Initiates, communicates with, or receives information from one or more use cases

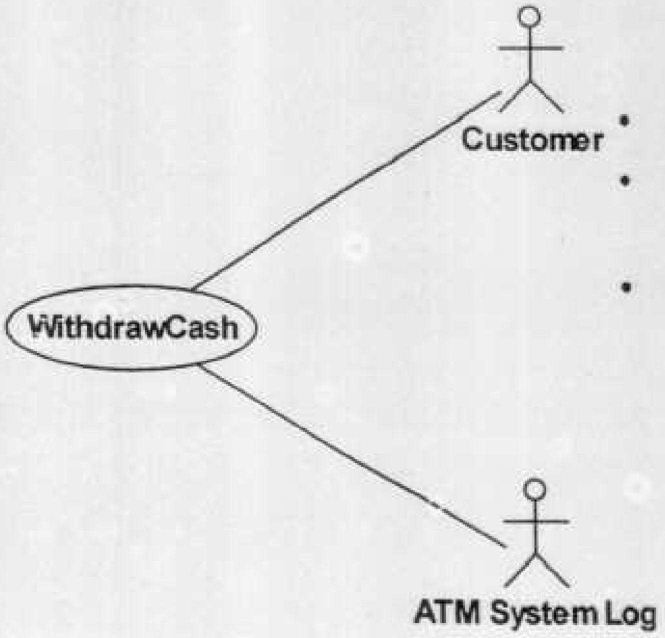


Use Case



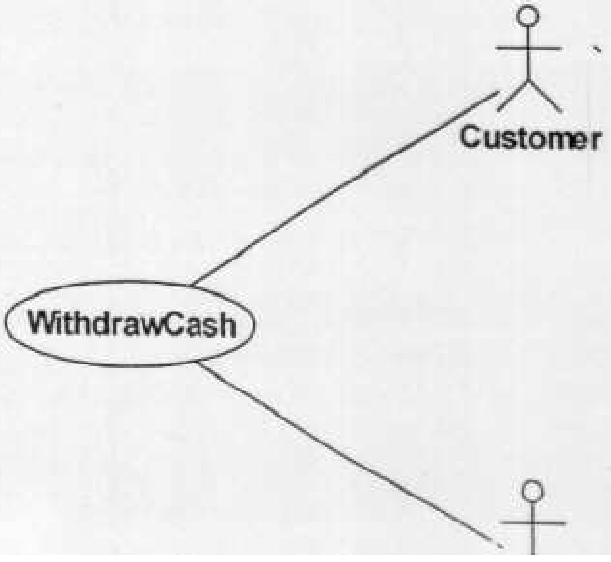
- Named for a ***goal*** of an actor
- Initiated by an actor
- An abstraction of an actor/system interaction
- Represents possible interaction sequences
  - Well-defined starting point and goal
  - Includes sequences that fail to reach the goal

Use Case Association



- Drawn between actor and use case
- Represents the communication between actor and system
- Can be:
  - undirected - either end might initiate communication
  - directed - shows direction of flow of the initiating event

Use Case Diagram Example: ATM - Withdraw Cash



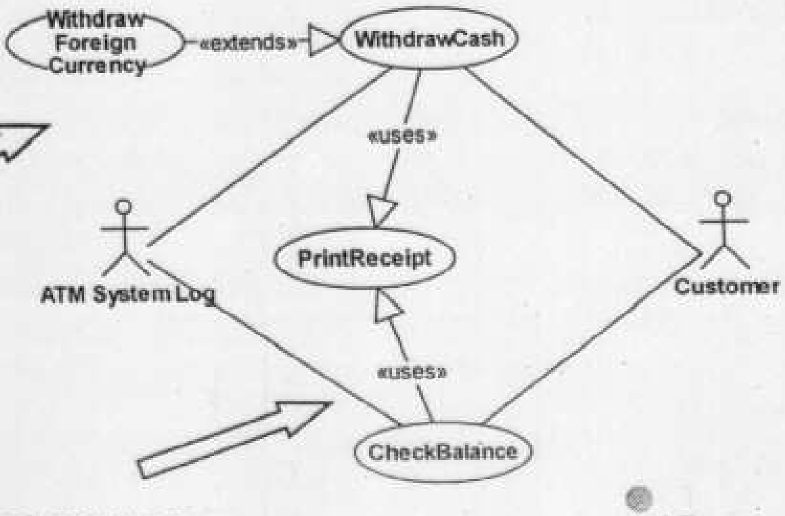
Use case diagram is named for the goal of the actor

Stereotypes

- It is possible to make extensions to the notation described in UML. The extensions are called stereotypes.
- A stereotype is written within guillemets, (« »). There are a number of predefined stereotypes, for example «actor», «uses», and «extends».
- Includes the properties of the base symbol

Use Case - «uses» and «extends».

<<extends>> shows optional behavior

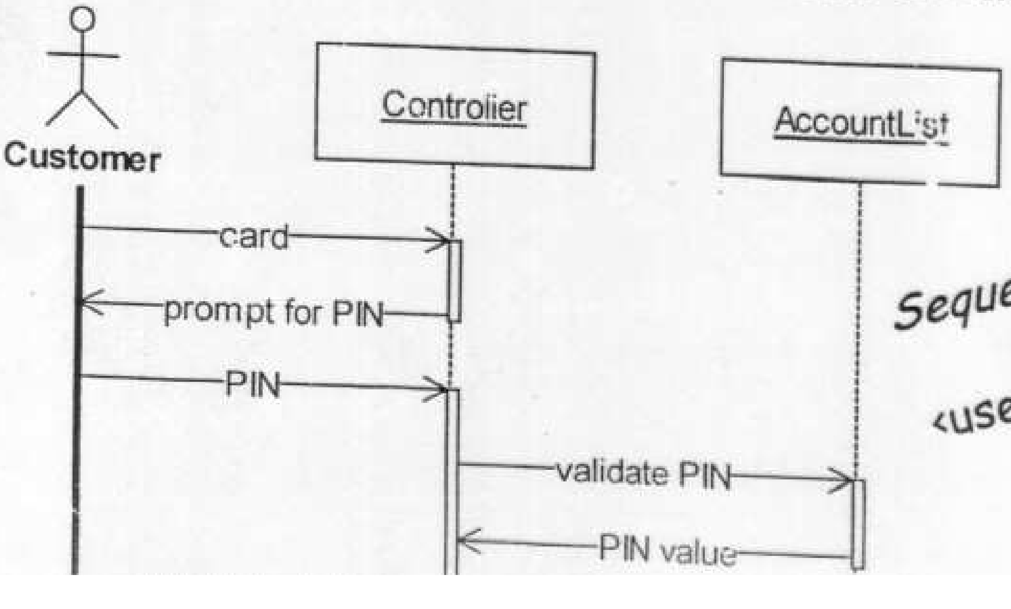


<<uses>> describes common

Modeling Object Interactions using Sequence Diagrams

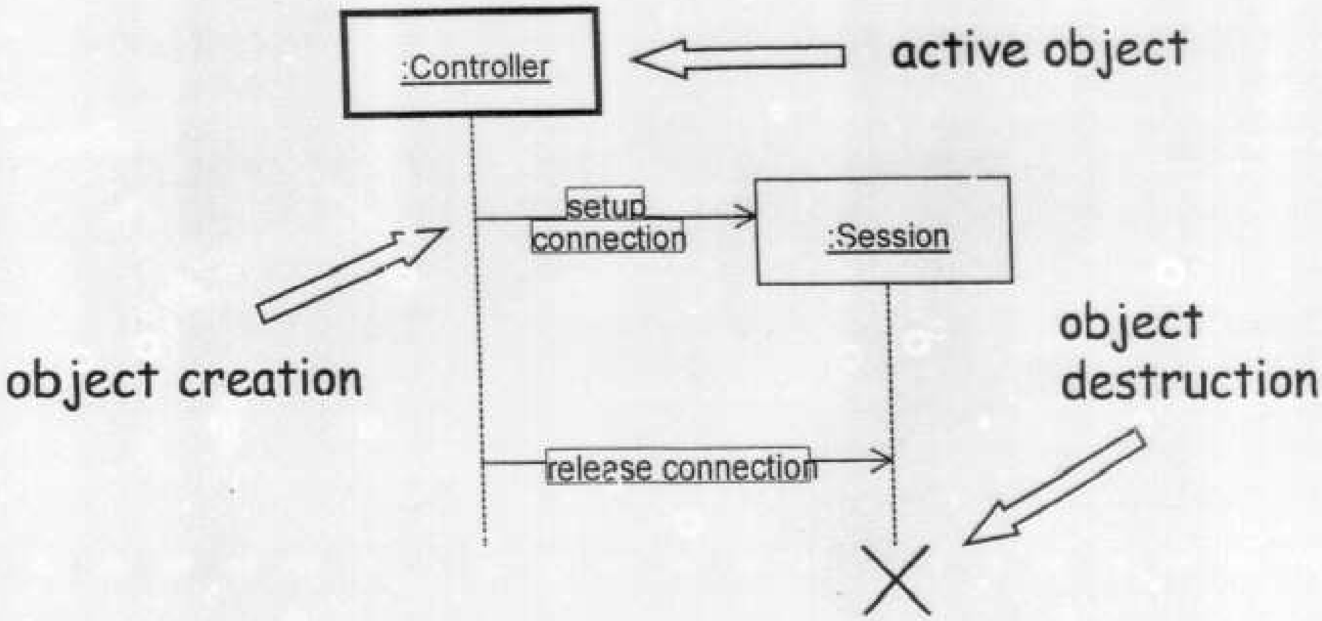
- Purpose: draw scenarios based on basic and alternate courses of use cases.
- Typically one or more sequence diagrams per use case, starting with the basic course
- Shows messages between objects, which is the only way objects can interact

Sequence Diagram Example Withdraw Cash:Basic Course



Sequence diagram is named as <use case>:<course>

Sequence Diagram (continued)

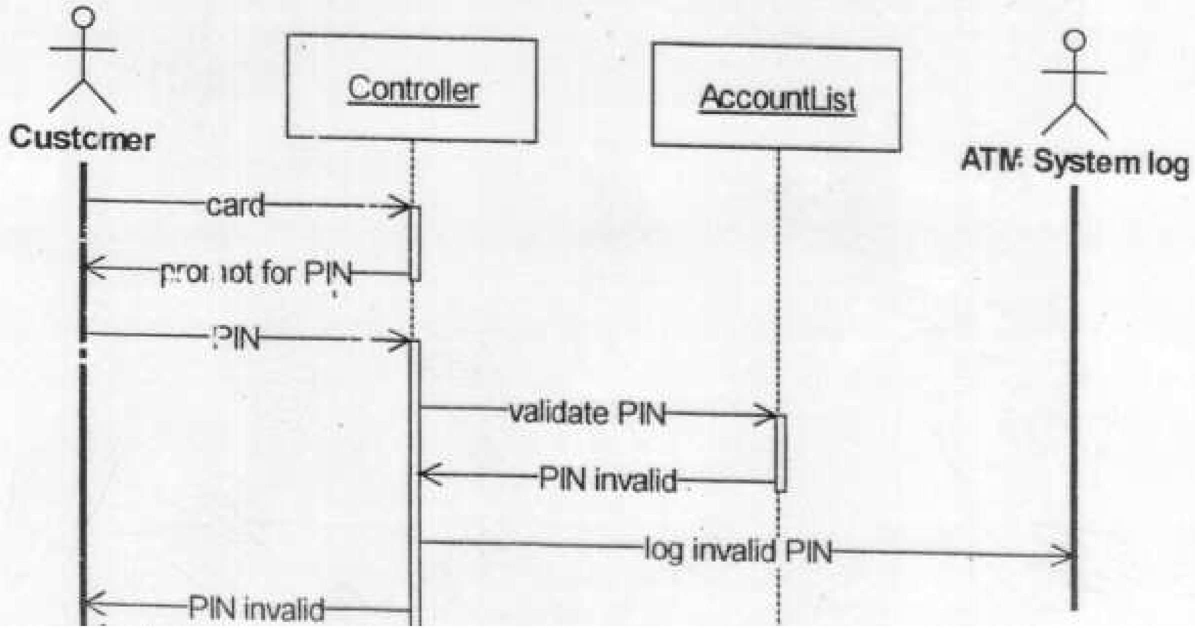


active object

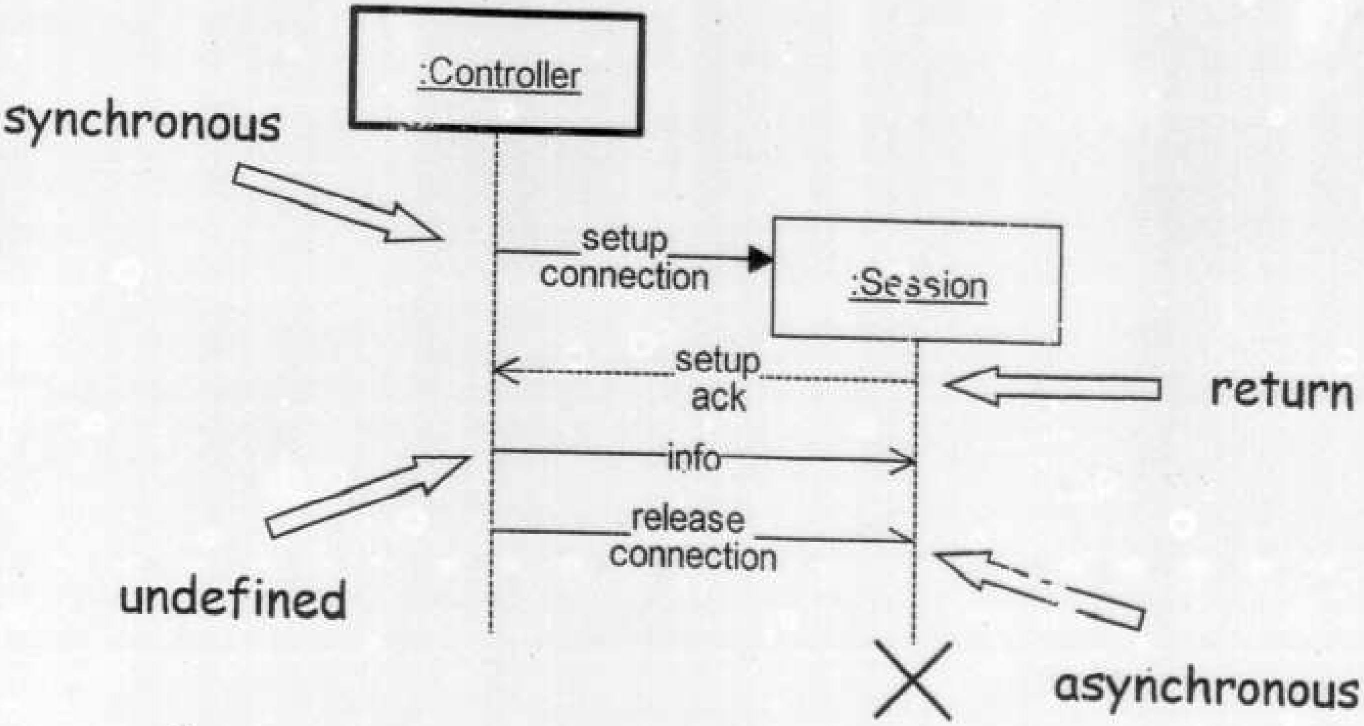
object creation

object destruction

Sequence Diagram Example: Withdraw Cash:Invalid PIN

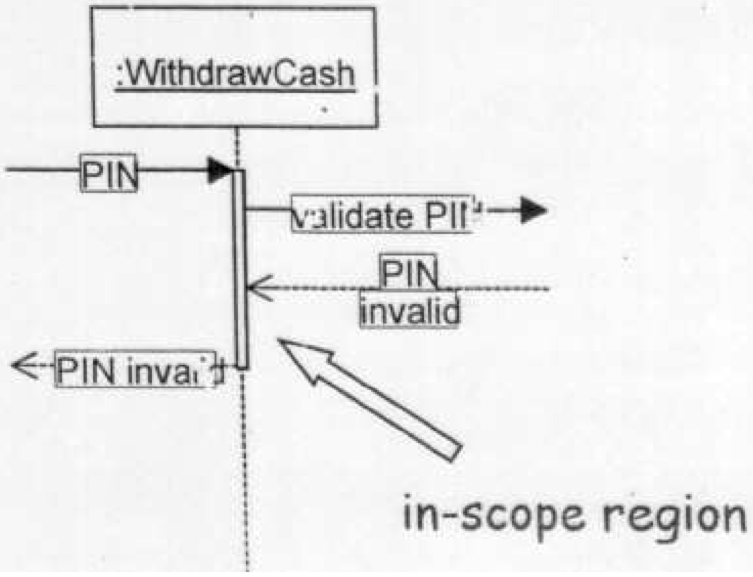


Sequence Diagram (continued)



Sequence Diagram (continued)

The period of time when the object is executing and/or waiting for a return message is called in-scope region.

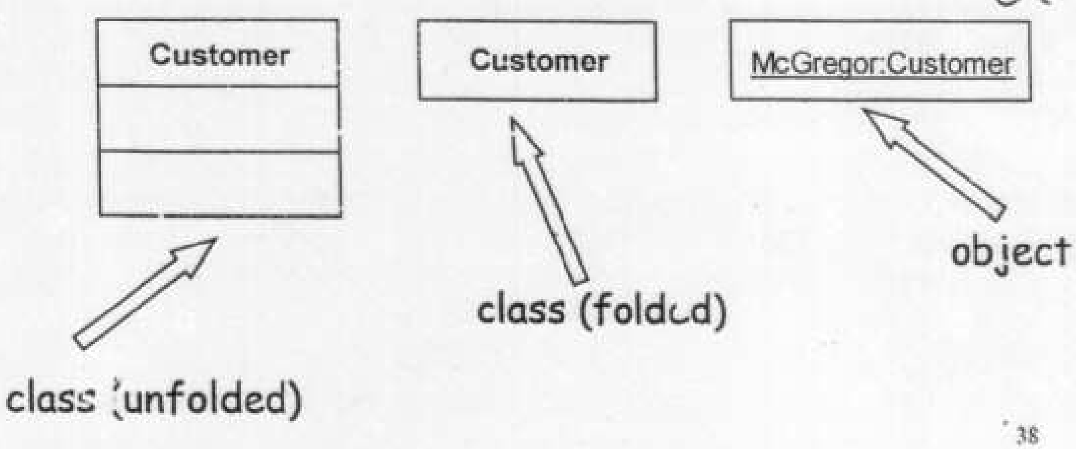


Build Static Model Using Class Diagrams

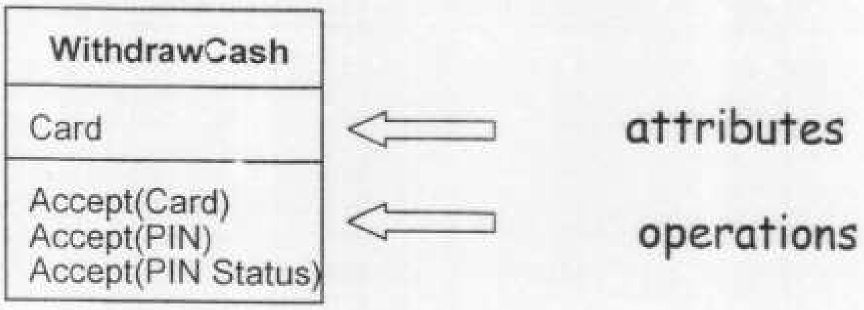
- Purpose: define the core classes which comprise the logical structure of the system
- The class diagram is the back-bone of the whole UML model
- Specifies the entities in the system, their attributes and operations and the way they relate to each other.

Classes and objects

- A ***class*** is a descriptor for a set of objects that share the same definitions of attributes, operations, and relationships
- An ***object*** is an instantiation of a class



Attributes and operations



WithdrawCash

Card

Accept(Card)
Accept(PIN)
Accept(PIN Status)

attributes

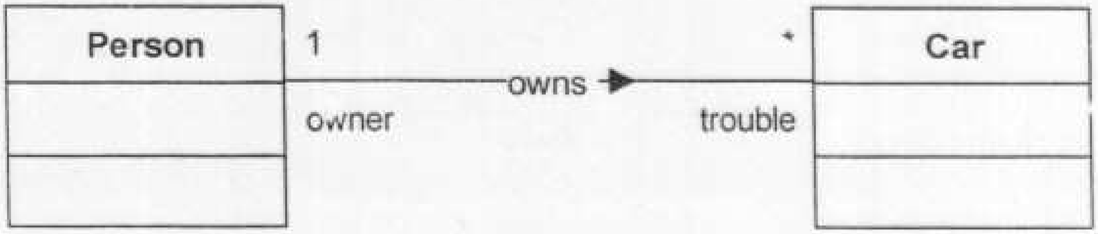
operations

- Attributes and operations are refined from messages (and the responsibilities those messages imply) in the classes that receive them.
- Operations will not necessarily coincide with messages.
- Message is an abstract concept, not a property of a class

Association

- An association shows that the classes are related with each other in a way that is interesting to model

"one person owns zero to many cars "



Person

1

owns

owner

*

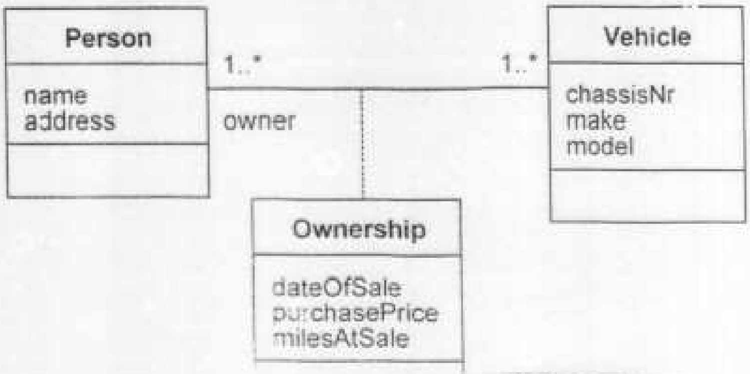
trouble

Car

" a person is the owner of a car "

Other elements - Association Classes

- Association classes occur with "many-to-many" associations
- It is rather an association with the characteristics of a class
  - Attributes and operations belong to the link, not to the objects


Person

name
address

1..*

owner

1..*

Vehicle

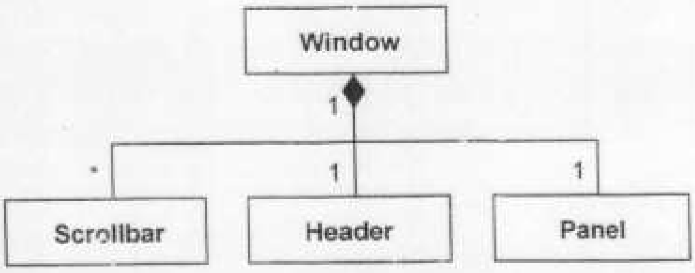
chassisNr
make
model

Ownership

dateOfSale
purchasePrice
milesAtSale

Composition

- A whole is responsible for the creation and destruction of its parts
- A part can belong to only one whole at a time
- When a whole is destroyed, its parts must cease to exist as well
- It's a strong form of aggregation


Window

1

*

1

1

Scrollbar

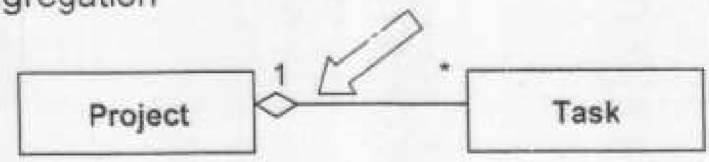
Header

Panel

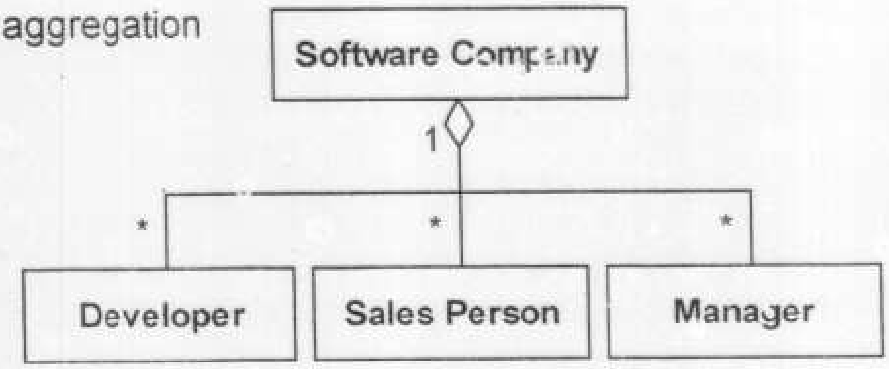
Aggregation

Aggregation symbol

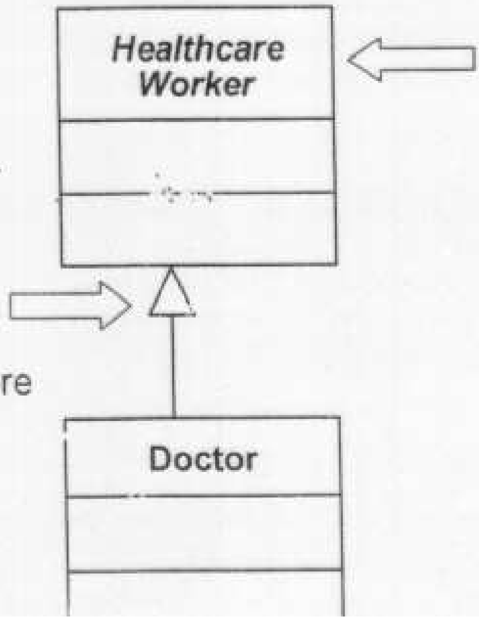
Homogeneous aggregation



Heterogeneous aggregation



Generalization



Note: italics indicate an abstract class (a class without any direct instances)

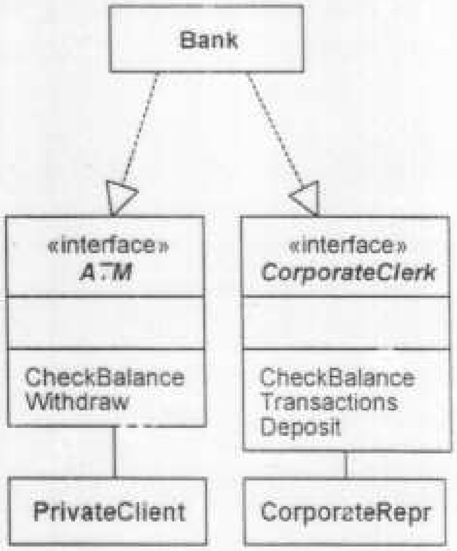
Hollow triangular arrowhead and solid line

Arrow points to the more general class

There are instances of "Doctor", but no instances of just "Health Care Worker"

Other elements - Interfaces

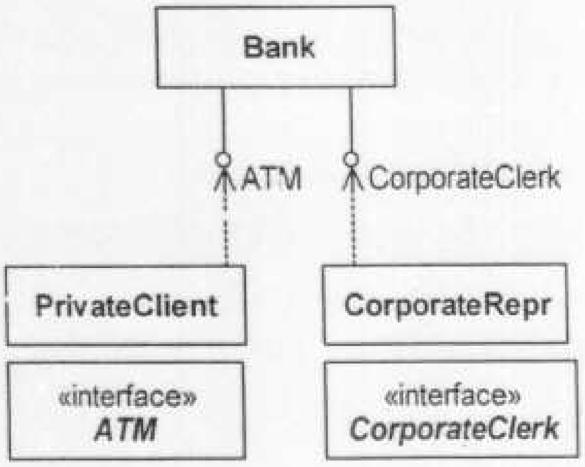
- A description of the externally visible and accessible behavior of a class, a package or a component.



Interfaces - The Interface symbol

- **UML notation:** represented as a small circle attached to the supporting entity by a solid line (a lollipop)

- **Application :** formally equivalent to an interface class. The interface class definition must appear in the class diagram.

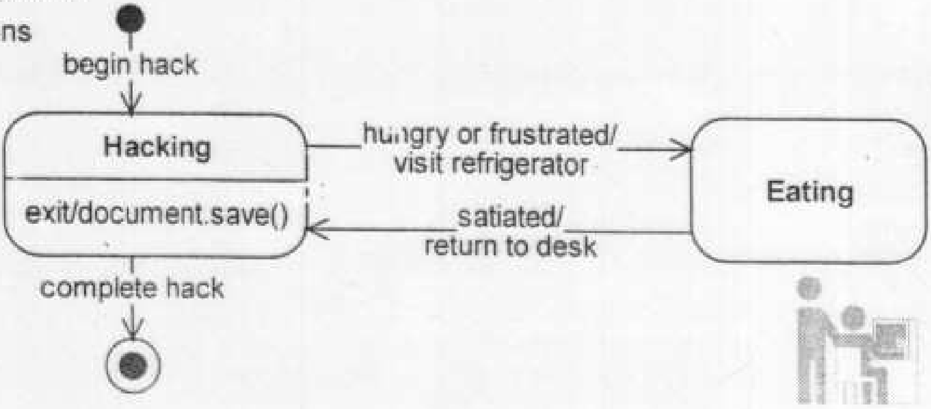


Building Dynamic Model using Statechart Diagrams

- Purpose
  - Specify the life-cycle of an object, the events it responds to, and its behavior in response to those events
- A dynamic model must be created, but statecharts do not have to be used for all parts of it
- In RT and embedded software development, statecharts are often considered to be *the* core technique.

UML statechart diagrams

- A statechart depicts the behavior of an object as
  - States the object can be in.
  - Permissible transitions between states
  - How the object responds to events with
    - Transitions
    - Actions



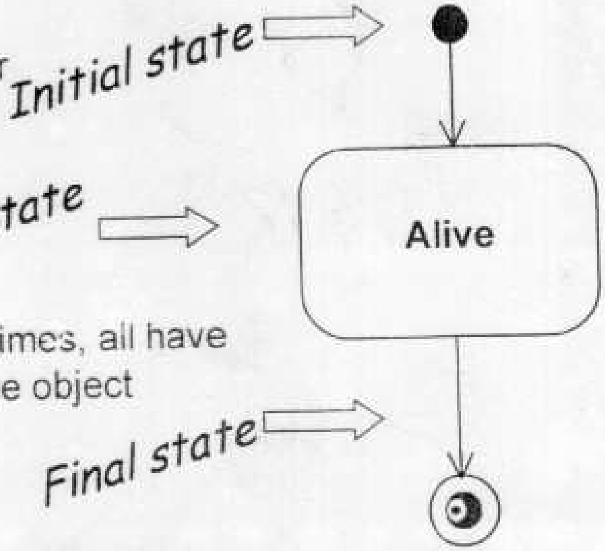
States

A period of time in the life of an object during which it:

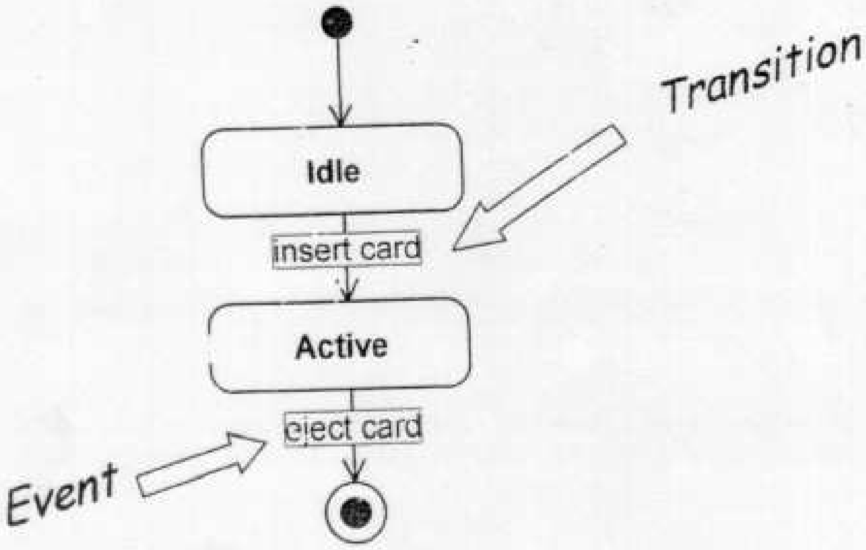
- Satisfies some condition, and/or
- Performs some activity, and/or
- Waits for some events

Some special states:

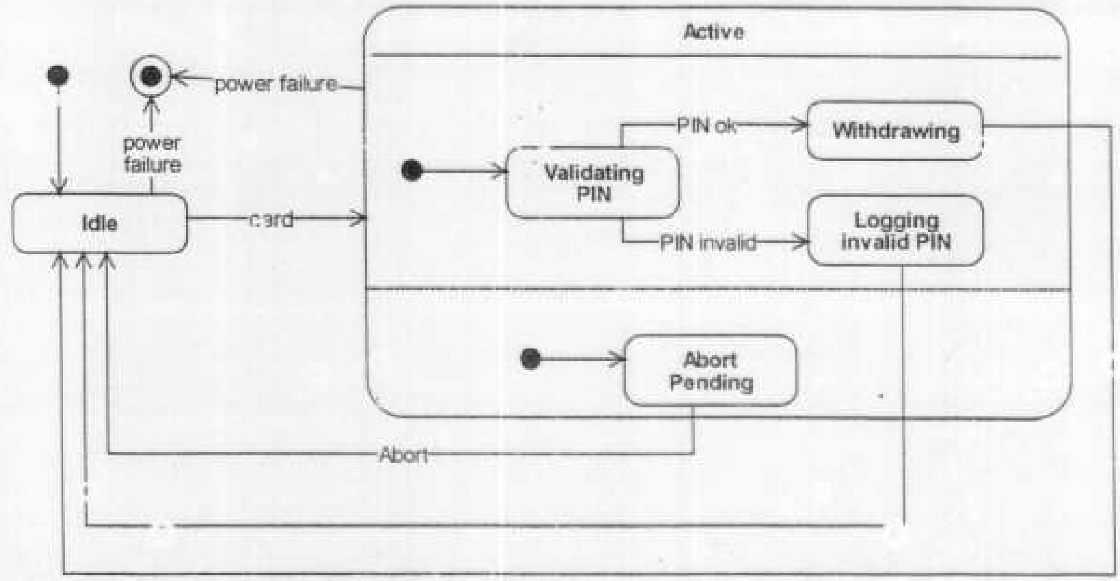
- Initial state: At most one
- Final state: Can appear many times, all have the same meaning - namely, the object ceases to exist



Transitions and Events

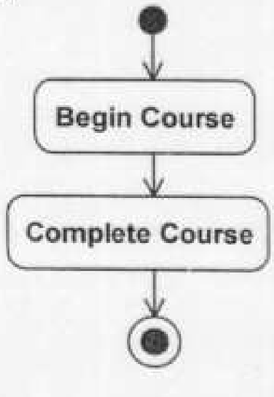


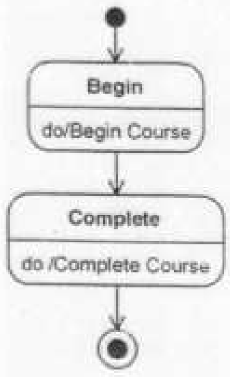
Statechart Example
WithdrawCash:Detailed



Other Notations - Activity Diagram

- Documenting procedural steps of a single operation
- Emphasizes procedural flow rather than (explicit) event-driven flow

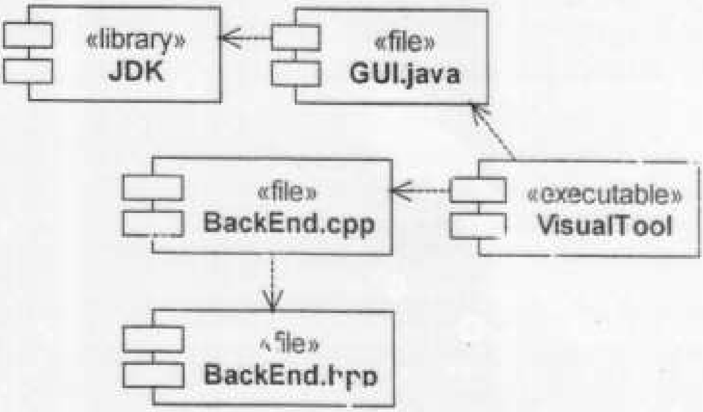




Activity diagram vs. *Statechart diagram*

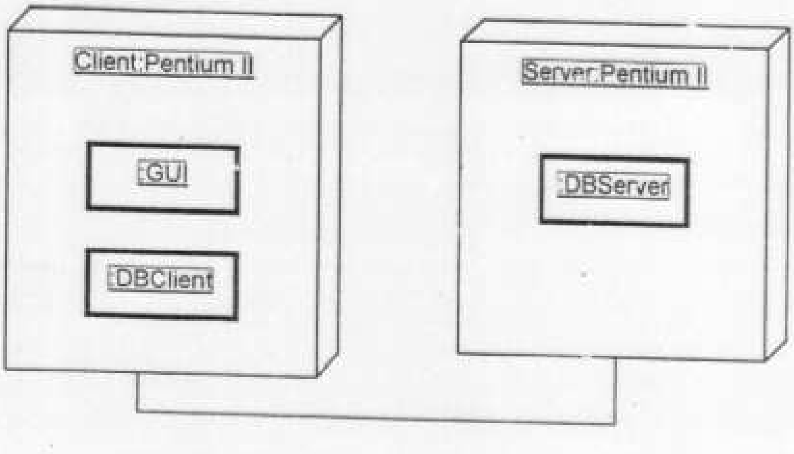
Other Notations - Component Diagram

A component diagram shows the dependencies among software components, including source code components, binary code components and executable components.

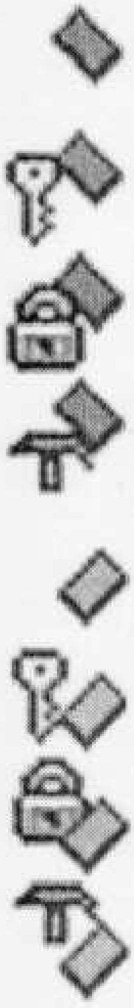


Other notations - Deployment Diagram

- Models how a system will be implemented
- Shows the configuration of run-time processing elements and the software components, processes and objects that live on them



Specifikace tříd



Attributes

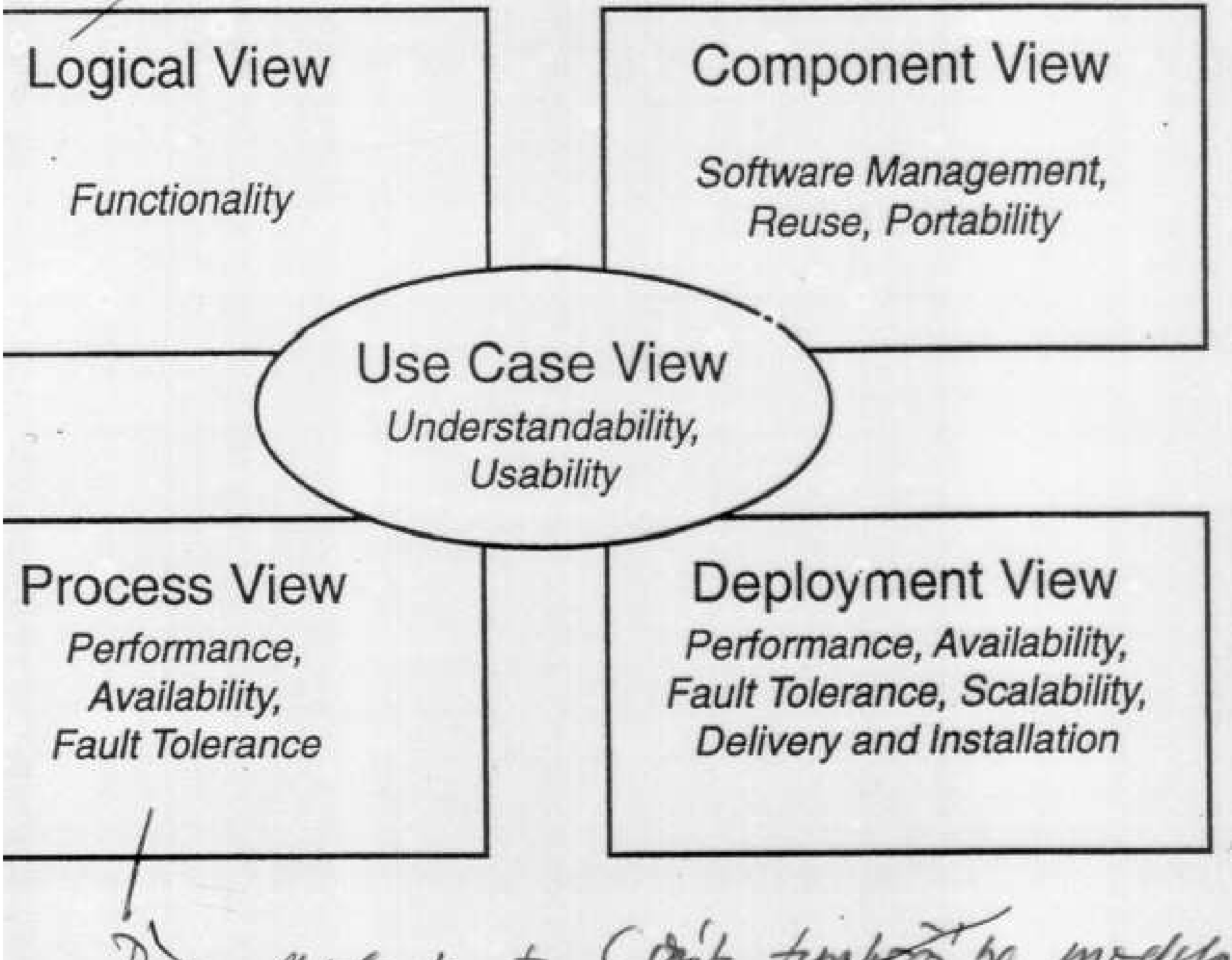
- Public
- Protected
- Private
- Implementation

Operations

- Public
- Protected
- Private
- Implementation

ZÁKLADNÍ DEKOMPOZICE PROSTŘEDÍ VIZUALIZACE (I PROGRAMOVÁNÍ) V UML (RR'98).

OBJEKTOVÝ MODEL (TŘÍDOVÝ, STAVOVÝ, SEKVENČNÍ, ...)



Logical View
Functionality
Component View
Software Management, Reuse, Portability
Use Case View
Understandability, Usability
Process View
Performance, Availability, Fault Tolerance
Deployment View
Performance, Availability, Fault Tolerance, Scalability, Delivery and Installation

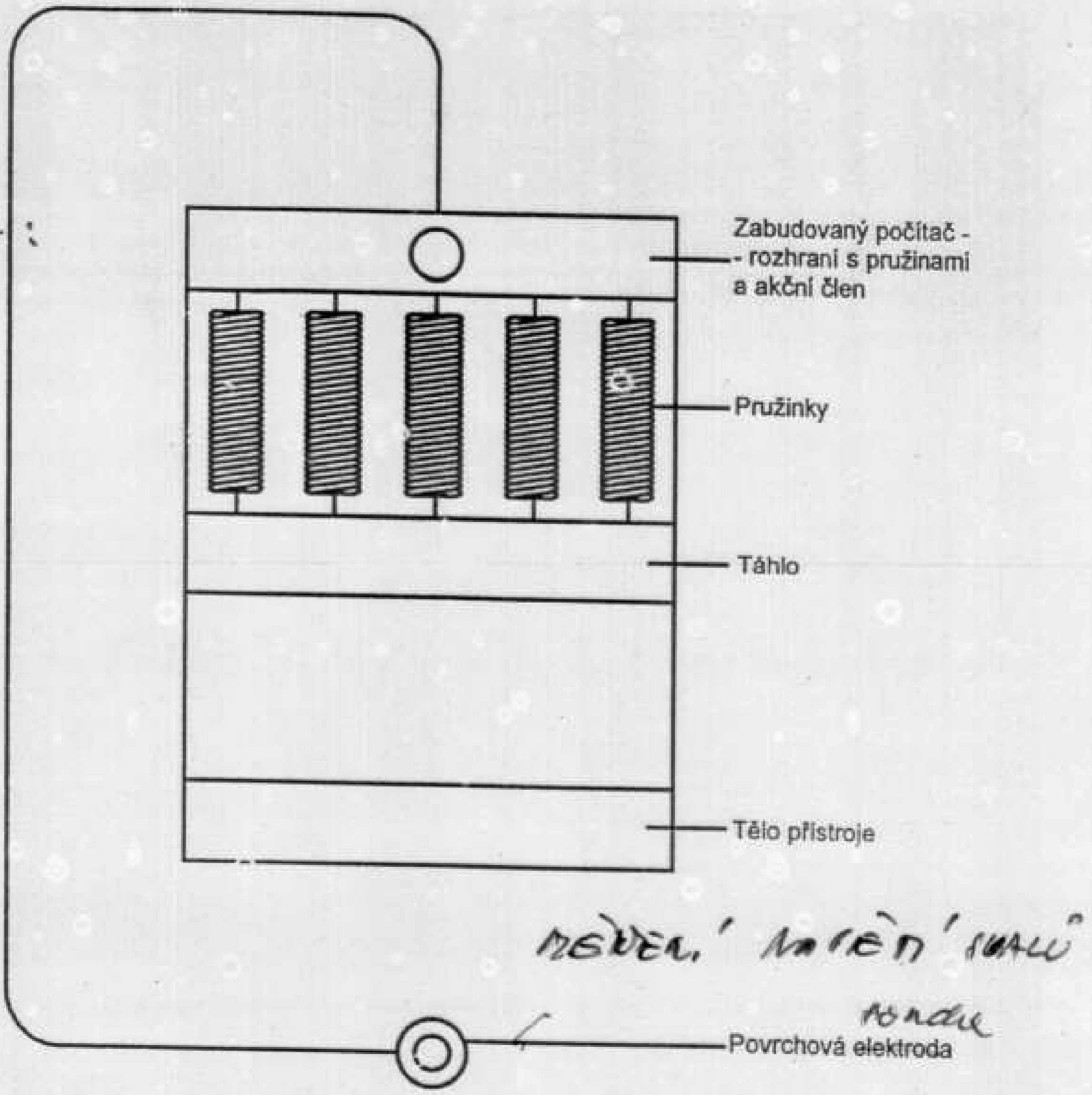
Dyn. model + (část funkčního modelu)

STAVOVÝ DIAGRAM

DIAGRAM INTERAKCE

POSILOVAČ SVALŮ DLANĚ

SCHEMA:



USE CASE

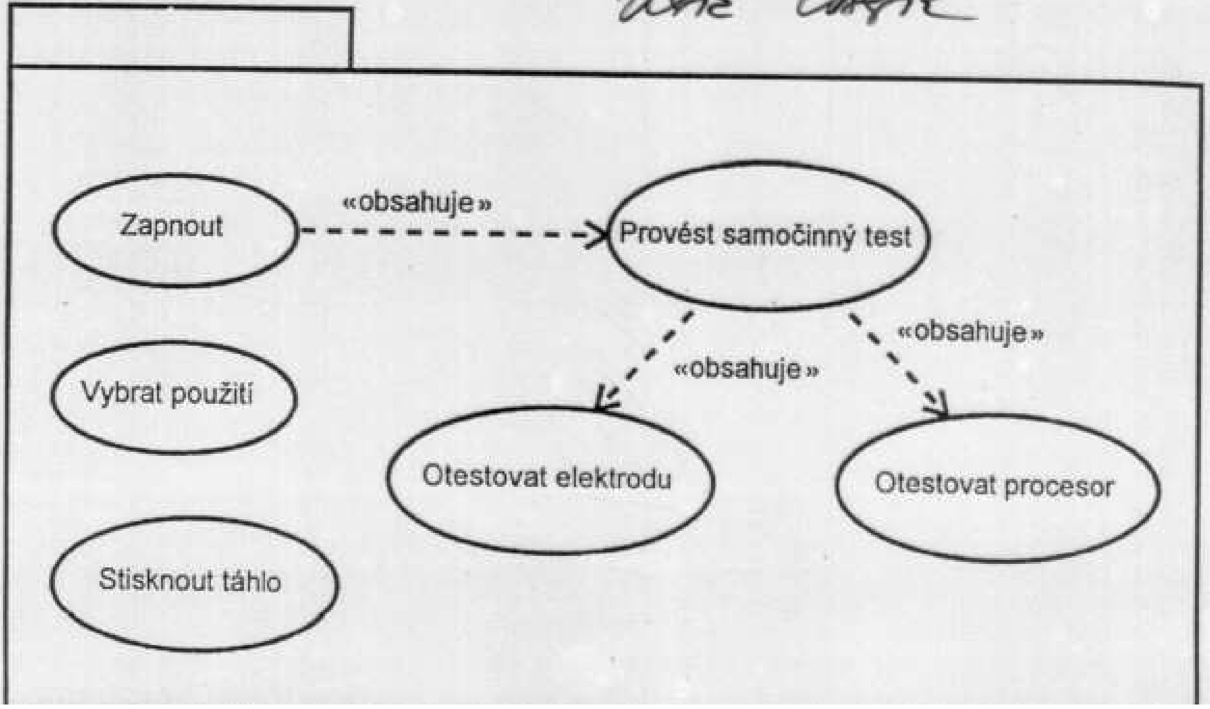
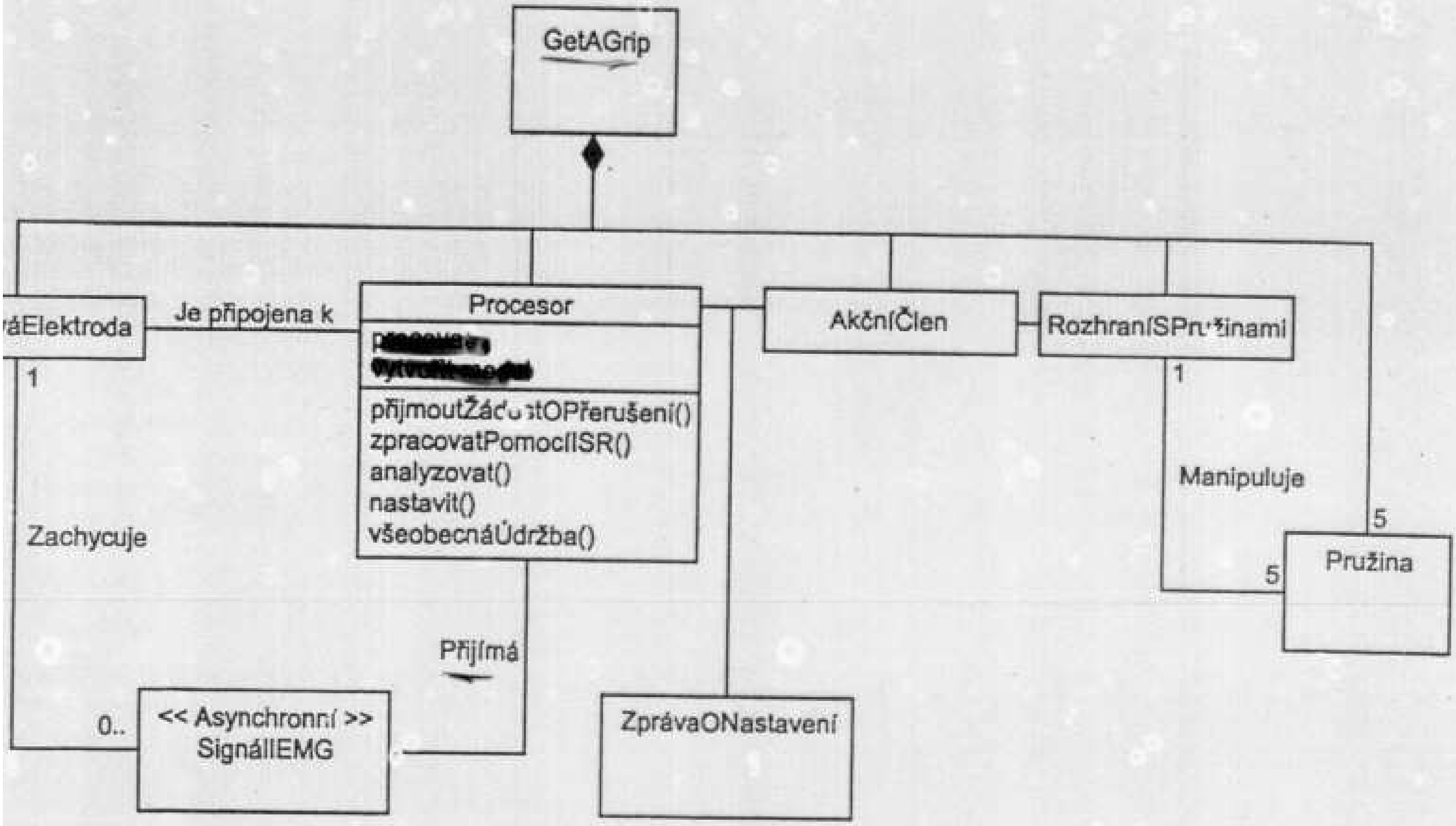
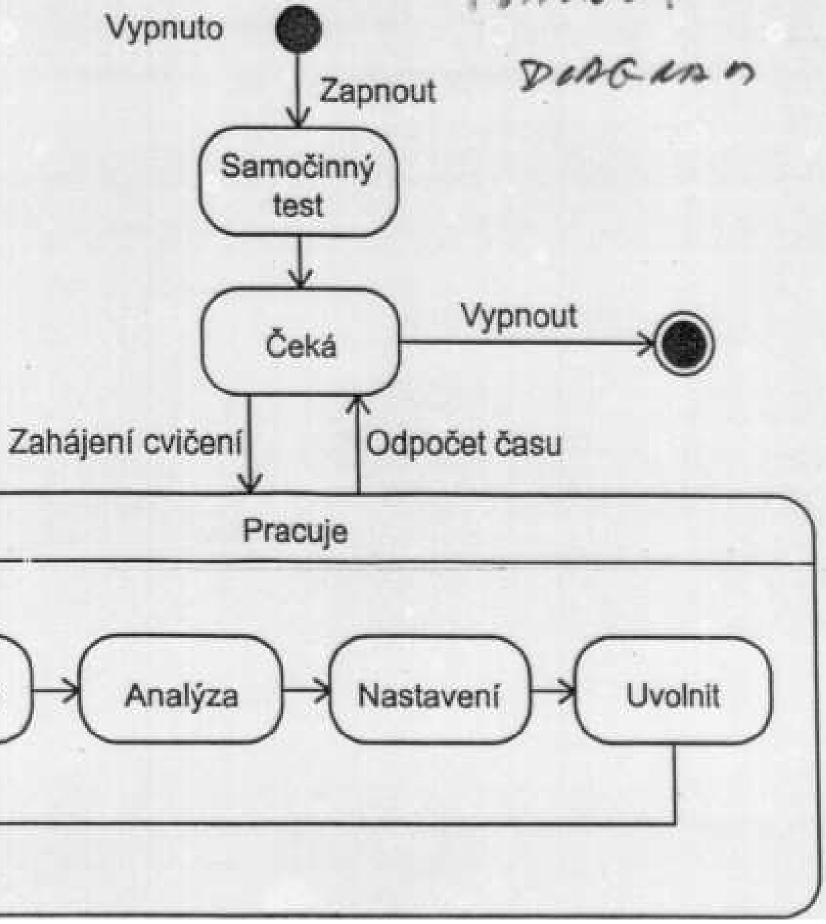


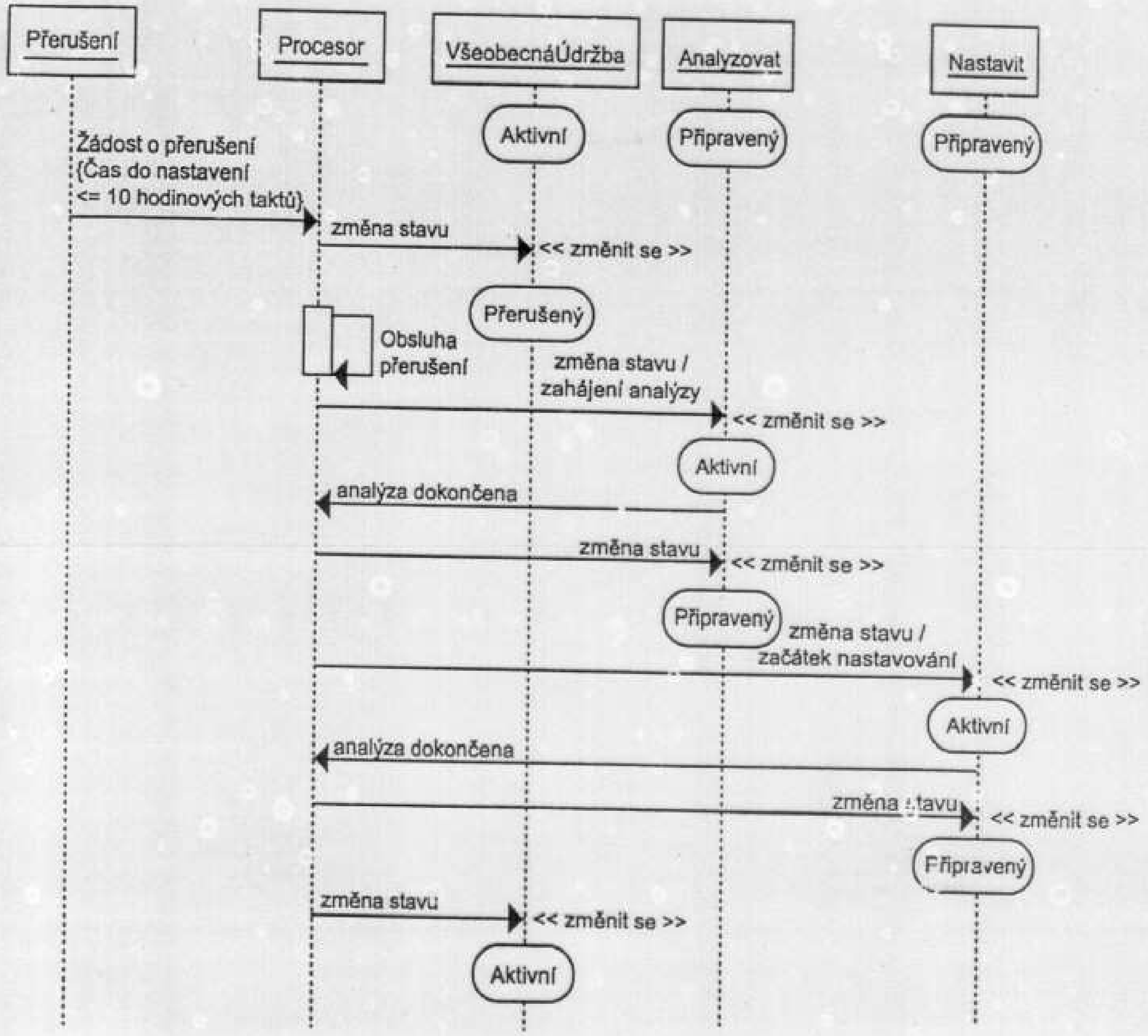
DIAGRAM TŘÍD



STAVOVÝ DIAGRAM



STISKNOUT TÁHLO – (SEKV. DIAG.) v DIAGRAM INTERAKCÍ



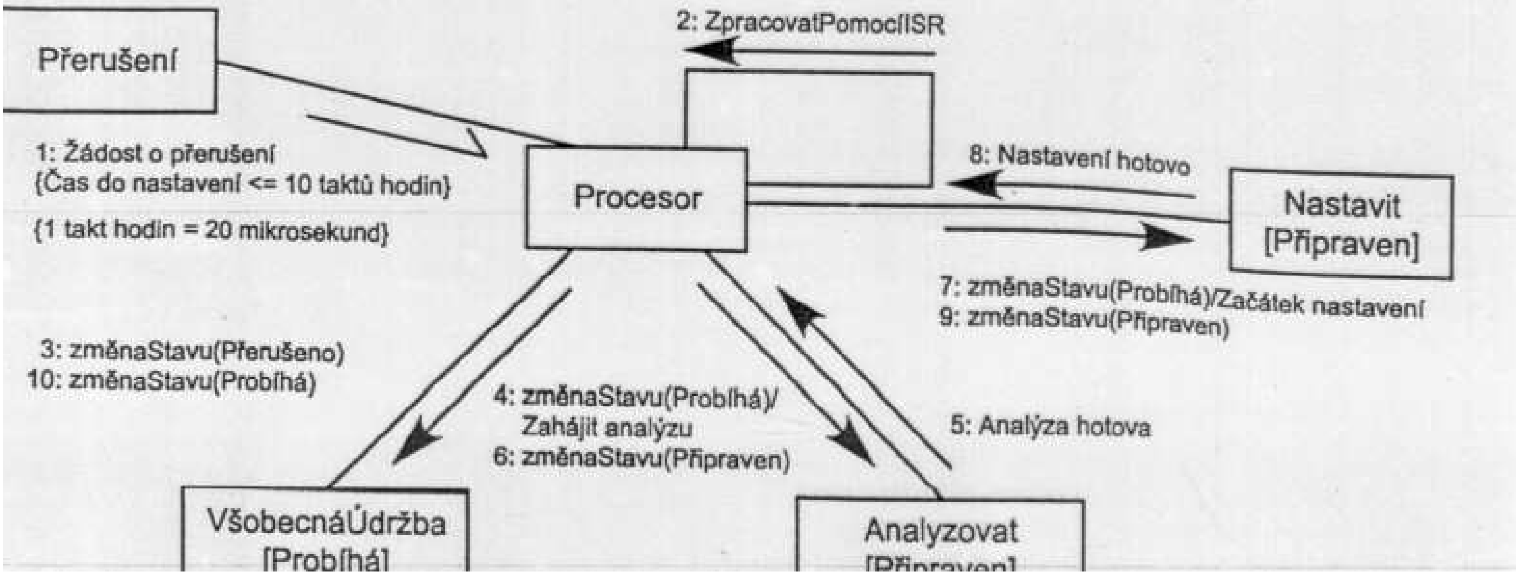


DIAGRAM SPOLUPRÁCE

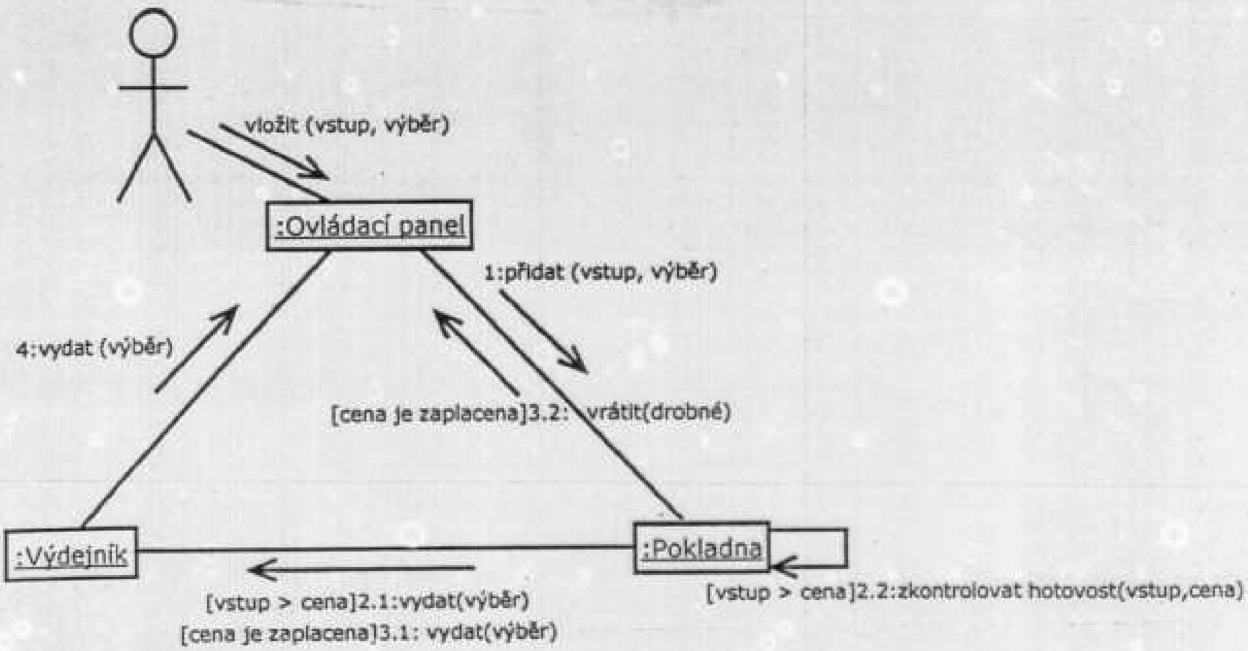

vložit (vstup, výběr)
:Ovládací panel
1:přidat (vstup, výběr)
4:vydat (výběr)
[cena je zaplacena]3.2: vrátit(drobné)
:Výdejník
:Pokladna
[vstup > cena]2.1:vydat(výběr)
[cena je zaplacena]3.1: vydat(výběr)
[vstup > cena]2.2:zkontrolovat hotovost(vstup,cena)

DIAGRAM SPOLUPRÁCE S DETAILY

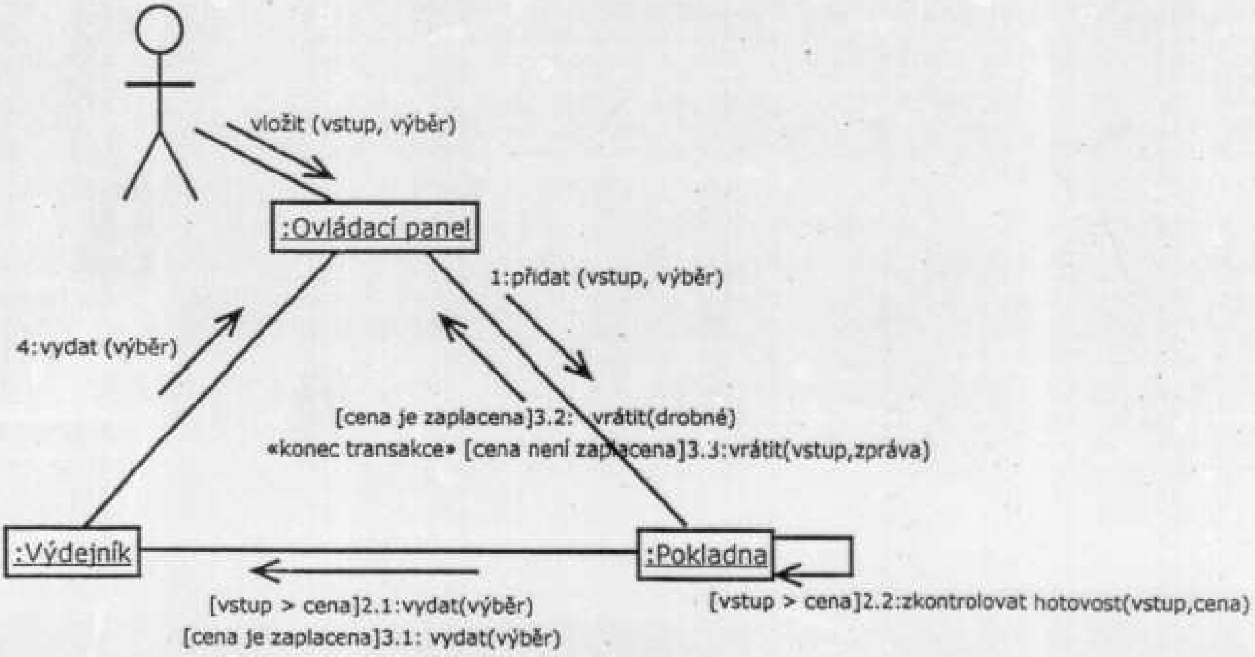

vložit (vstup, výběr)
:Ovládací panel
1:přidat (vstup, výběr)
4:vydat (výběr)
[cena je zaplacena]3.2: vrátit(drobné)
«konec transakce» [cena není zaplacena]3.3:vrátit(vstup,zpráva)
:Výdejník
:Pokladna
[vstup > cena]2.1:vydat(výběr)
[cena je zaplacena]3.1: vydat(výběr)
[vstup > cena]2.2:zkontrolovat hotovost(vstup,cena)

DIAGRAM INTERAKCÍ

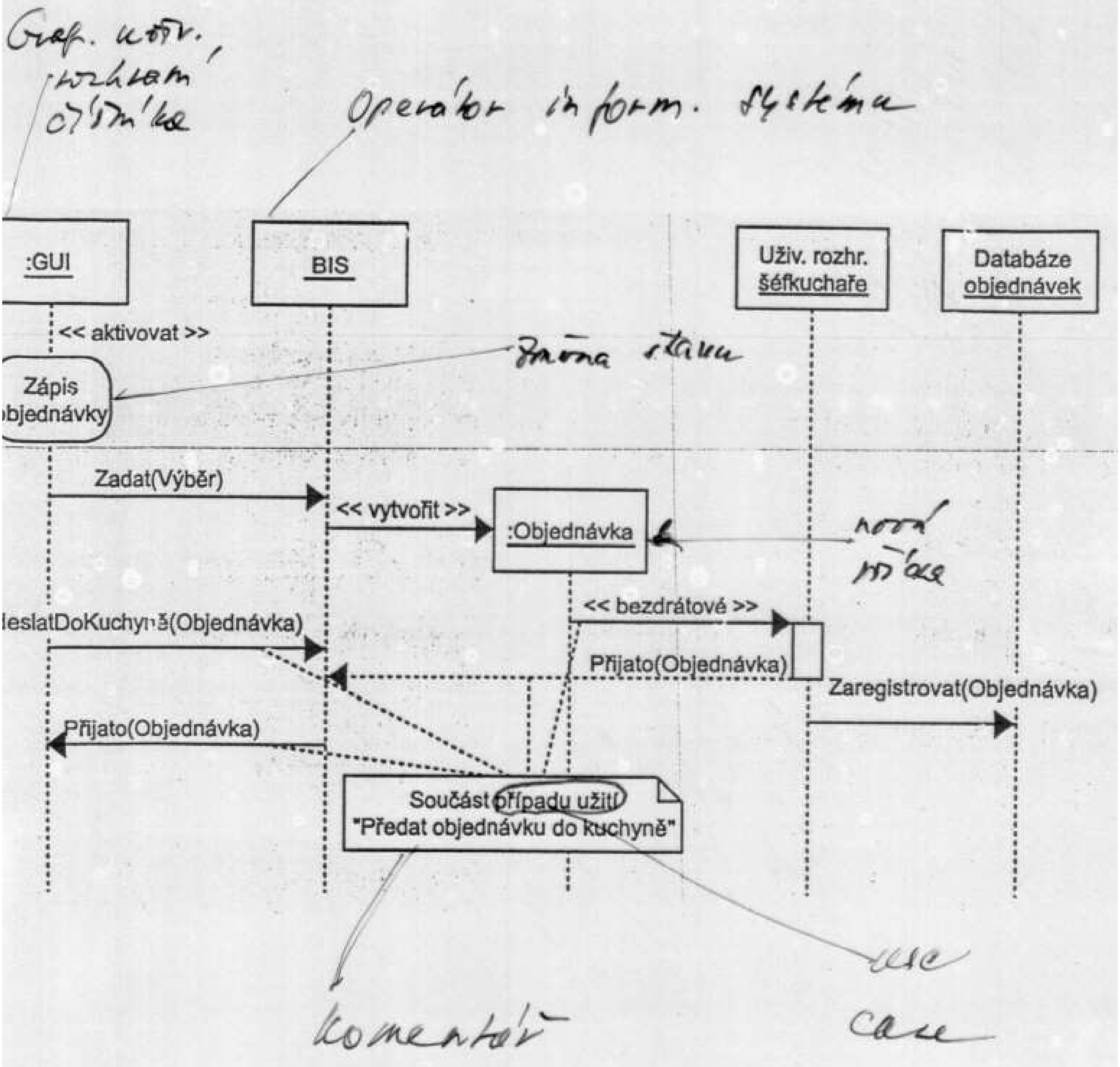


DIAGRAM ČINNOSTÍ

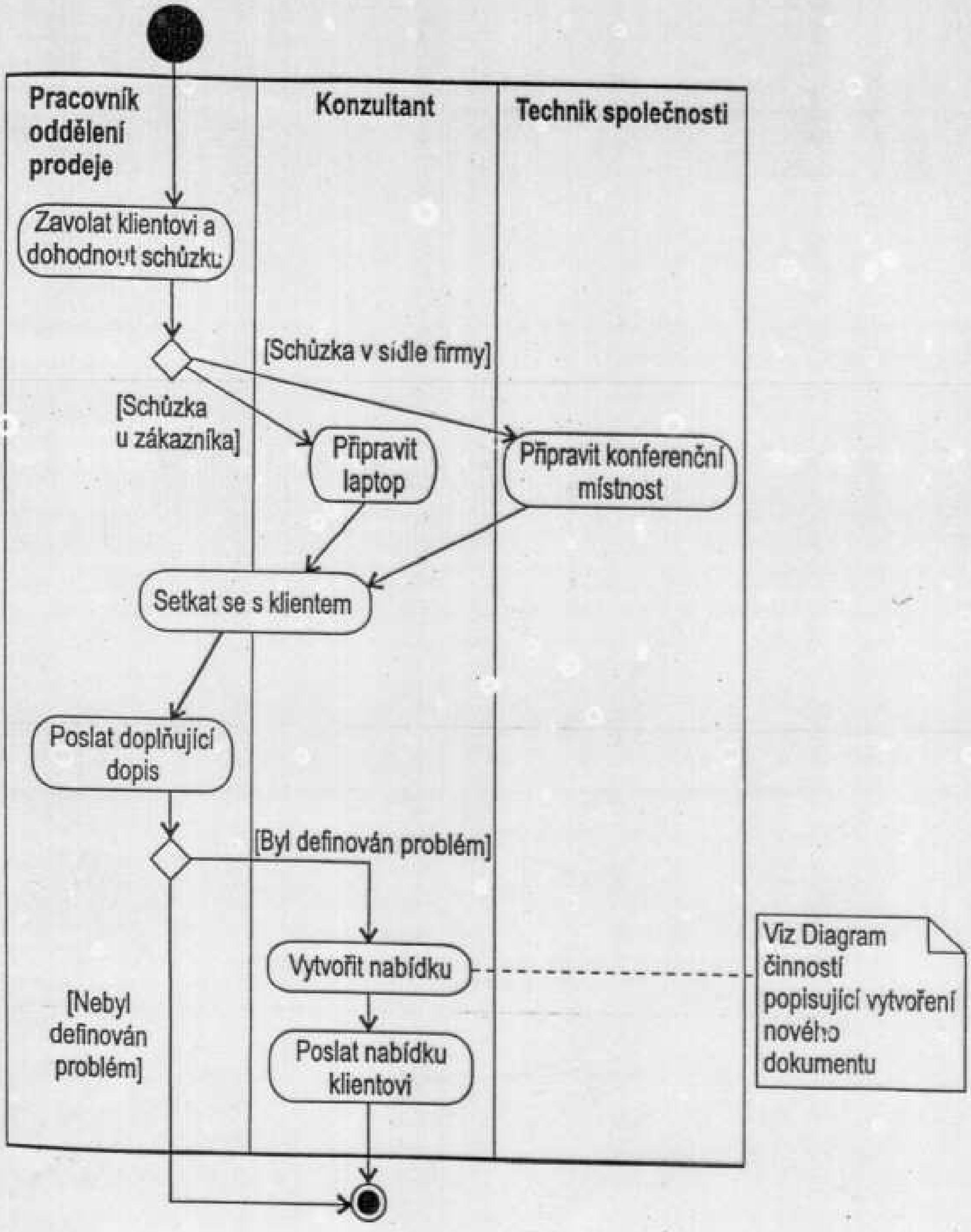
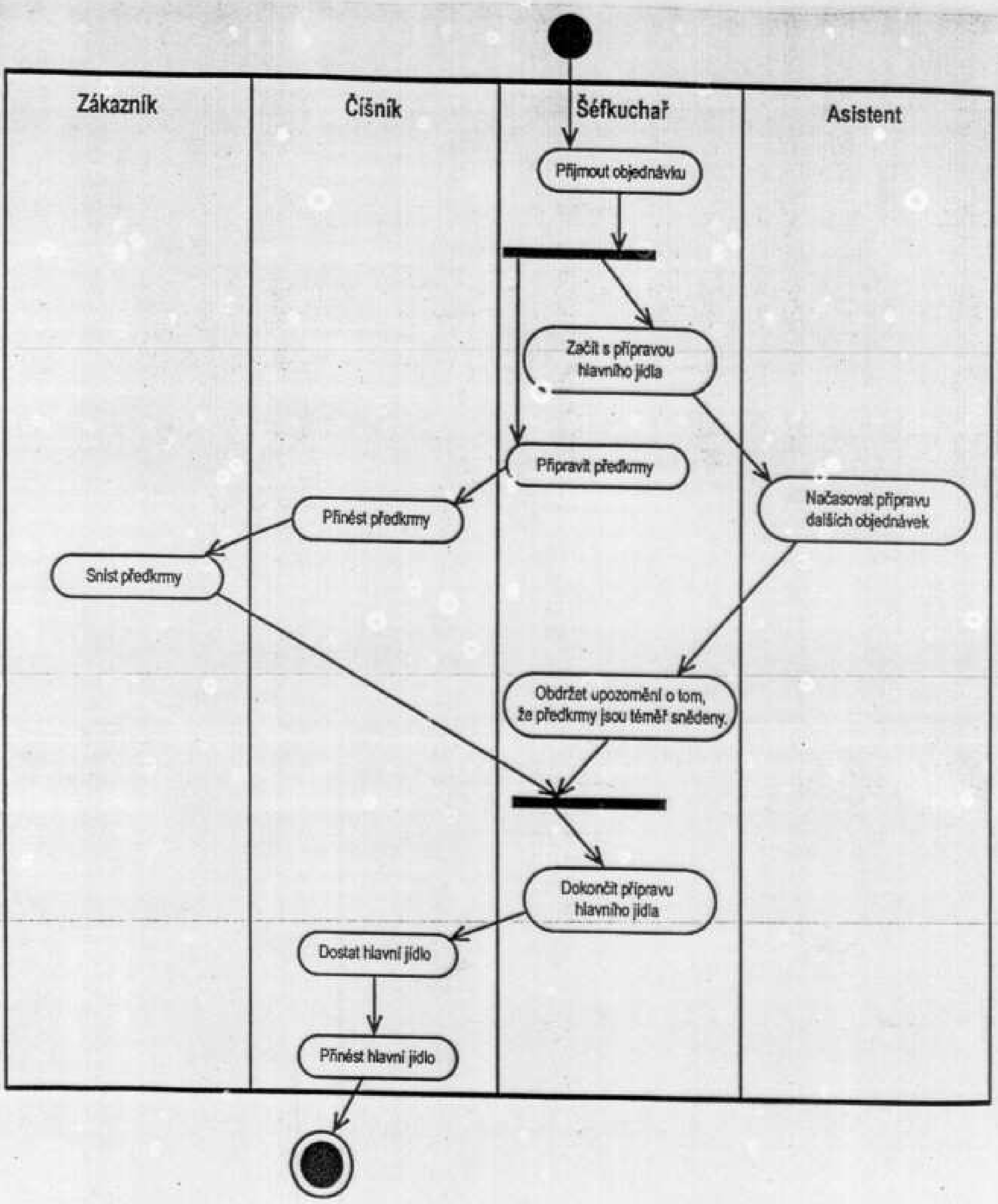
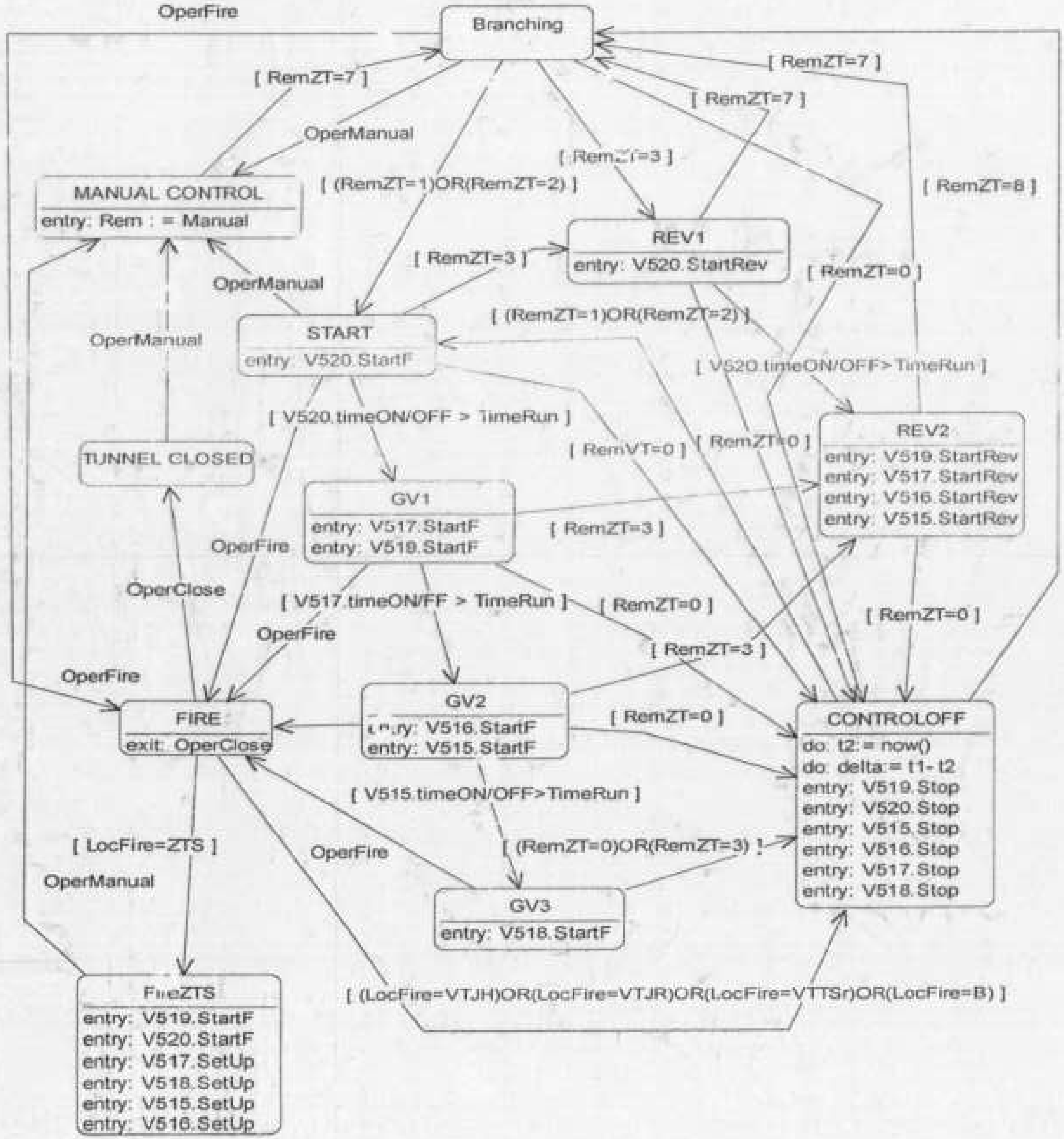


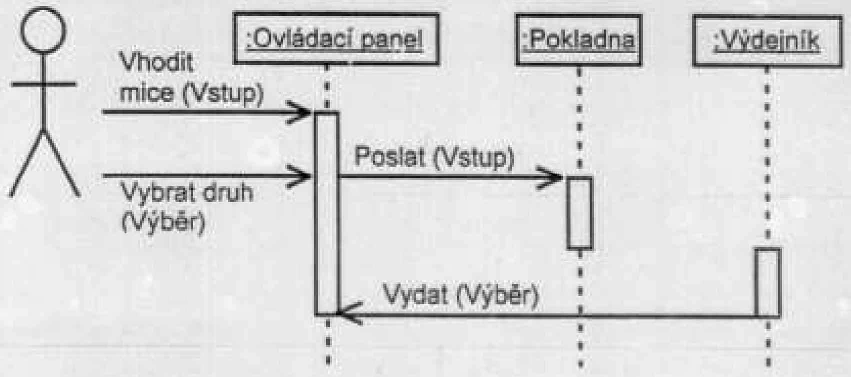
DIAGRAM ROLÍ – DETAIL


Zákazník
Číšník
Šéfkuchař
Asistent
Přijmout objednávku
Začít s přípravou hlavního jídla
Připravit předkrmy
Načasovat přípravu dalších objednávek
Přinést předkrmy
Sníst předkrmy
Obdržet upozornění o tom, že předkrmy jsou téměř snědeny.
Dokončit přípravu hlavního jídla
Dostat hlavní jídlo
Přinést hlavní jídlo

State Diagram of the class Tsf6



SEKVENČNÍ DIAGRAM



SEKVENČNÍ DIAGRAM S VĚTVENÍM

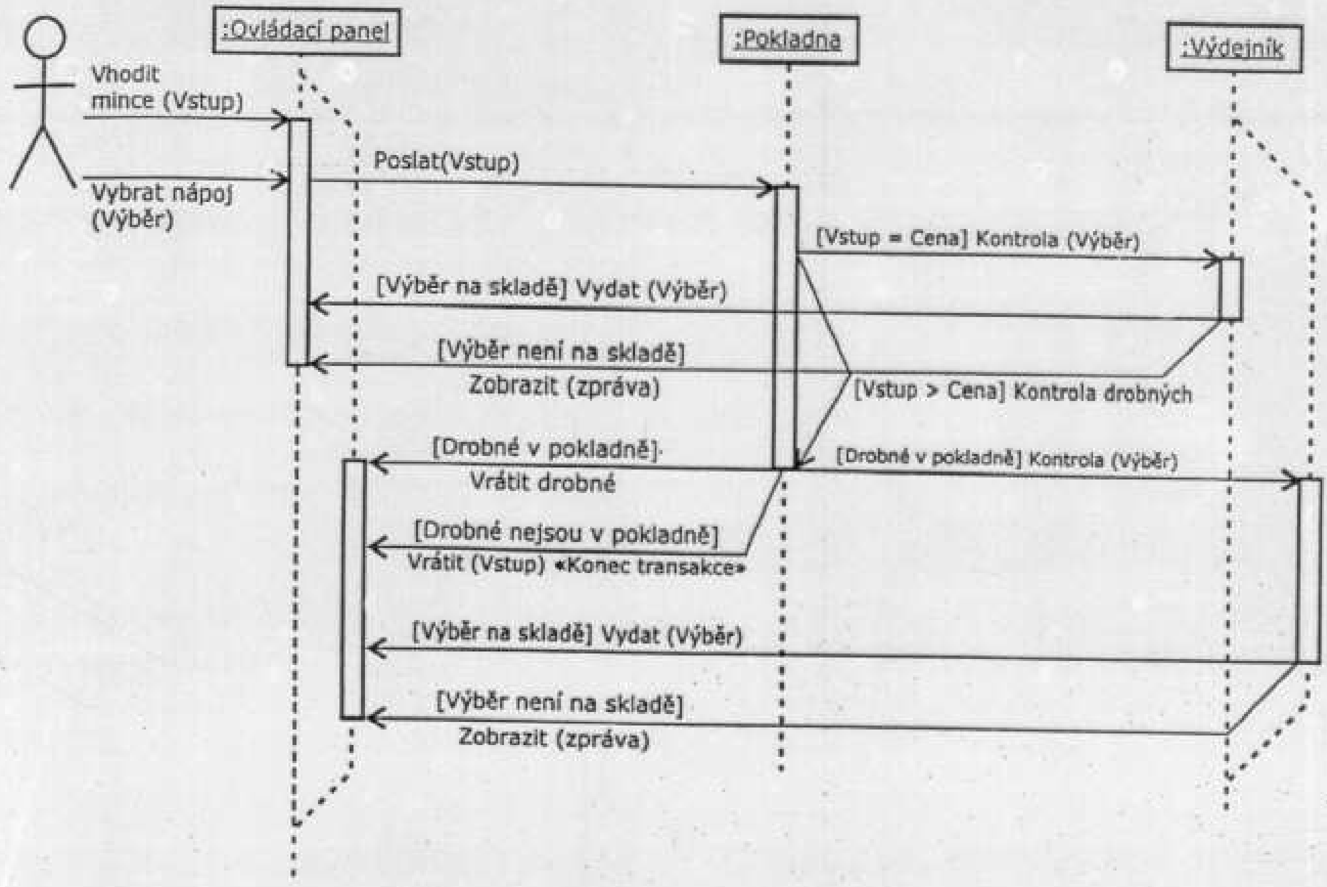
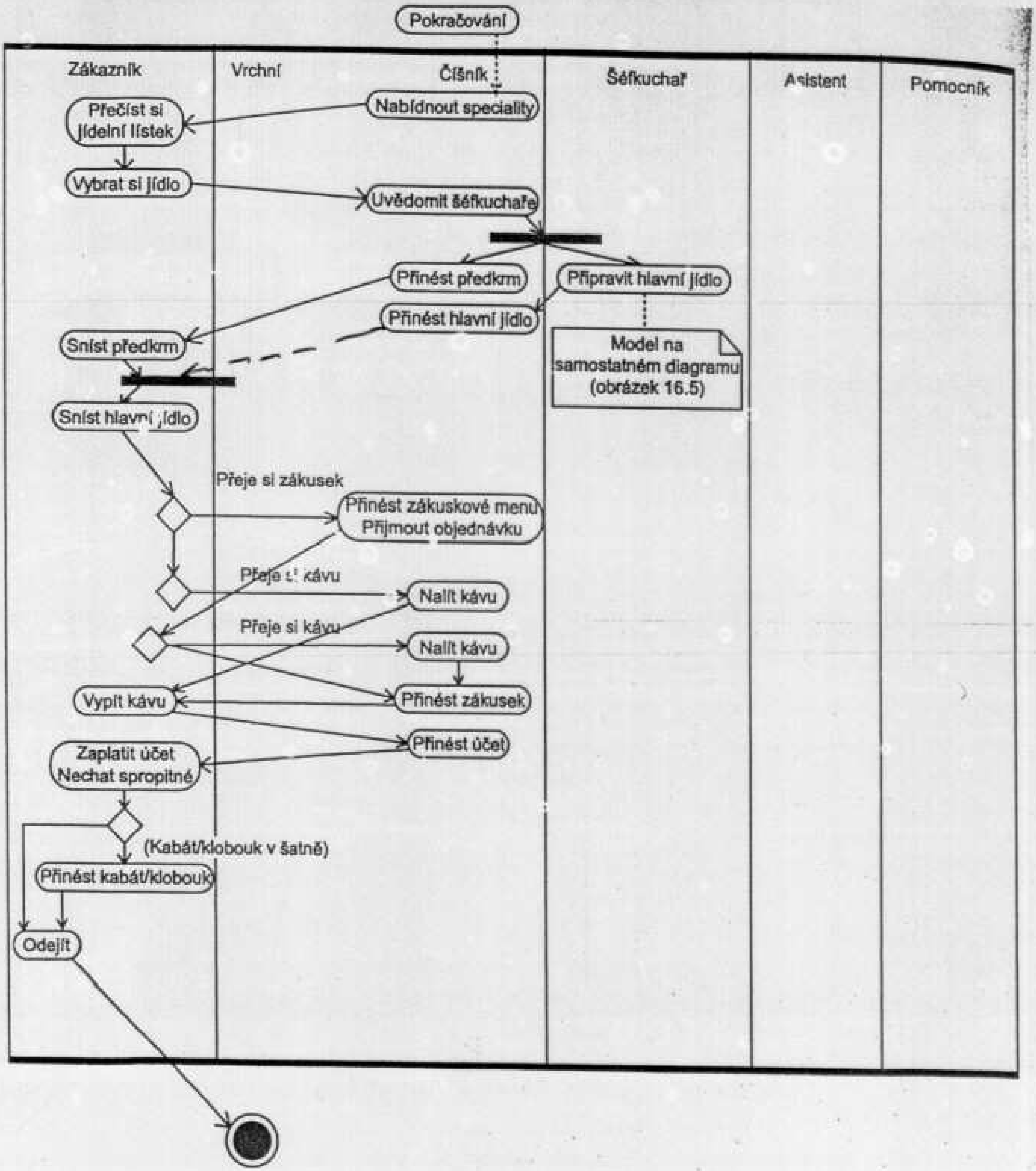
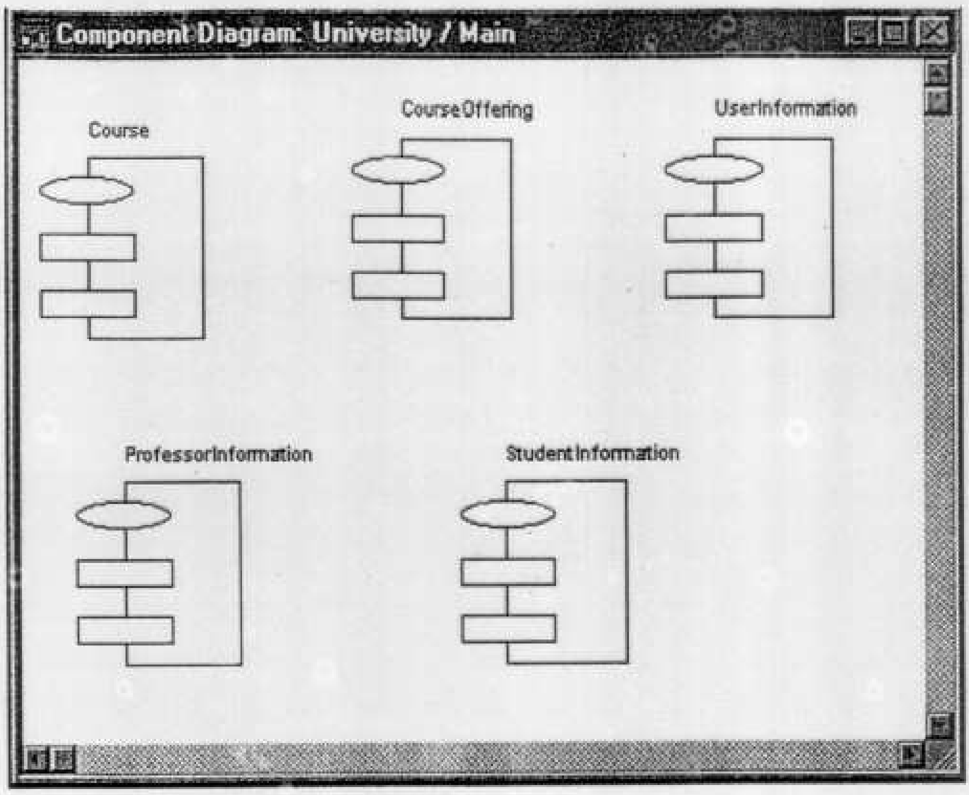
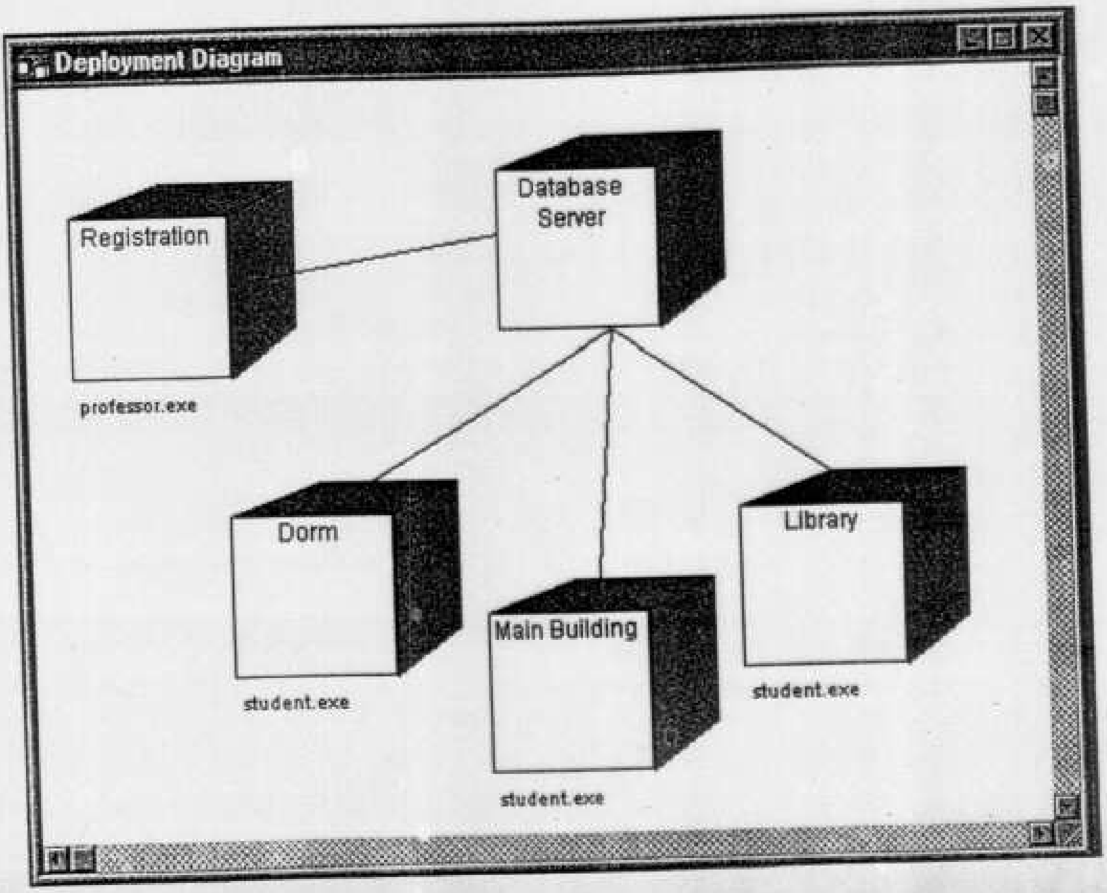

:Ovládací panel
:Pokladna
:Výdejník
Vhodit mince (Vstup)
Vybrat nápoj (Výběr)
Poslat(Vstup)
[Vstup = Cena] Kontrola (Výběr)
[Výběr na skladě] Vydat (Výběr)
[Výběr není na skladě] Zobrazit (zpráva)
[Vstup > Cena] Kontrola drobných
[Drobné v pokladně] Vrátit drobné
[Drobné v pokladně] Kontrola (Výběr)
[Drobné nejsou v pokladně] Vrátit (Vstup) «Konec transakce»
[Výběr na skladě] Vydat (Výběr)
[Výběr není na skladě] Zobrazit (zpráva)

DIAGRAM ROLÍ




Component Diagram: University / Main
Course
CourseOffering
UserInformation
ProfessorInformation
StudentInformation


Deployment Diagram
Registration
professor.exe
Database Server
Dorm
student.exe
Main Building
student.exe
Library
student.exe