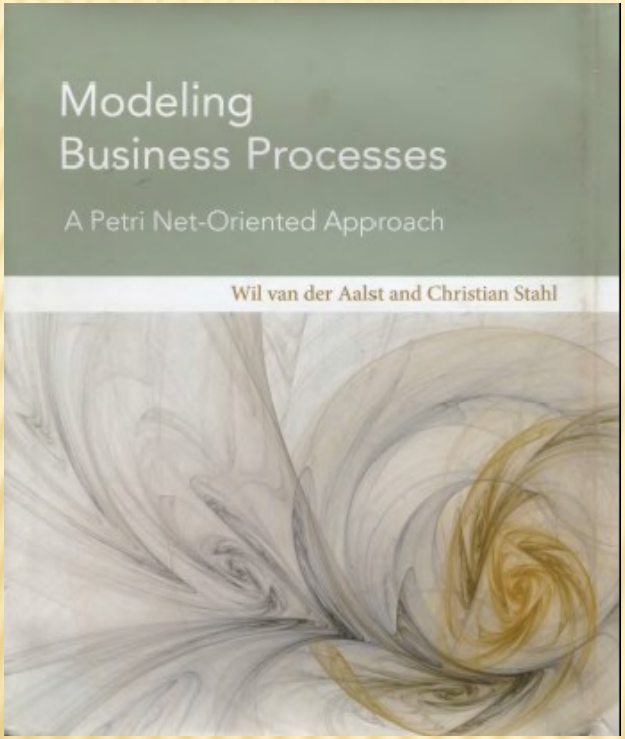
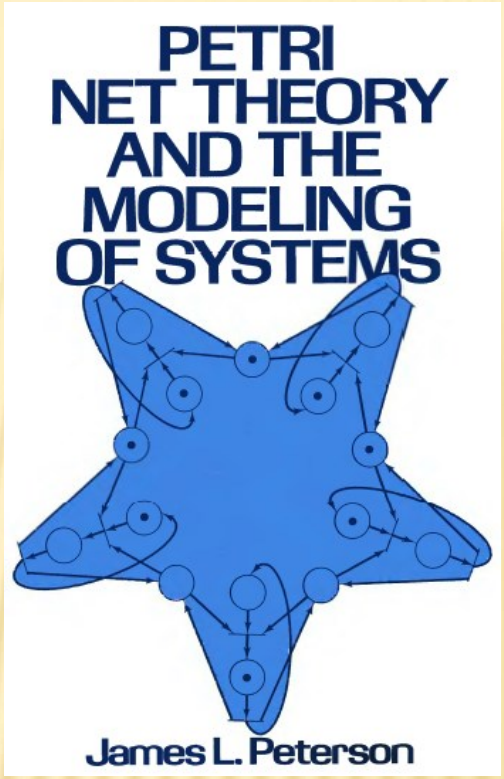
Curs 9 – Management Logistic
Introducere in PetriNet

Galatus Ramona

2019

REFERINTE BIBLIOGRAFICE





CONTINUT

- Definitia sistemelor informationale (Informational system – IS)
- Modelarea sistemelor de business prin sisteme informationale/ reatia dintre ele
- Clasificarea IS si pasii de dezvoltare a acestor sisteme informationale + Exemple
- Modele de descriere a IS – Petri Net
- Implementare:

http://www.pntool.ac.tuiasi.ro/download.php

https://www.youtube.com/watch?v=dUYRjsEUr1o

SISTEME INFORMATIONALE (SI)

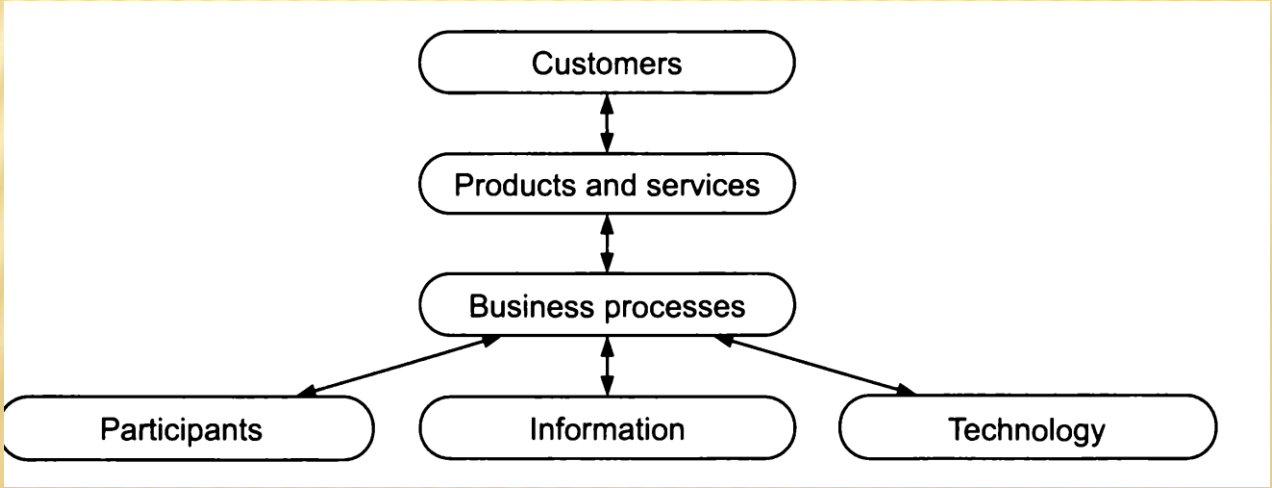
Information systems have become the backbone of most organizations. Banks could not process payments, governments could not collect taxes, hospitals could not treat patients, and supermarkets could not stock their shelves without the support of information systems. In almost every sector—education, finance, government, health care, manufacturing, and businesses large and small—information systems play a prominent role. Every day work, communication, information gathering, and decision making all rely on information technology (IT). When we visit a travel agency to book a trip, a collection of interconnected information systems is used for checking the availability of flights and hotels and for booking them. When we make an electronic payment, we interact with the bank's information system rather than with personnel of the bank. Modern supermarkets use IT to track the stock based on incoming shipments and the sales that are recorded at cash registers. Most companies and institutions rely heavily on their information systems. Organizations such as banks, online travel agencies, tax authorities, and electronic bookshops can be seen as IT companies given the central role of their information systems.

SI – MODELAREA SISTEMELOR DE BUSSINESS

- Se vor discuta aspect legate de modelarea proceselor de business
- Procesul de business – “flow of work” (planificarea activitatilor in cadrul unei organizatii)
- Management logistic – modelarea/planificarea activitatilor de logistica

SI – COMPONENTE

- Informatie – act de comunicare ->schimb mesaje intre agenti (participantii din sistem)-> ofera cunostinte fundamentale (knowledge)
- Cunostinte (knowledge, beliefs) – o stare de fapt care ofera un punct de vedere asupra unei situatii
- Obiecte – reprezinta cunostintele accumulate
- Exemple – client=agent



DEFINITIE

Definition 1.3 (Business process) A *business process* consists of a set of activities that is performed in an organizational and technical environment. These activities are coordinated to jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.

Definition 1.5 (Information system) An *information system* is a software system to capture, transmit, store, retrieve, manipulate, or display information, thereby supporting people, organizations, or other software systems.

CLASIFICARE

- 1. Sistem informational personal (ex. Fisa pacient)
- 2. Sistem informational al unei organizatii (Enterprise information system)
  - EDI – ELECTRONIC DATA INTERCHANGE
  - ERP – ENTREPRISE RESOURCE PLANING
  - **Exemple**: workflow management system, data warehouse, process engineering/manufacturing system, procurement system, sales and marketing system, delivery system, finance system, product design, workflow management system, data warehouse, business intelligence system (analizarea performantei rularii proceselor de business)

BUSINESS INTELLIGENCE SYSTEM

- Business performance management (indicatori de performanta)
- Querying and reporting
- Data mining: classification, clustering, regression, association rule learning
- Process mining
- Timp de viata a unui SI in cadrul unei industrii/organizatii – exemplu SAP tool:

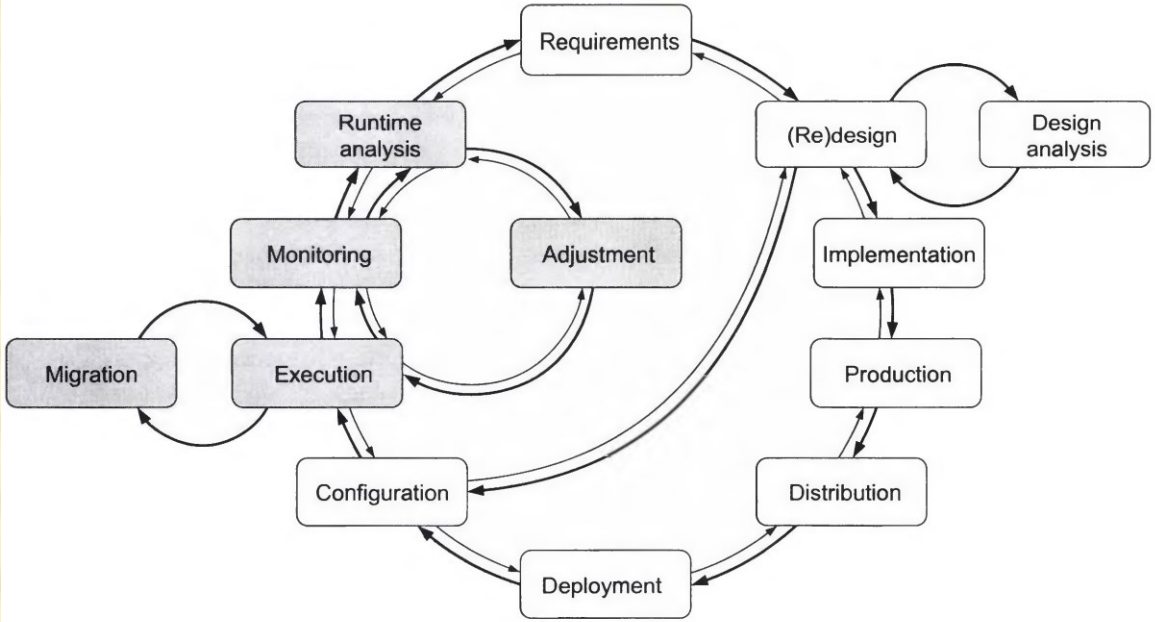


Figure 1.2
The life cycle model of an enterprise information system.

We distinguish the following eleven phases for customized information systems: *requirements phase, design phase, design analysis phase, implementation phase, production phase, distribution phase, deployment phase, configuration phase, execution phase, monitoring phase,* and *runtime analysis phase*. Not all of these phases are relevant for all information systems; for example, production, distribution, and deployment phases are only relevant in the case of generic (i.e., made-to-stock) information systems, such as ERP systems, Microsoft Office tools, or database management systems.

STUDIU DE CAZ – MODELARE ATM

1. Requirements – strangerea de informatii necesare pentru modelare

Example 1.8 The requirements phase in the development of a new (simplified) ATM leads to the following requirements. The ATM should allow its clients to query their current account balances and to withdraw money. If clients want to withdraw money, then the ATM should offer them several amounts, but it should also allow them to choose an amount of money. There are several restrictions. For example, the amount of money clients withdraw should be less than a maximum amount (e.g., 500 euros for each day), and it should not lower the client's account balance below a predefined lower bound. Furthermore, if clients just query their current account balance, then their account balance should not change.

2. Modelarea sistemului-> functionalitati

2. DESIGN PHASE

Example 1.9 In the example of the ATM, the designer constructs a functional design model of the ATM on the basis of the previously developed domain model. The functional design model can be an algebraic specification of the static information (e.g., querying the current account balance returns an account balance in euros) and a business process model describing the order of activities (e.g., clients choose to withdraw money, next they can choose between a standard amount or a customized amount, and so on). In addition, the functional design model can contain a prototype showing the user the possible interactions with the ATM. With this model, the user and designer can discuss all open issues of the final design of the ATM.

In the next step, the designer develops an implementation model based on the functional design model. This model may contain detailed information about how the database of the bank must be queried, how the chosen security mechanisms must be implemented, and how the interplay of the information system with the hardware of the ATM must be implemented. The implementation model serves as a basis for discussion between the designer and the software developer to identify the way in which the ATM should be implemented.

In the ATM example, the mediator role of the designer and the benefit of the two models becomes clear. The designer uses the functional design model to communicate with the user and the implementation model to communicate with the software developers.

3. DESIGN ANALYSIS

- Scenarii de test

Example 1.10 Using the ATM example, we can specify a scenario of a client who first queries an account balance and afterward withdraws 100 euros. By using simulation, we can execute this scenario on the model and check whether this model behaves as expected. Simulation also allows performance analysis; for example, we could check whether the database system can retrieve the current account balance within a certain time interval. It would be important to verify that clients cannot crash the ATM.

Implementation Phase The fourth phase in the life cycle model is the *implementation phase*. In this phase, the information system is constructed. Because an information system is a software system, construction means either programming the entire functionality from scratch or extending or reimplementing existing functionality. Nowadays, software projects increasingly develop generated code. Development tools, such as Eclipse, may generate template code to create a graphical user interface, for instance. The programmer can later modify and refine this generated code. This significantly increases a programmer's productivity.

Production Phase The fifth phase is the *production phase*, in which software of an information system in prepared for distribution. Unlike classical manufacturing processes, it is relatively easy to produce software, because this boils down to copying and downloading. For widely used standard products, such as database management systems and the Microsoft Office tools, however, the production of manuals, CDs, and so forth may be nontrivial. For product software, licensing issues may also require effort.

Distribution Phase In the case of mass production, there is a sixth phase, the *distribution phase*. The goal of this phase is to make the information system available to its future users. The marketing for the information system is also a part of this phase. The production and distribution phases do not apply to customized information systems.

Deployment Phase In the *deployment phase*, the information system is installed in its target environment, and the users of the information system are trained to use it or to work with it. For example, in the case of a health care system in a hospital, professionals must be trained. Training is important in other domains as well, because information systems, such as ERP systems and database management systems, provide a multitude of functionality. The deployment phase is the seventh phase in the life cycle model.

Configuration Phase Many organizations do not implement their information systems from scratch but instead buy standard software, which is often referred to as *commercial off-the-shelf* software or *product* software. In this case, the information system needs to be configured and customized to the organization and its business processes. Even when organizations develop their own software, there is often the need for configuration. This is the subject of the eighth phase in the life cycle model, the *configuration phase*.

Execution Phase After the deployment and configuration, organizations can finally run their information systems. In an ideal world, this *execution phase*—the ninth phase in the life cycle model—would be the final phase of the development process, with maintenance consisting of the organization keeping the data up to date and making backups. Because of its complexity, however, it is unlikely that an information system meets all requirements and performs in a way it is expected at the start of this phase. Moreover, the environment of the information system is changing over time. To simplify error detection and to get insight into what functionality is actually used (and also how it is used), information systems log an enormous number of events. These event logs provide detailed information about the activities that have been executed. Event logs play an important role in the successive phases of the life cycle model.

Monitoring Phase In the tenth phase, the *monitoring phase*, organizations extract real-time information about how their information systems perform. Monitoring provides information on the current state of each business process instance and on the performance of the previously executed activities. In a way, the monitoring phase is a simulation of the running business processes in practice. The extracted information can be compared with the domain model (i.e., the requirements) and the functional design model. Unlike the process shown in figure 1.2, the execution phase and the monitoring phase typically run in parallel. The monitoring phase is using data from the execution phase, but it can also influence execution through the adjustment phase (see section 1.3.4).

Runtime Analysis Phase Monitoring is performed while the information system is running, but it is not intended to change information systems or redesign business processes. Monitoring provides relatively simple types of diagnostic information. More advanced analysis techniques are possible and are performed in the *runtime analysis phase*.

In contrast to the design analysis phase in which information system models are analyzed, the runtime analysis phase analyzes whether the implemented information system conforms to its specification. Event logs play an important role in this phase.

STARILE UNUI AUTOMAT ATM

S = {idle, card, pin, balance, money,
offer, choice, payout, violation, output_card}.

The following set of transitions is possible:

TR = {(idle, card), (card, pin), (pin, balance), (pin, money), (money, offer),
(money, choice), (offer, payout), (offer, violation), (choice, payout),
(choice, violation), (violation, money), (violation, output_card),
(balance, output_card), (payout, output_card), (output_card, idle)}.

The ATM is initially in state idle. A client then inserts a bank card (yielding state card) and keys a pin (pin). Next, a client can either query an account balance yielding state balance or withdraw money yielding state money. In state money, a client can either choose an amount of money (choice) or select an offered amount of money (offer). If the chosen amount of money is not too high, the money is paid out (payout), and the ATM returns the card to the client (output_card). Otherwise, the ATM enters state violation, from which the menu can be reached (state money), or the client asks the ATM to return the card (output_card). From state output_card, the ATM moves to state idle from which it can serve the next client.

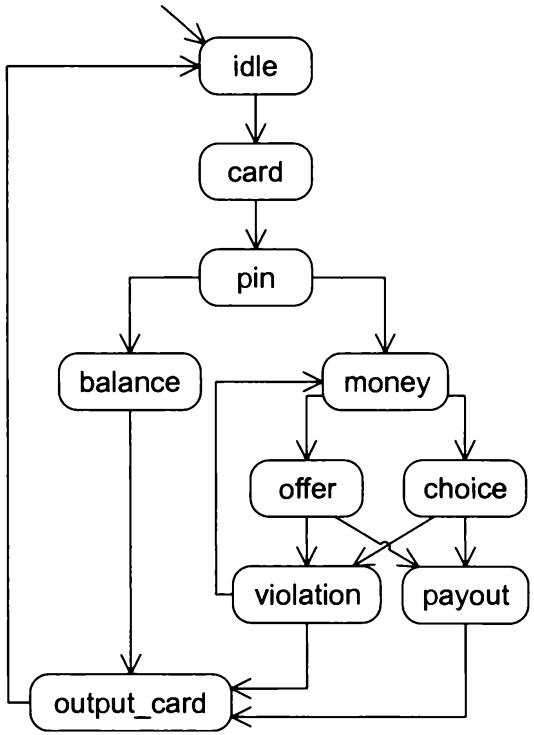
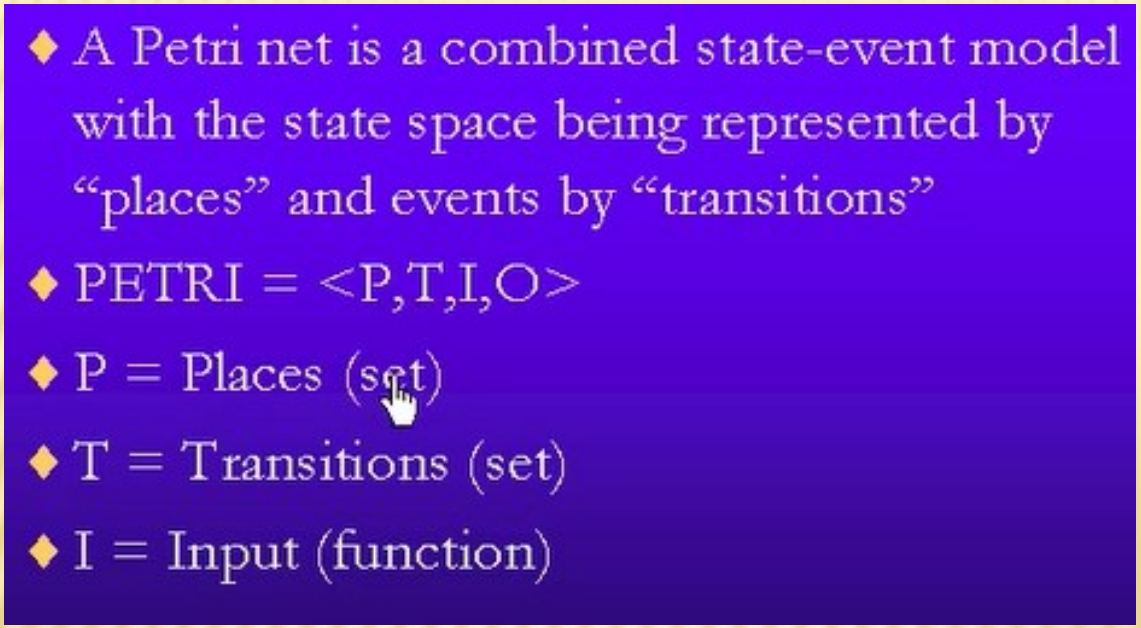


Figure 1.3
A state-transition diagram of the ATM.

SISTEM MATHEMATIC DE DESCRIERE A SI

Definitie



- Token-a thing serving as a visible or tangible representation of a fact, quality, feeling, etc..

TRANZITIILE POSIBILE

Definition 1.15 (Transition system) A *transition system* is a triple (S, TR, s_0) where S is a finite state space, $TR \subseteq S \times S$ is a transition relation containing all possible state changes, and $s_0 \in S$ is the initial state.

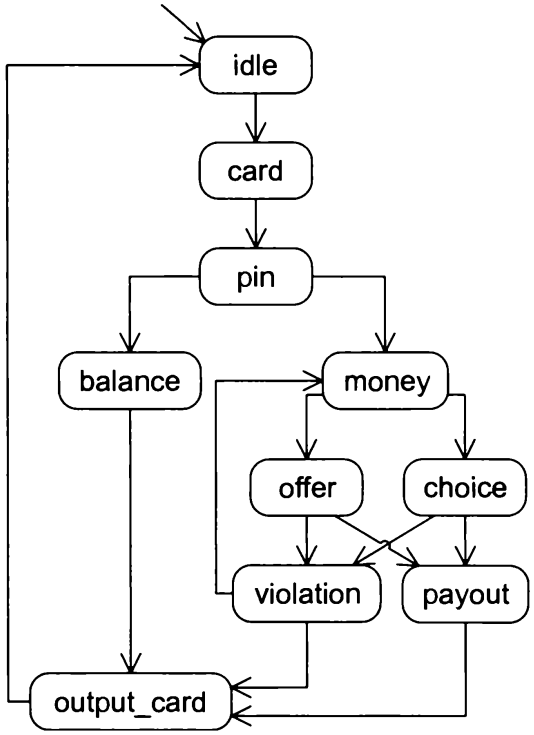


Figure 1.3
A state-transition diagram of the ATM.

Definition 1.17 (Reachable state) A *reachable state* is a state that the system can reach from the initial state after zero or more transitions.

Question 1.18 The initial state of the ATM in example 1.14 is state idle. Are all states reachable?

We can determine reachability by following the arrows in the state-transition diagram in figure 1.3. State idle can be reached after zero transitions. From state idle we can go to state card, next to state pin, to state balance, to state output_card, and to state idle again.

A sequence of states reached by following the arrows in figure 1.3 is a *transition sequence*. For example,

⟨idle, card, pin, money, choice, payout, output_card, idle⟩

is a transition sequence that represents withdrawing money by selecting a specific amount of money. The transition sequence for querying the account balance is:

⟨idle, card, pin, balance, output_card, idle⟩.

Example 1.19 The state-transition diagram in figure 1.3 models the functional design of an ATM. To simulate this model, we need to add time information and probabilities. As an example, reading the bank card in state idle may take two seconds, checking the pin code in state card five seconds, and returning the bank card to the client in state payout three seconds. In state pin, the client may choose in 70% of the cases withdrawal of money and in 30% an account balance. After adding probabilities to all transitions, we can simulate this model and, for example, calculate the probability that the ATM enters state violation.

EXEMPLU – PROGRAMARE INTERSECTIE

Exercise 1.14 A Dutch traffic light is an example of a system with three possible states: R (red), G (green), and O (orange). Model a T-junction with three traffic lights (see figure 1.6) as a transition system and draw the state-transition diagram. The traffic lights are programmed in such a way that at least two lights are red at the same time—that is, for at most one direction of the traffic, the traffic light can be green or orange. (Hint: Represent each state by a combination of three colors.)

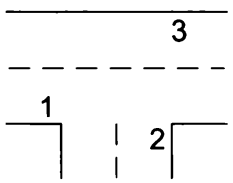


Figure 1.6
A T-junction with three traffic lights.

The goal is to program the traffic light system such that crossing cars coming from different directions cannot have a green light at the same time. Furthermore, if at any point lights turn orange, they must first turn red before other lights can change. Model this system as a transition system and draw the state-transition diagram.

Exercise 1.18 Figure 1.8 depicts a simplified remote control of a TV. It has six buttons to choose a channel, one button to regulate the volume, one button to mute the sound (and to turn it on again), and one button to switch the TV on or off. We consider the remote control and the corresponding TV as a system and assume that the possible states of this system are controlled by the buttons on the remote control.

1. Is this system a discrete dynamic system?
2. Describe all possible states of this system.

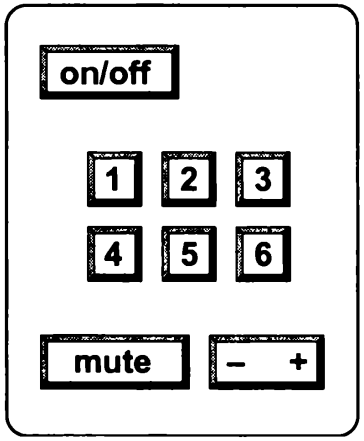


Figure 1.8
The remote control of a TV.

3. The transition relation is too large to be depicted as a state-transition diagram. Give examples of possible and impossible transitions. Pay attention to switching the TV on and using the volume button in combination with the mute button.

Exercise 1.19 Consider a transition system with state space

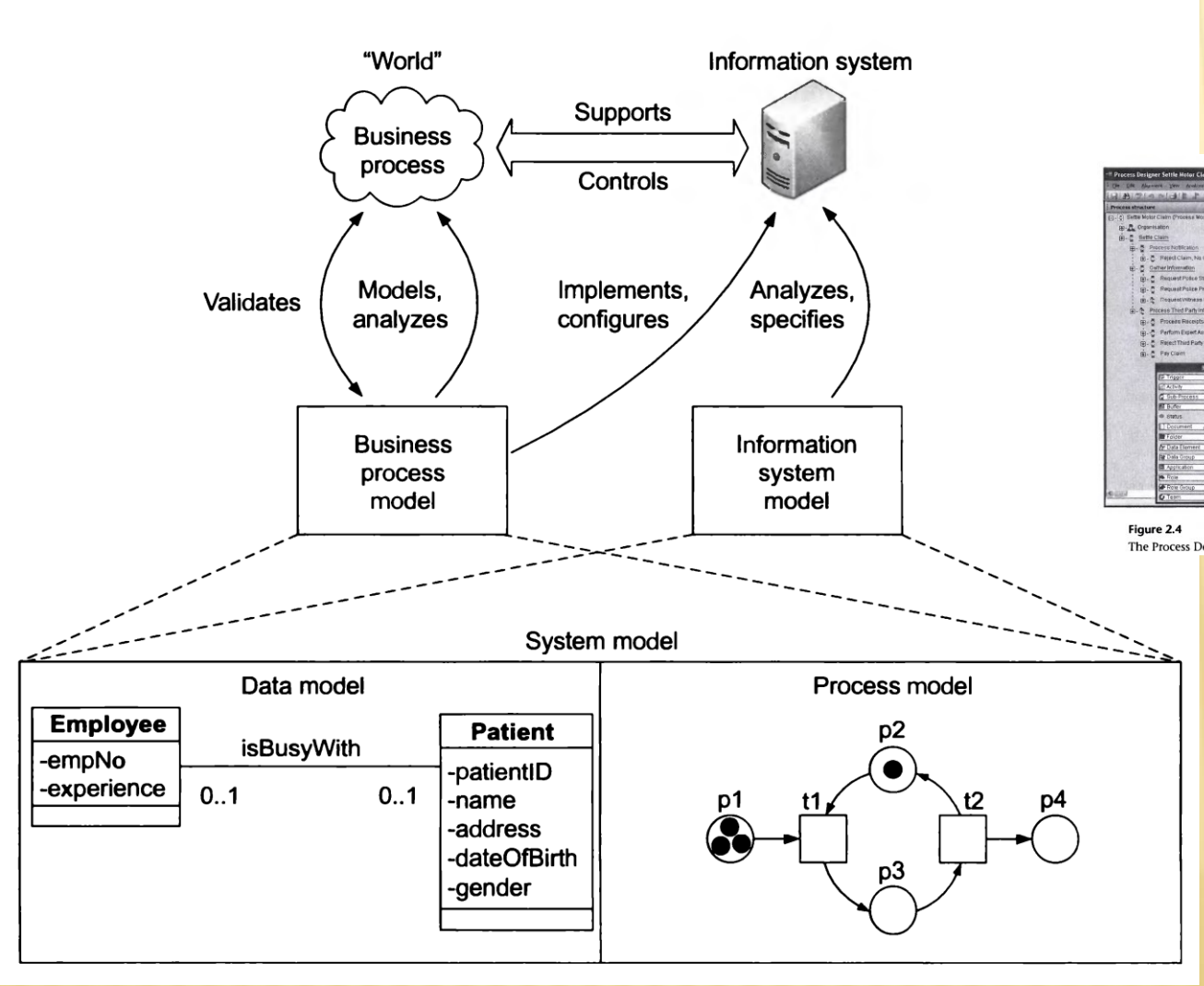
$\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

and with transition relation

$\{ (0, 1), (1, 2), (2, 3), (3, 4), (3, 5), (5, 0), (5, 4), (4, 4), (6, 7), (7, 6), (7, 8) \}$.

1. Draw the state-transition diagram.
2. Which states are reachable from the initial state 0?
3. List three transition sequences that start in state 0.
4. Does the system have a terminal state?

RELATIA PROCESE BUSINESS- IS-MODELE ASOCIATE + TOOLS



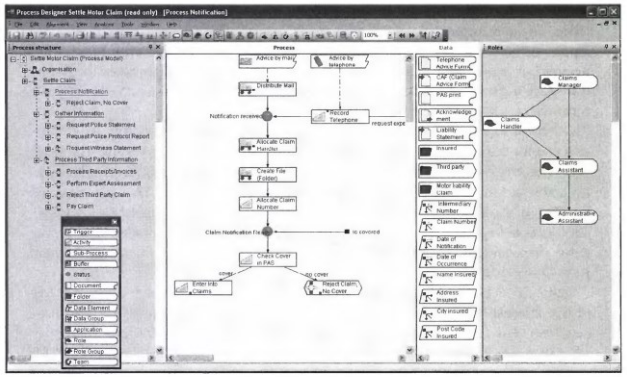


Figure 2.4
The Process Designer in BPM|one.

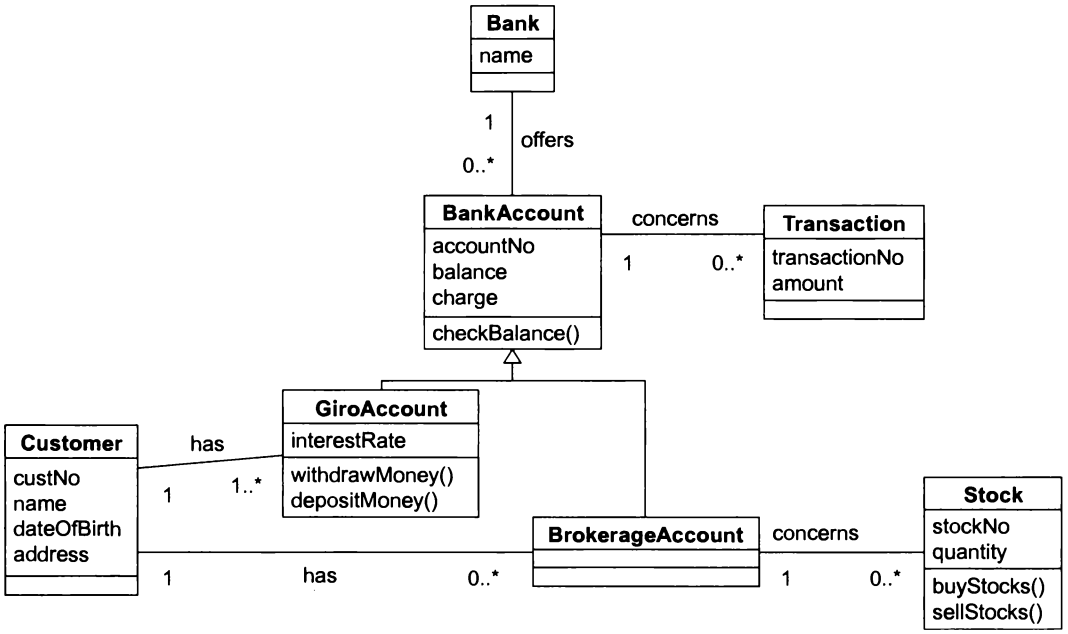


Figure 2.10
UML class diagram of the bank account example.

Exercise 2.1 Determine the UML class diagram for the following model of booking a train. A train has a number and several coaches. For each coach, the number of the coach and the number of seats is determined. A train may make several trips. Each trip has a departure day and time and an arrival day and time. It further has a departure station and an arrival station. A station is identified by a name and a postal code. Customers can make a reservation for a train trip. A seat reservation specifies a coach number and a seat number.

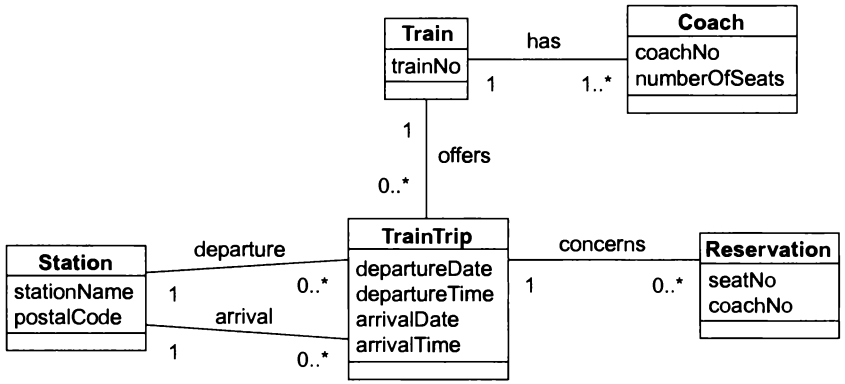


Figure 2.11
UML class diagram of exercise 2.1.

3.1.1 A First Example of a Petri Net: Modeling an Elevator

We discuss Petri nets by using the elevator example of exercise 1.6. Figure 3.1 depicts a Petri net modeling the behavior of the elevator.

The elevator moves between five floors and can stop at any of these floors. We model each floor as a *place*. Graphically, a circle represents a place. There are five places in figure 3.1: floor0, floor1, floor2, floor3, and floor4. Place floor0 models the ground floor, place floor1 the first floor, and the other three places model the corresponding remaining floors.

At each floor, the elevator can go up or go down. The Petri net in figure 3.1 models each move of the elevator by a *transition*, which is represented by a square. The Petri

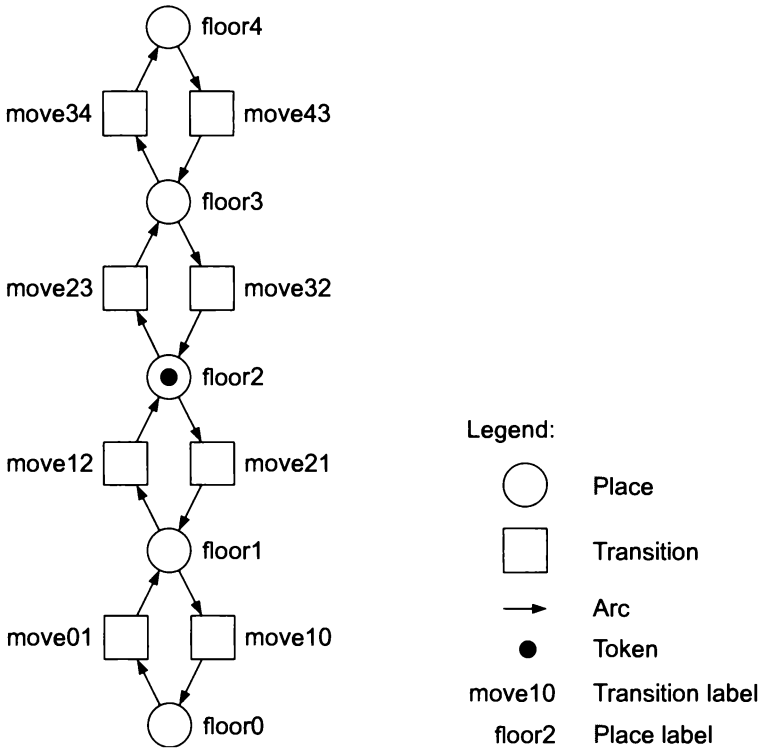


Figure 3.1
The elevator modeled as a Petri net.

Figure 3.3 depicts a first attempt at a Petri net modeling the business process of the X-ray machine. Each token in place wait represents a patient in the queue at the X-ray machine. A token in place before represents a patient in the X-ray room before the X-ray is made. After the X-ray has been made, there is a token in place after. Each token in place gone represents a patient who has left the X-ray room after a photo has been made.

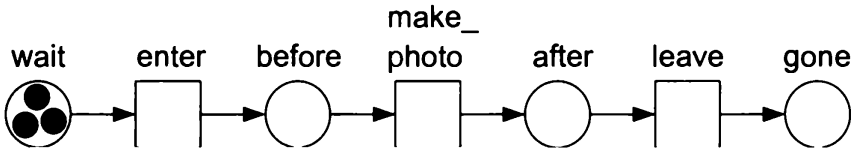


Figure 3.3
A first attempt at modeling the X-ray machine.

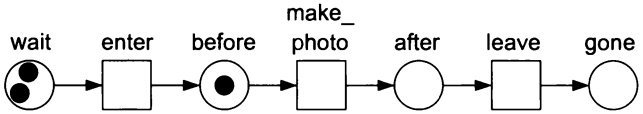


Figure 3.4
A Petri net model of a business process of an X-ray machine: The marking after transition enter has fired.

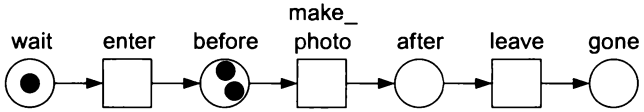


Figure 3.5
A Petri net model of a business process of an X-ray machine: The marking after transition enter has fired again.

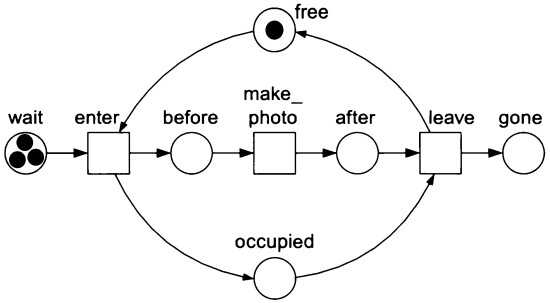


Figure 3.6
An improved Petri net for the business process of an X-ray machine.

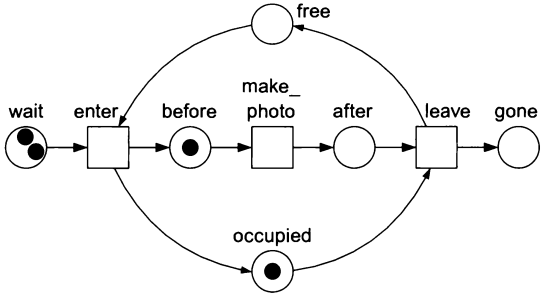


Figure 3.7
The marking of the improved Petri net for the business process of an X-ray machine after transition enter has fired.