SWI041:
Analýza

Hledáme odpověď na otázku:

Co se má udělat?

Nejprve trochu kontroly

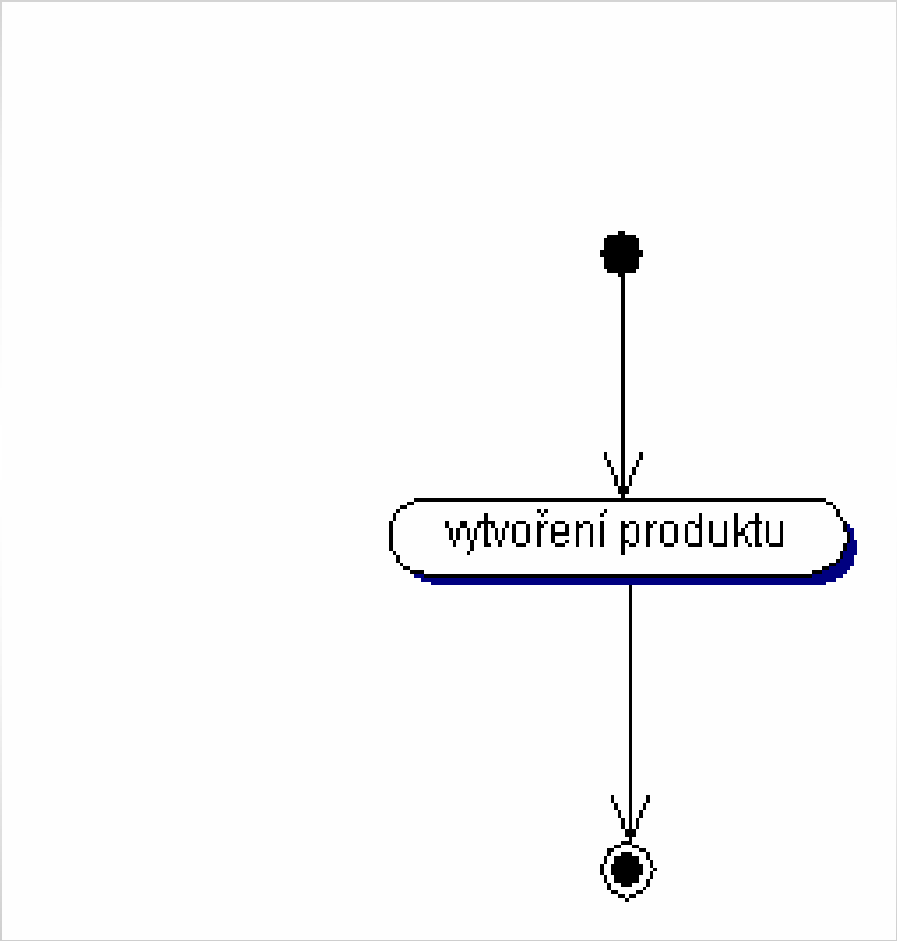
Stav projektů

Proč vytvářet úvodní studii

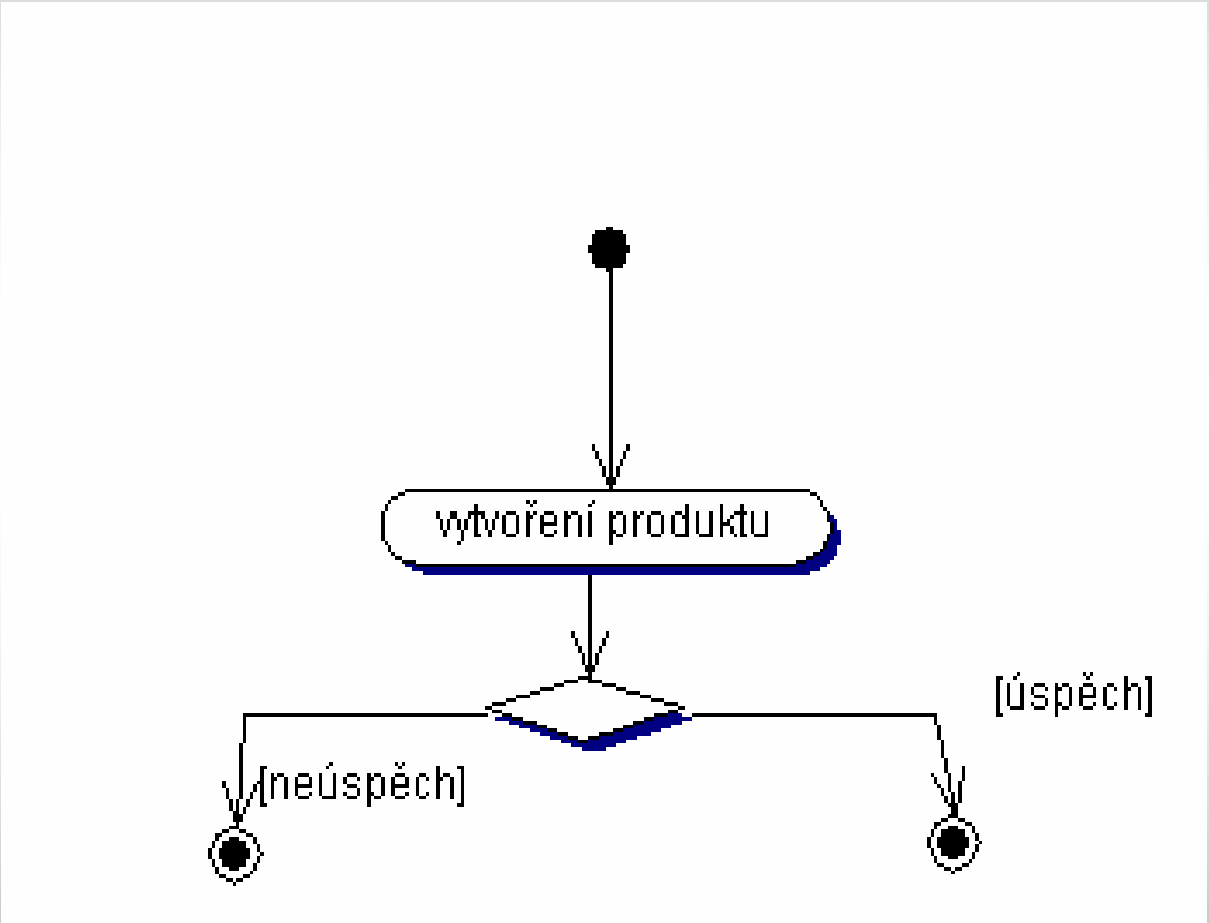
A. Protože to Richta chce.
B. Protože se to v komunitě informatiků sluší.
C. Protože to může ušetřit výdaje.

C je správně !!

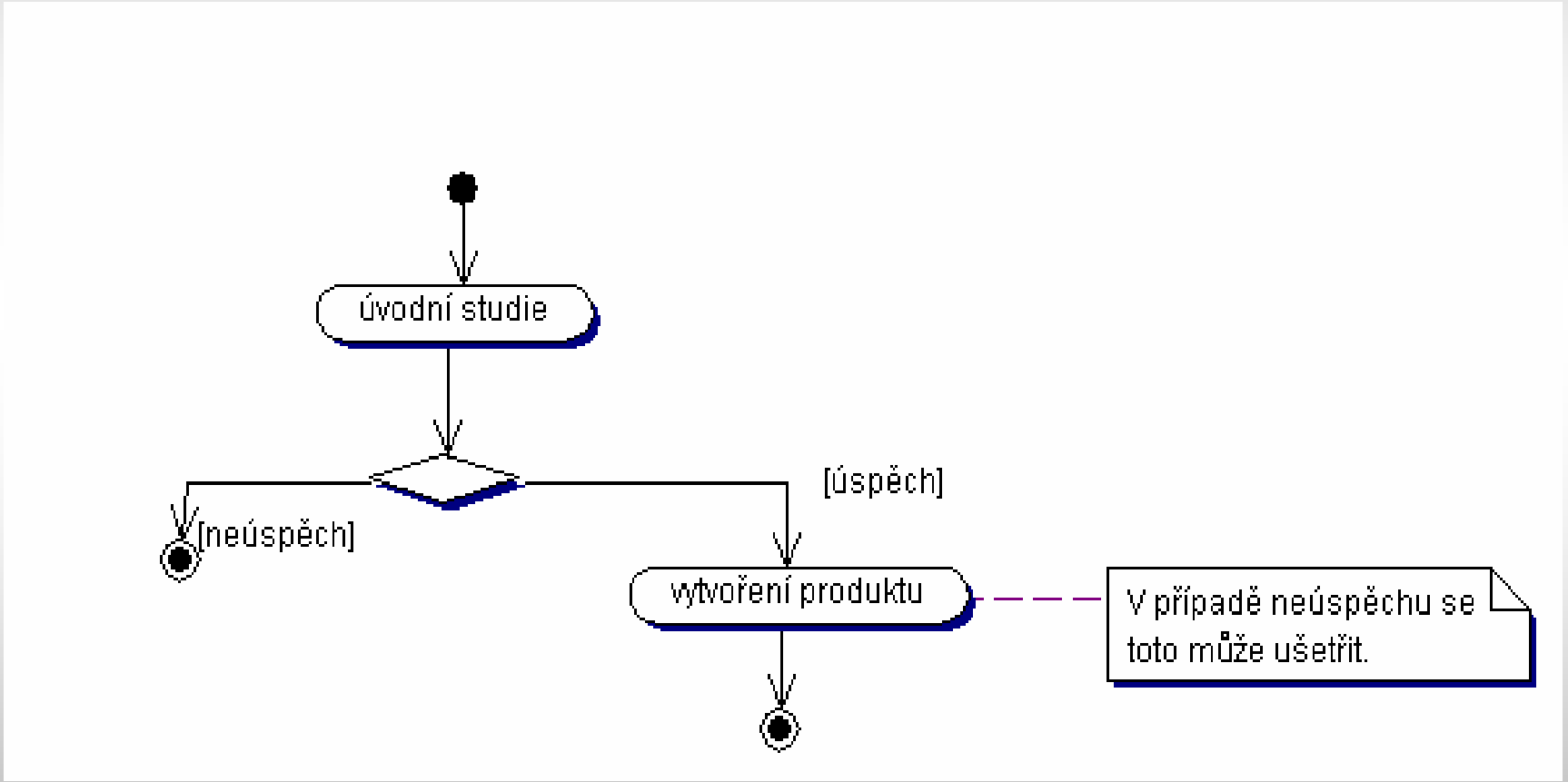
Jak to je ideální



Ale nemusí to vždy dopadnout



Úvodní studie může něco ušetřit



Co obsahuje úvodní studie?

- Stručný náznak o jaký produkt se jedná - definice hranice systému
- Katalog požadavků
- Odhad nákladů a výnosů.
- Pokus o přesvědčení investora, že se vyplatí do projektu vložit peníze.
- Často je vhodné vymezit různé varianty řešení.

Stručná definice hranice systému

- Stručná informace, bez zbytečných podrobností (ty se později doplní v analýze).
  - Deklarace záměru
  - Odborný článek
  - Model jednání (Use Case Model UML, příp. kontext)
  - Slovník použitých termínů (pojmů)
  - Základní datový model

Jazyková agentura - Microsoft Internet Explorer

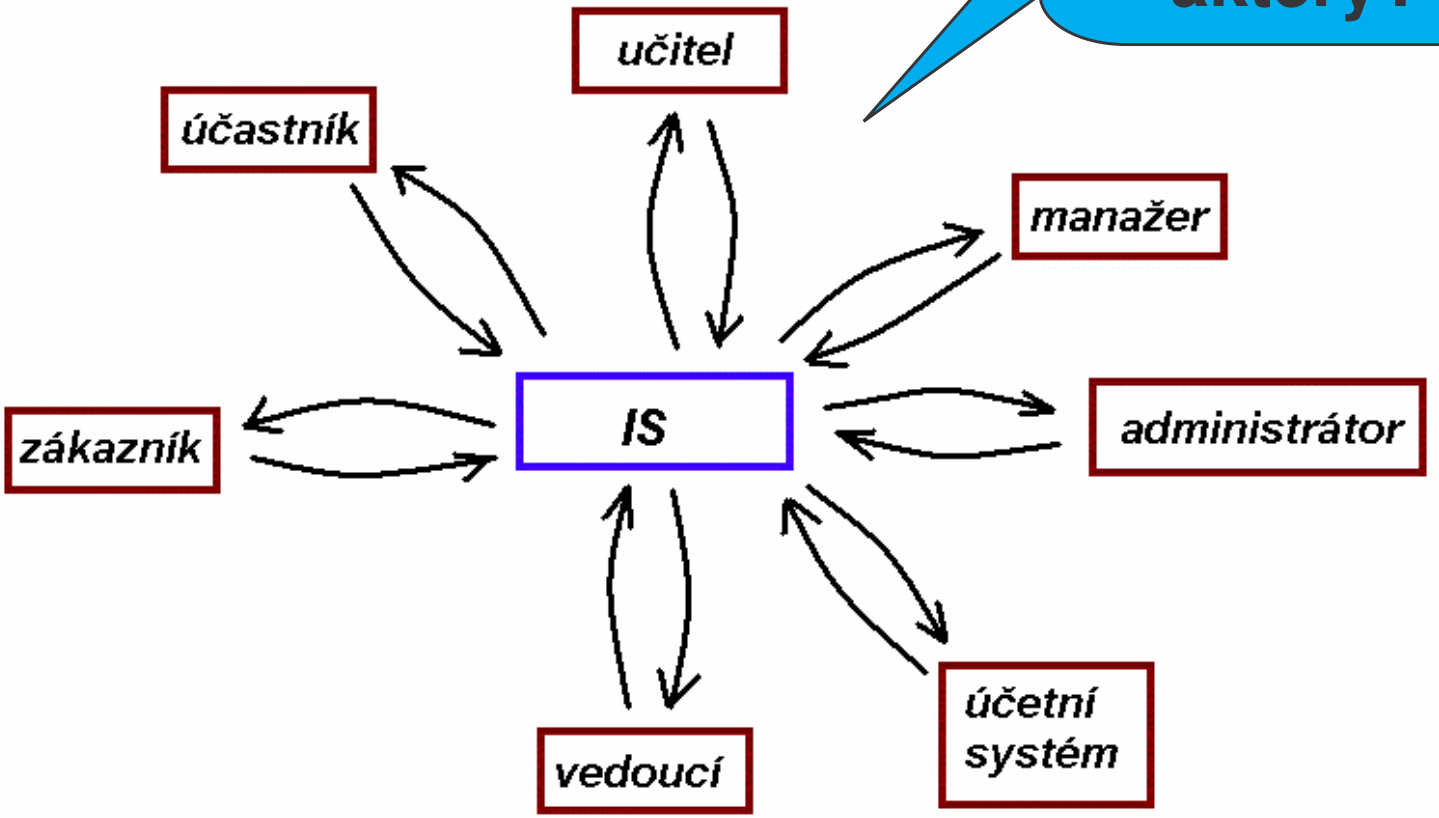
Soubor Úpravy Zobrazit Oblíbené Nástroje Nápověda

Zpět Vpřed Zastavit Aktualizovat Domů Hledat Oblíbené Historie Pošta Tisk Upravit

Adresa https://service.felk.cvut.cz/courses/36SI/prj/36SI3/

Kontextový diagram

Jedná se o různé aktéry?



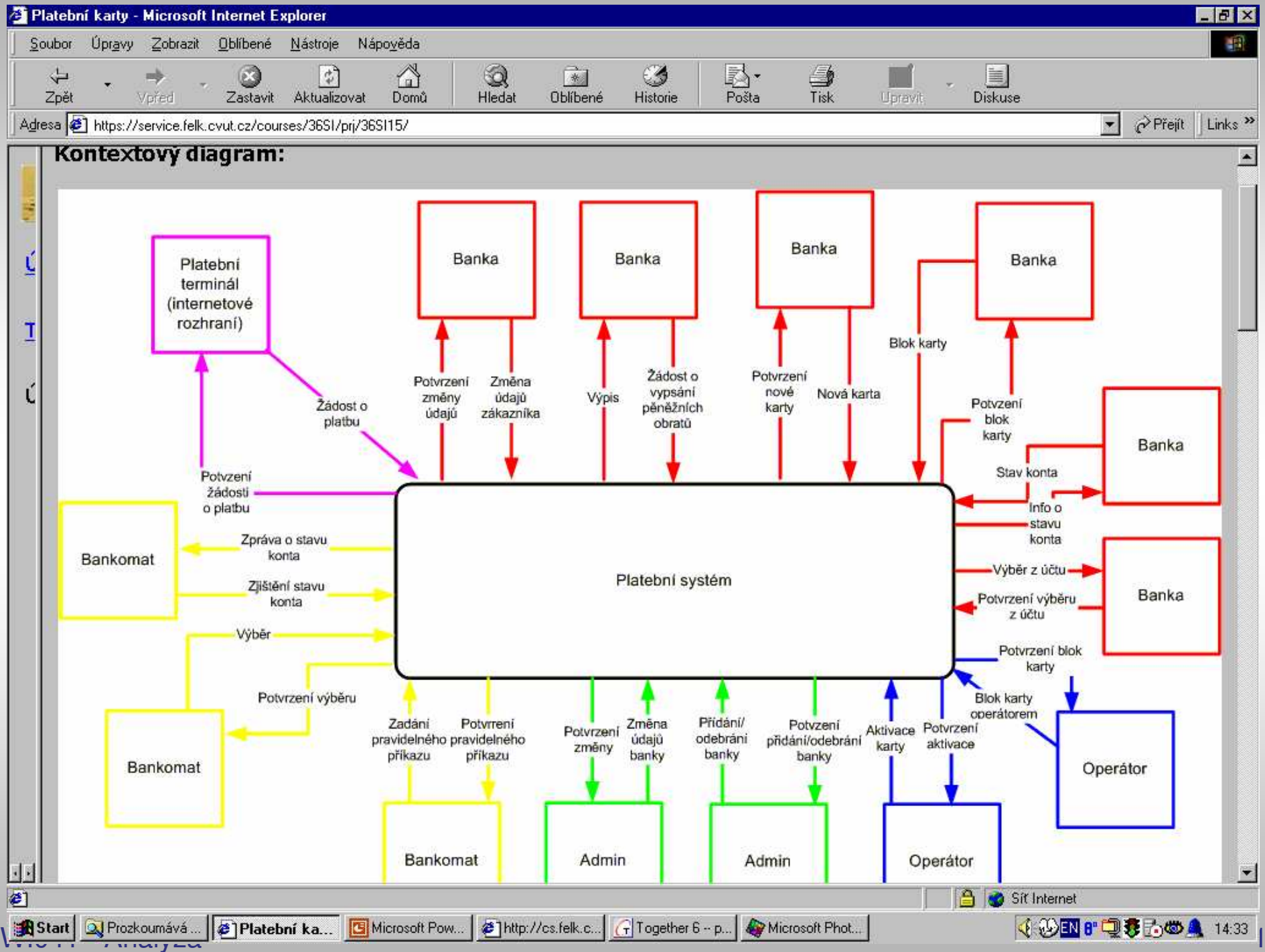
Jednotlivé akce v kontextovém diagramu:

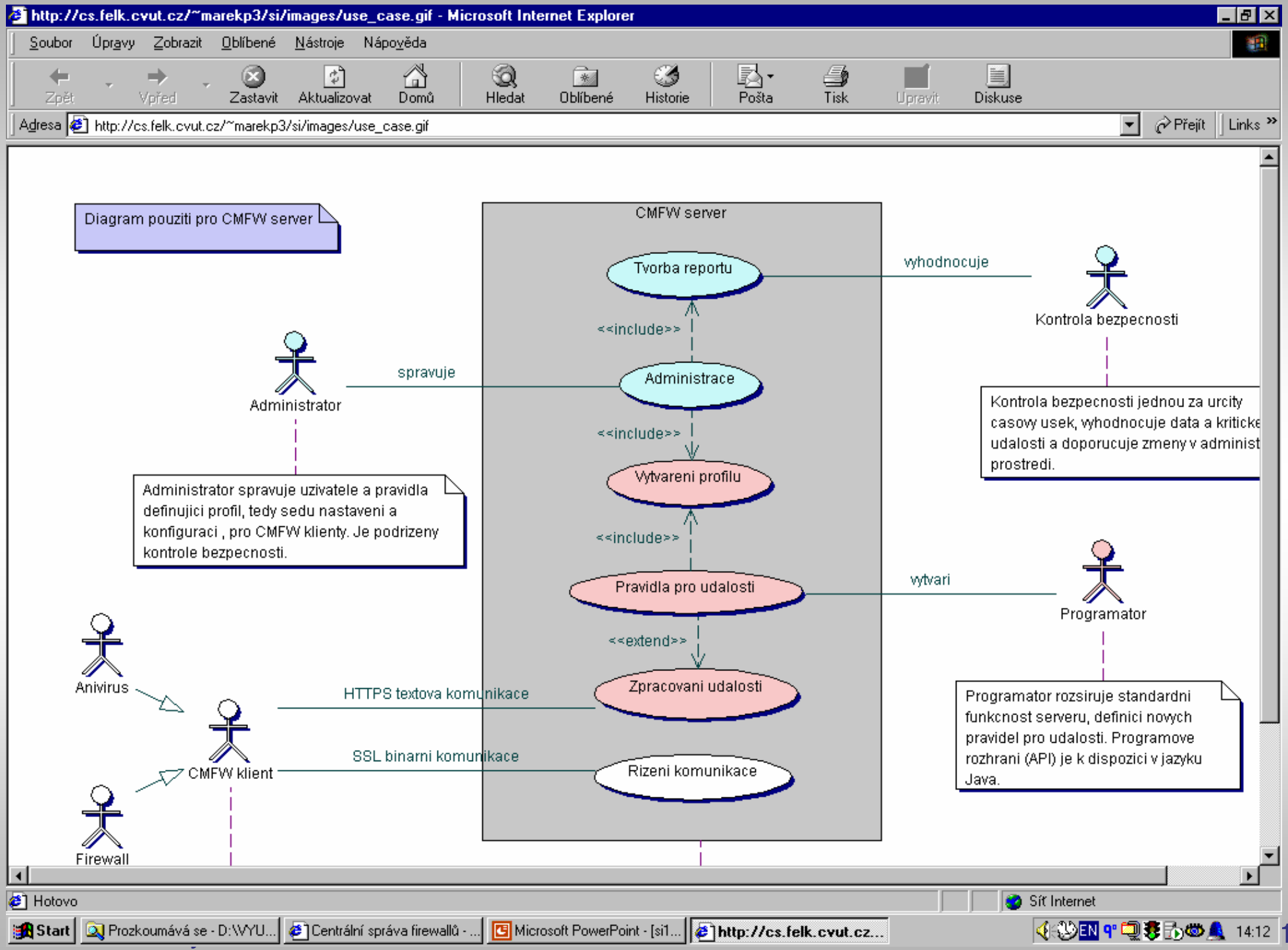
Hotovo

Síť Internet

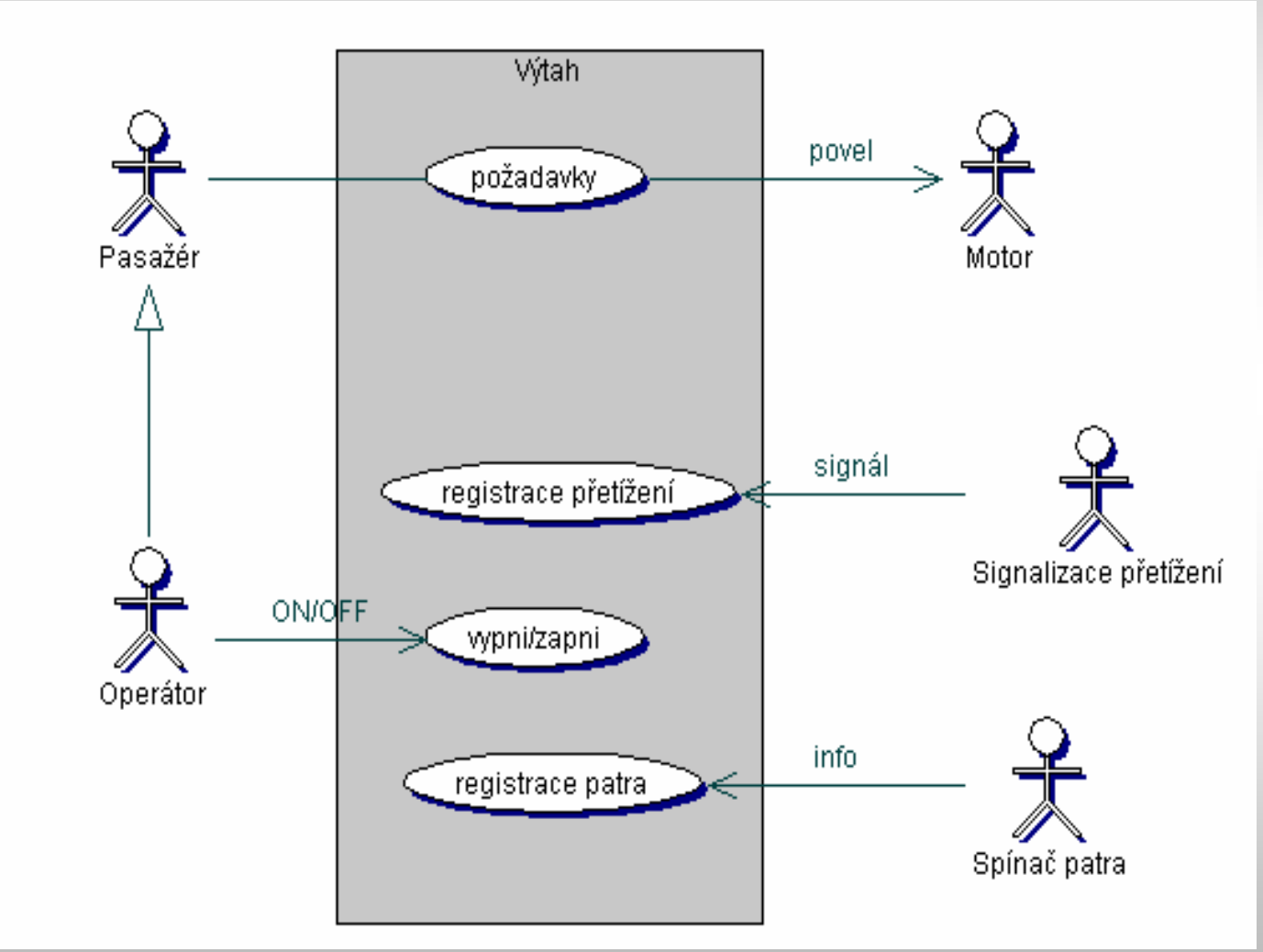
Start | Prozkoumává se - D:\VYU... | Jazyková agentura - ... | Microsoft PowerPoint - [si1...

EN 14:10

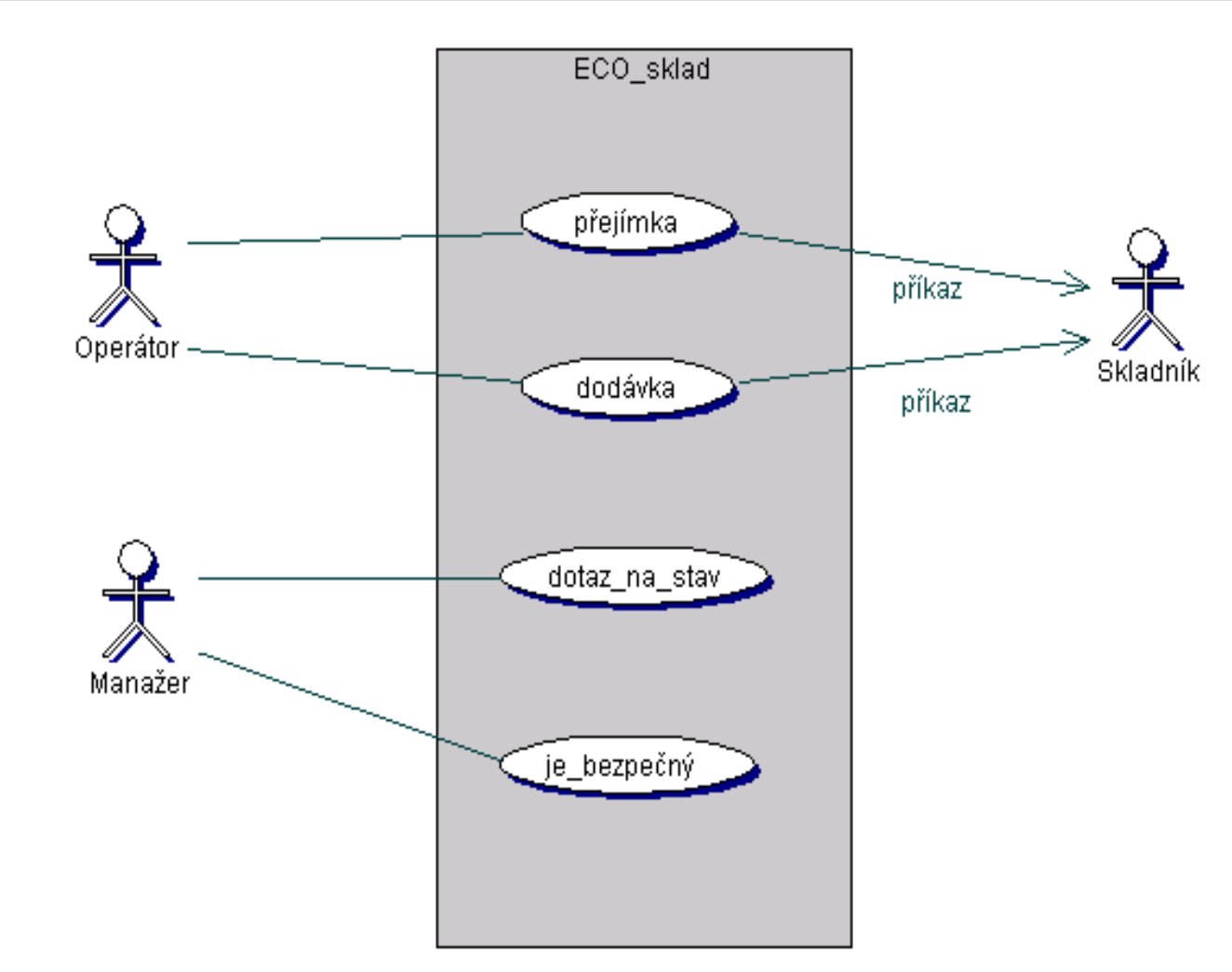

Platební karty - Microsoft Internet Explorer
Soubor Úpravy Zobrazit Oblíbené Nástroje Nápověda
Zpět Vpřed Zastavit Aktualizovat Domů Hledat Oblíbené Historie Pošta Tisk Upravit Diskuse
Adresa https://service.felk.cvut.cz/courses/36SI/prj/36SI15/
Přejít Links
Kontextový diagram:
Platební terminál (internetové rozhraní)
Banka
Banka
Banka
Banka
Banka
Banka
Platební systém
Bankomat
Bankomat
Bankomat
Admin
Admin
Operátor
Operátor
Žádost o platbu
Potvrzení žádosti o platbu
Potvrzení změny údajů
Změna údajů zákazníka
Výpis
Žádost o vypsání pěněžních obratů
Potvrzení nové karty
Nová karta
Blok karty
Potvrzení blok karty
Stav konta
Info o stavu konta
Výběr z účtu
Potvrzení výběru z účtu
Potvrzení blok karty
Blok karty operátorem
Zpráva o stavu konta
Zjištění stavu konta
Výběr
Potvrzení výběru
Zadání pravidelného příkazu
Potvrrení pravidelného příkazu
Potvrzení změny
Změna údajů banky
Přidání/ odebrání banky
Potvzení přidání/odebrání banky
Aktivace karty
Potvrzení aktivace
Síť Internet
Start
Prozkoumává ...
Platební ka...
Microsoft Pow...
http://cs.felk.c...
Together 6 -- p...
Microsoft Phot...
14:33

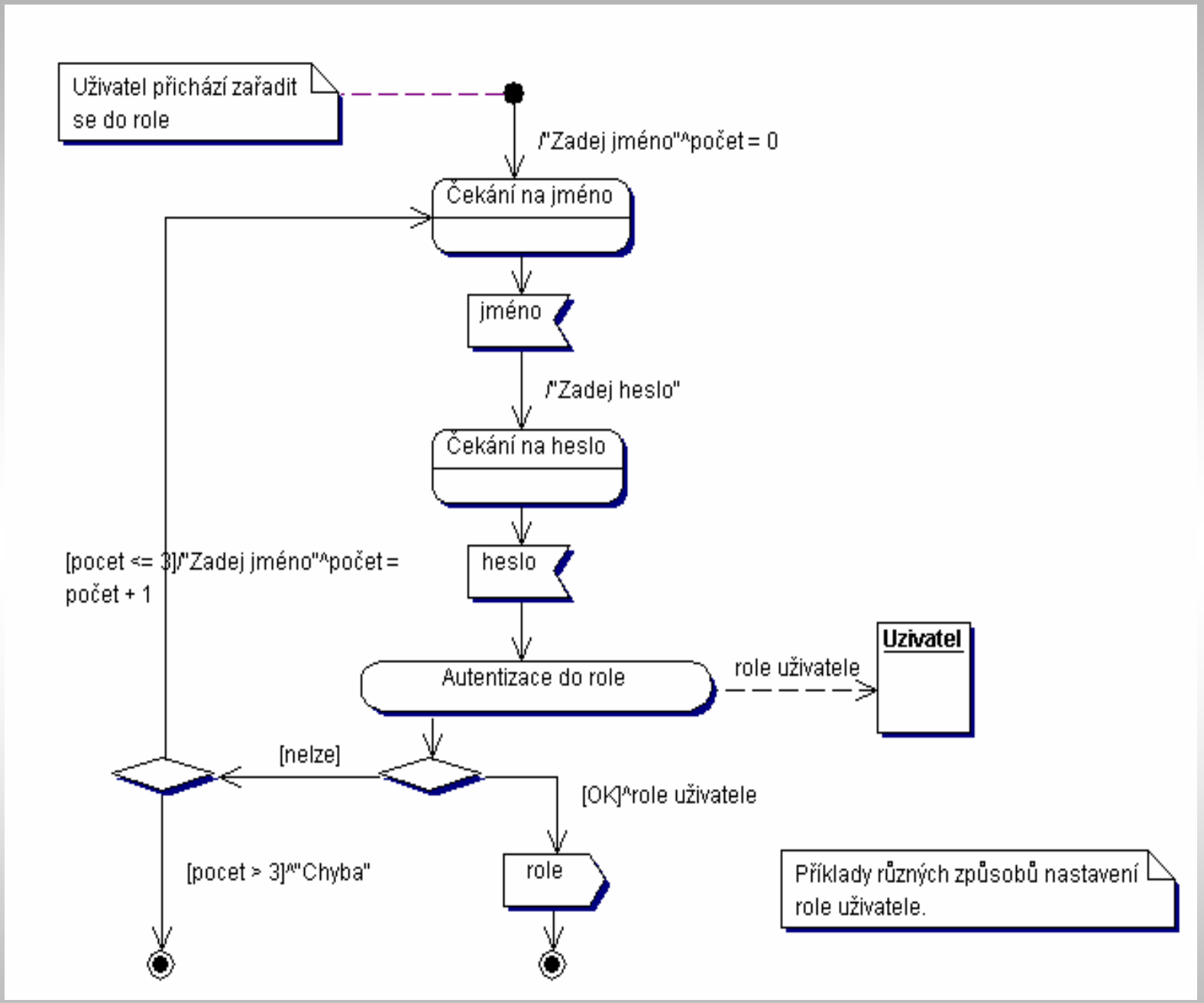

http://cs.felk.cvut.cz/~marekp3/si/images/use_case.gif - Microsoft Internet Explorer
Soubor Úpravy Zobrazit Oblíbené Nástroje Nápověda
Zpět Vpřed Zastavit Aktualizovat Domů Hledat Oblíbené Historie Pošta Tisk Upravit Diskuse
Adresa http://cs.felk.cvut.cz/~marekp3/si/images/use_case.gif
Přejít Links
Diagram pouziti pro CMFW server
CMFW server
Tvorba reportu
vyhodnocuje
Kontrola bezpecnosti
<<include>>
Administrator
spravuje
Administrace
<<include>>
Kontrola bezpecnosti jednou za urcity casovy usek, vyhodnocuje data a kritické udalosti a doporucuje zmeny v administ prostredi.
Vytvareni profilu
<<include>>
Administrator spravuje uzivatele a pravidla definujici profil, tedy sedu nastaveni a konfiguraci , pro CMFW klienty. Je podrizeny kontrole bezpecnosti.
Pravidla pro udalosti
vytvari
Programator
<<extend>>
Anivirus
HTTPS textova komunikace
Zpracovani udalosti
Programator rozsiruje standardni funkcnost serveru, definici novych pravidel pro udalosti. Programove rozhrani (API) je k dispozici v jazyku Java.
CMFW klient
SSL binarni komunikace
Rizeni komunikace
Firewall
Hotovo
Síť Internet
Start
Prozkoumává se - D:\VYU...
Centrální správa firewallů - ...
Microsoft PowerPoint - [si1...
http://cs.felk.cvut.cz...
EN
14:12

Model jednání pro Výtah



Model jednání ECO-skladu




Uživatel přichází zařadit se do role
/"Zadej jméno"^počet = 0
Čekání na jméno
jméno
/"Zadej heslo"
Čekání na heslo
heslo
[pocet <= 3]/"Zadej jméno"^počet = počet + 1
Autentizace do role
role uživatele
Uživatel
[nelze]
[OK]^role uživatele
[pocet > 3]^"Chyba"
role
Příklady různých způsobů nastavení role uživatele.

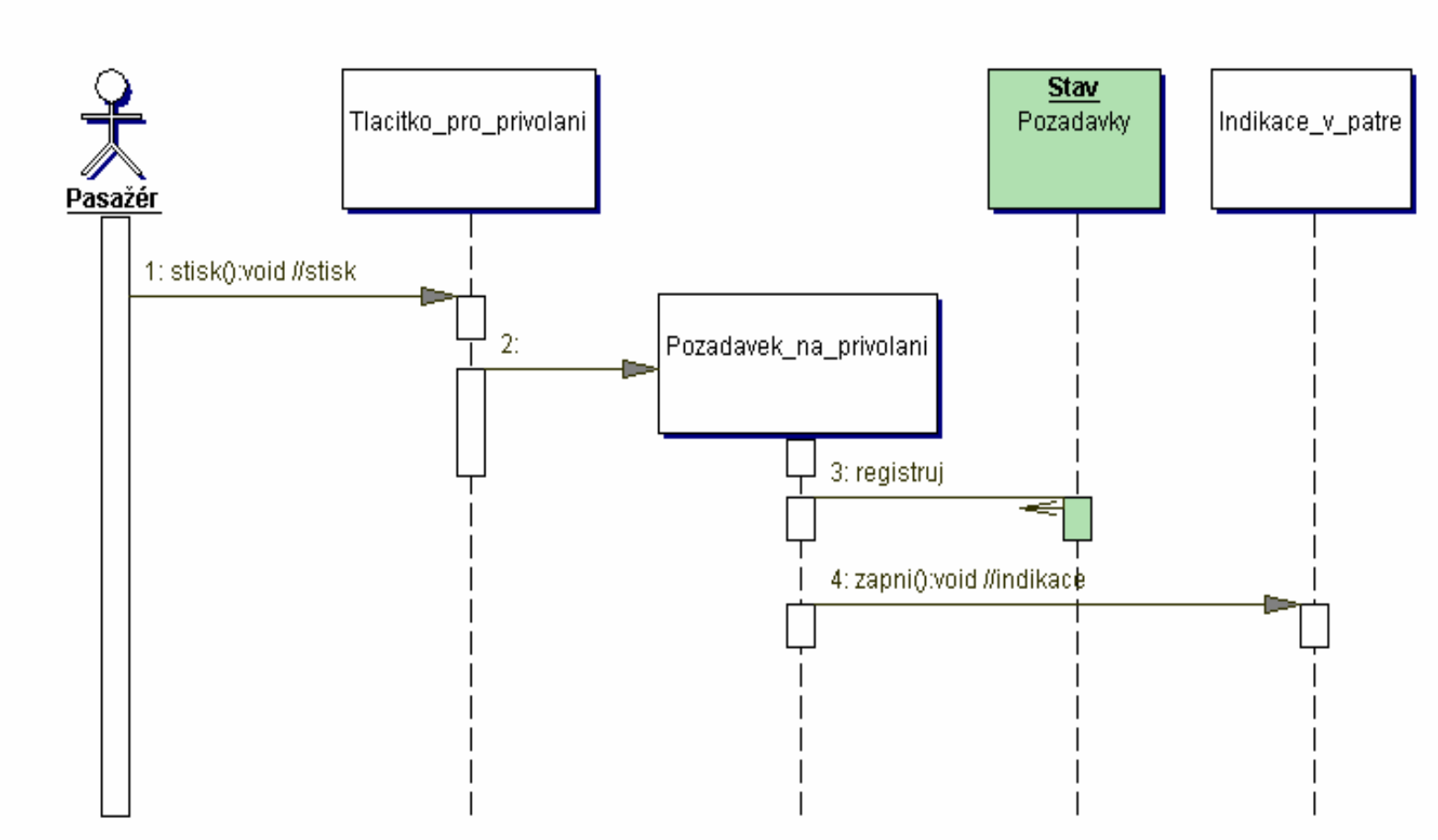
Scénáře událostí (Sequence diagrams)

(zachycení sledu událostí)

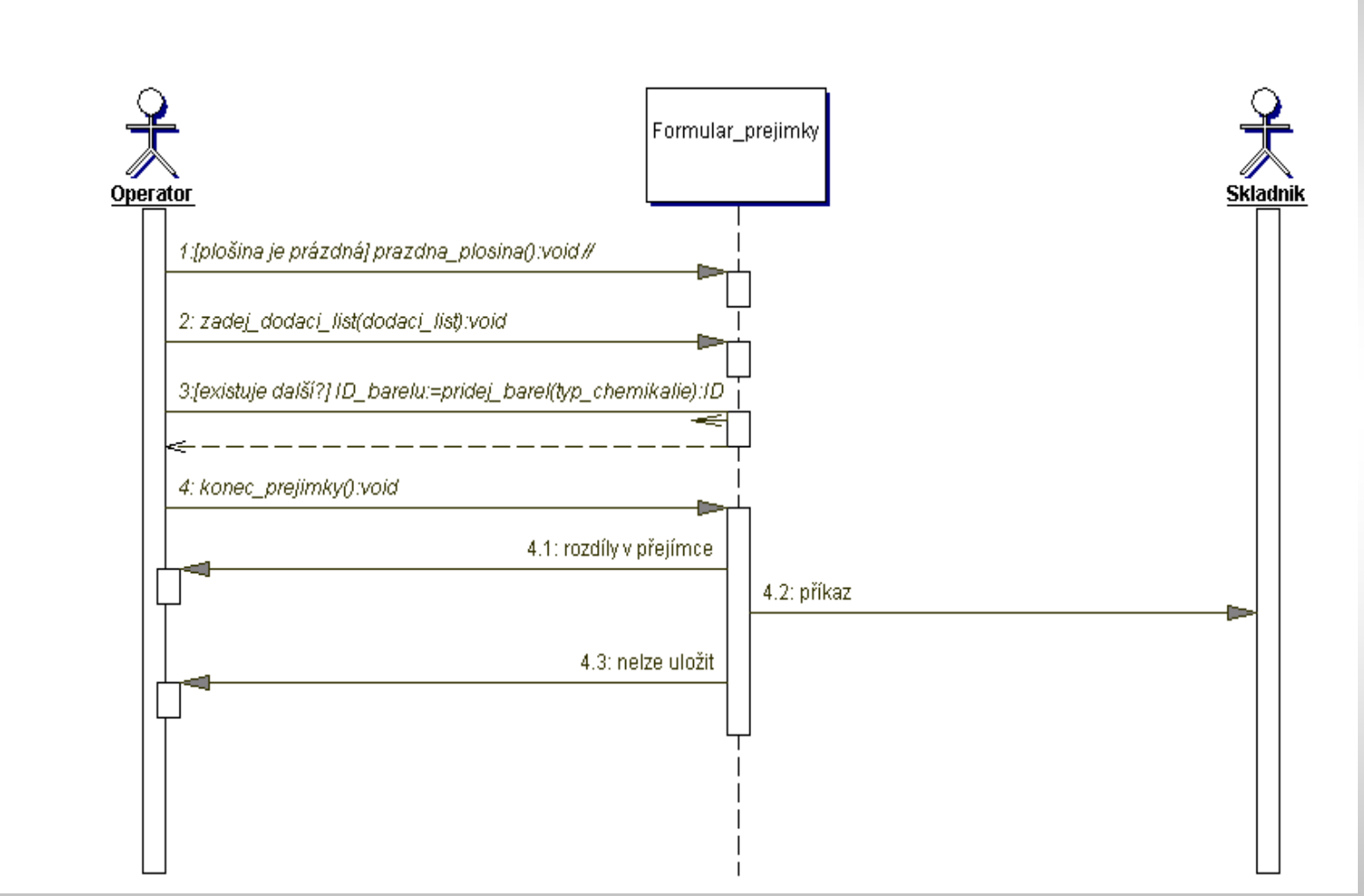
Prvky:

- **objekty** - znázorněné obvykle jako sloupce
- **interakce mezi objekty** (stimuly) - orientované šipky mezi objekty
- **události** - události, které vyvolaly interakci
- **reakce** - odezvy na události (výstupy)
- **časová osa** - pro vyznačení sledu událostí

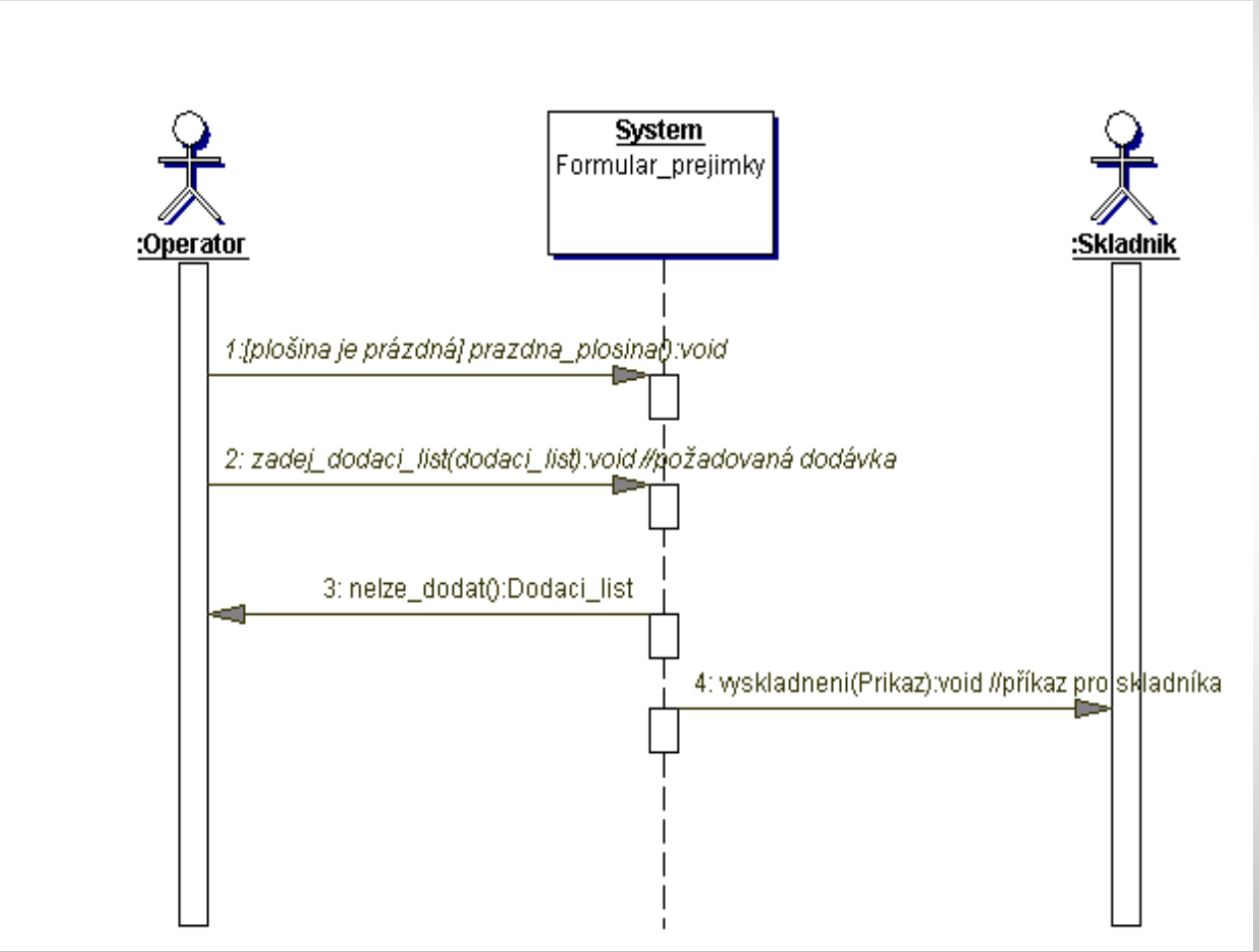
Scénář pro přivolání



Scénář pro “přejímku”



Scénář pro “dodávku”

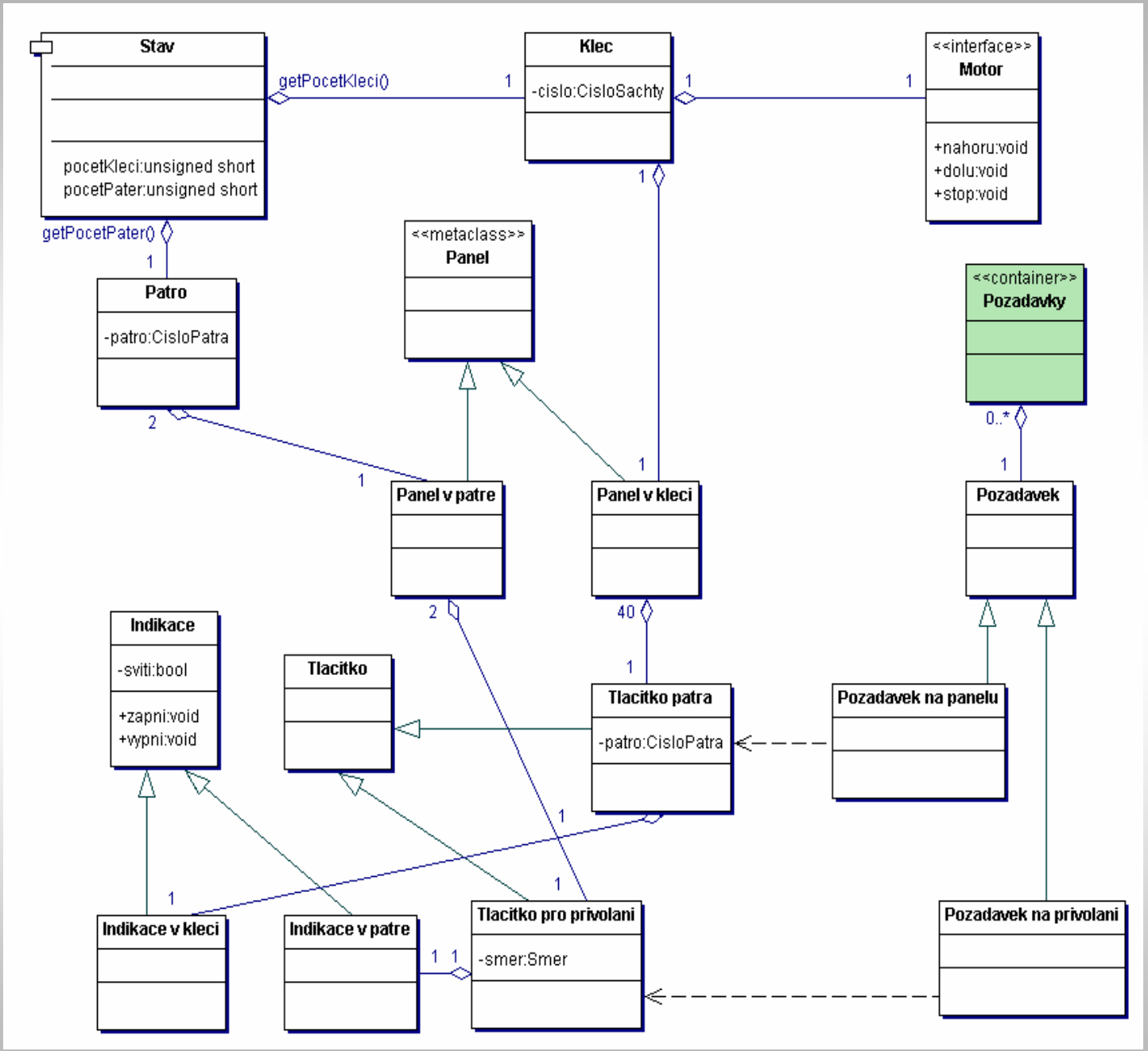


Datový model (konceptuální)

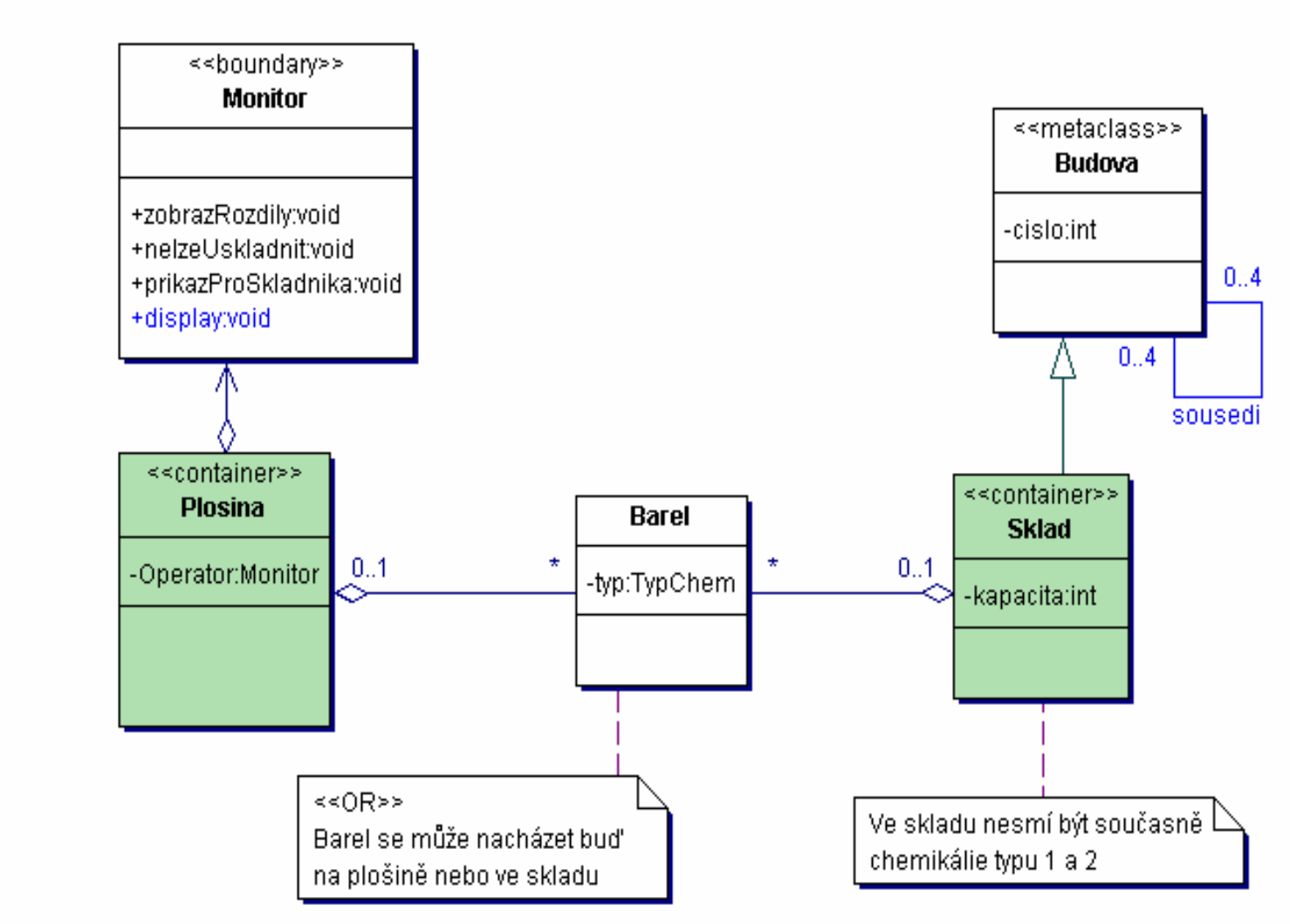
(zachycení analýzy dat)

Komponenty:

- **typy objektů** (entity) - entita = rozlišitelný identifikovatelný objekt
- **vztahy** (relationships) - množiny instancí reprezentujících vztahy mezi (2 a více) objekty
- indikace **přidružených** objektů - pro vztahy o nichž si potřebujeme něco pamatovat
- indikace vztahů **nadtyp-podtyp**, **celek-část** (gen-spec, whole-part) - vyjádření vztahu společný - speciální (dědičnost)

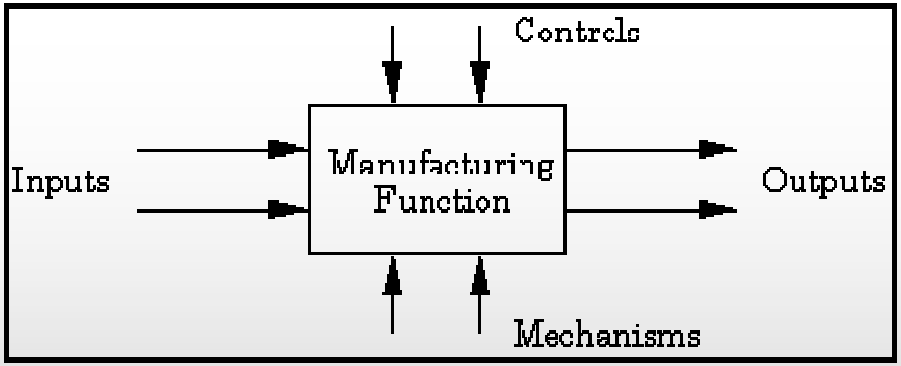

Stav
pocetKleci:unsigned short
pocetPater:unsigned short
getPocetKleci()
getPocetPater()
Klec
-cislo:CisloSachty
<<interface>>
Motor
+nahoru:void
+dolu:void
+stop:void
Patro
-patro:CisloPatra
<<metaclass>>
Panel
<<container>>
Pozadavky
Panel v patre
Panel v kleci
Pozadavek
Indikace
-sviti:bool
+zapni:void
+vypni:void
Tlacitko
Tlacitko patra
-patro:CisloPatra
Pozadavek na panelu
Indikace v kleci
Indikace v patre
Tlacitko pro privolani
-smer:Smer
Pozadavek na privolani

Datový model ECO (1.verze)



Alternativní notace

- Integrated Definition – IDEF (U.S. Air Force - http://www.idef.com)



Alternativní notace

- Architecture of Integrated Information Systems – ARIS
  (prof. Scheer, SAP)
  - http://www.ids-scheer.com

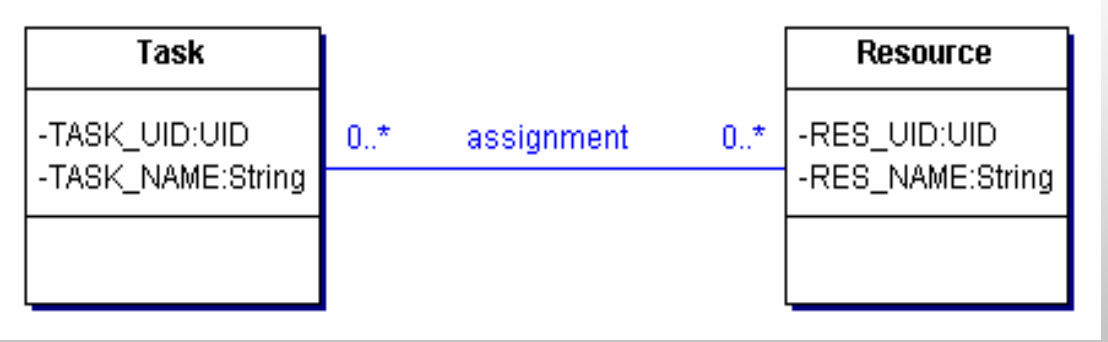
Datové modelování

Fáze datového modelování

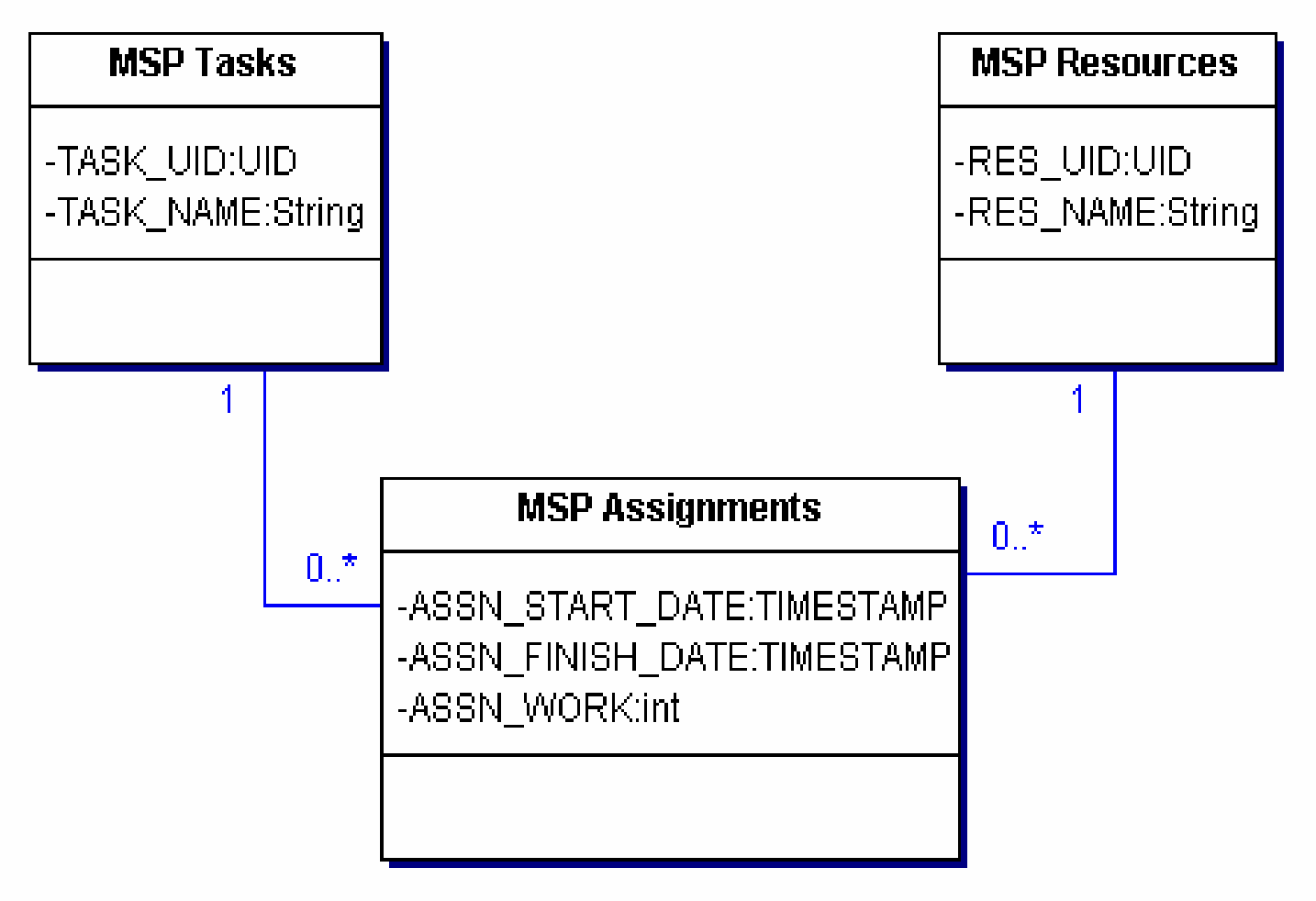
- Sběr požadavků
- Analýza dat a vytvoření **konceptuálního datového modelu** (ER-model, model tříd, PIM)
- Návrh reprezentace dat – **logický datový model** (např. relační model, objektový model, PSM)
- Implementace datového modelu – skutečné vyjádření datových charakteristik pro konkrétní prostředí (**fyzický model**)

Příklad: MS Project

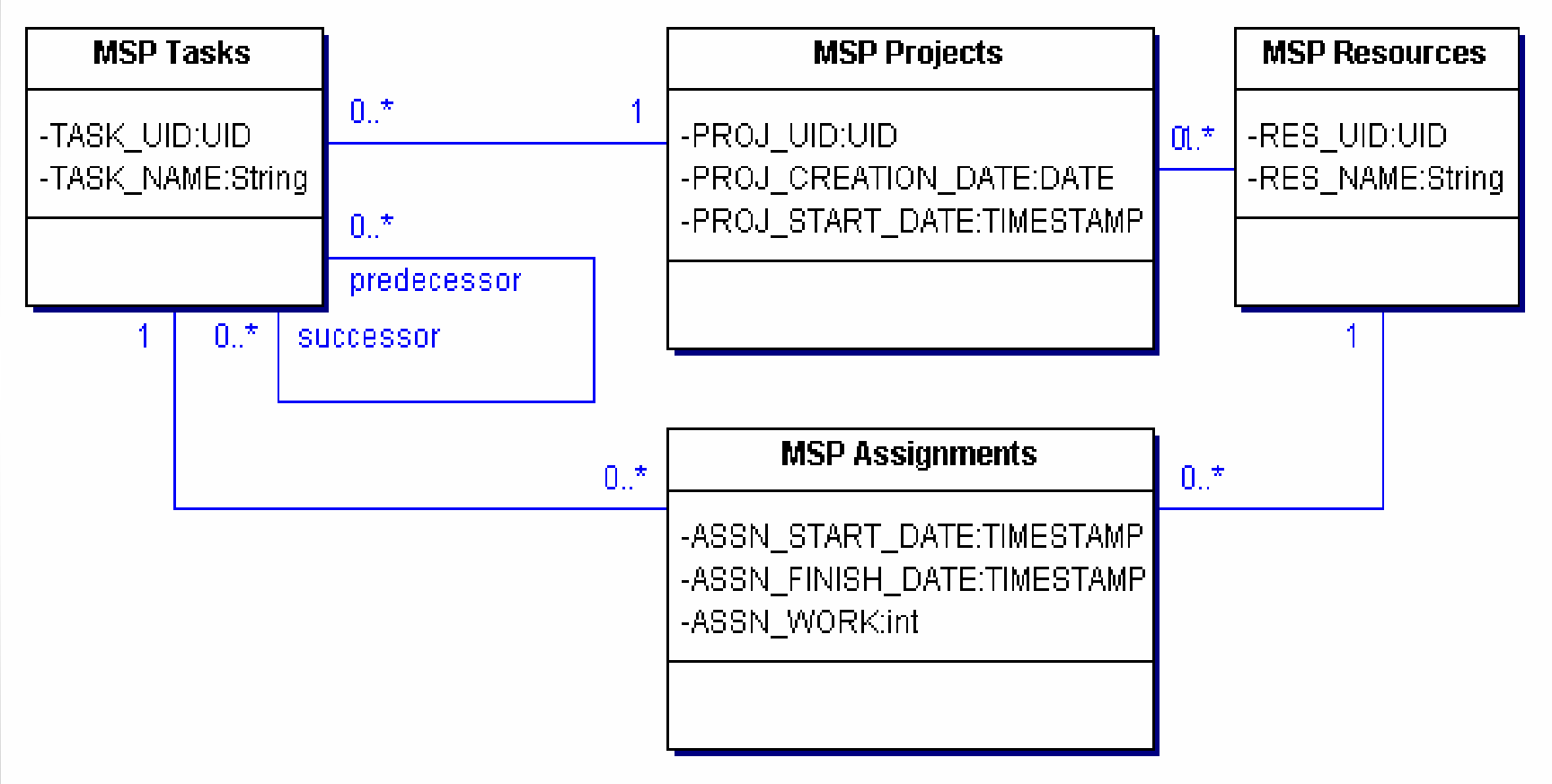
- Požadavky: aplikace bude pracovat s úlohami, zdroji a vztahy.
- Odtud kandidáti na entity (typy objektů, třídy):
  - Úloha
  - Zdroj
  - Přiřazení
- První model:



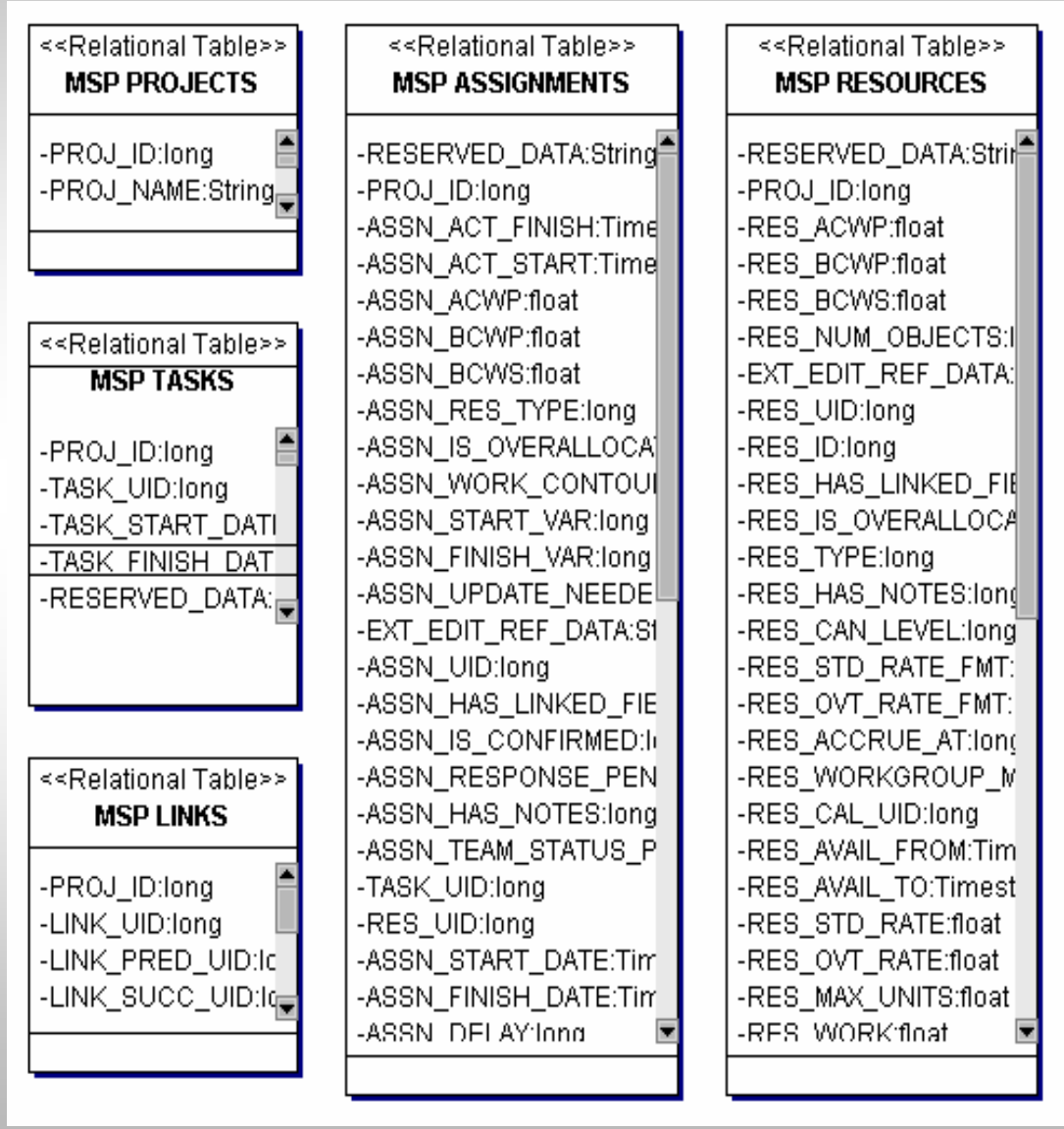
Podrobnější model



Ještě podrobnější model



Použijeme-li relační databázi (část)



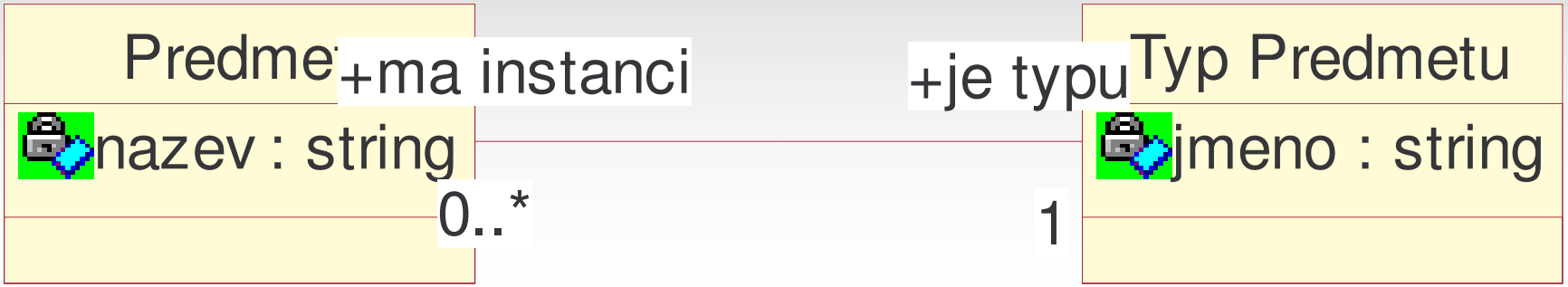
Skutečná implementace

```
CREATE TABLE  MSP_TASKS (
PROJ_ID NUMBER(18,0),
TASK_UID NUMBER(18,0), … , PRIMARY KEY (PROJ_ID,TASK_UID)
);

CREATE TABLE MSP_RESOURCES (
PROJ_ID NUMBER(18,0),
RES_UID NUMBER(18,0),
RES_NAME VARCHAR2(255), … , PRIMARY KEY (PROJ_ID,RES_UID)
);

CREATE TABLE MSP_LINKS (
PROJ_ID NUMBER(18,0),
LINK_UID NUMBER(18,0),
LINK_PRED_UID NUMBER(18,0),
LINK_SUCC_UID NUMBER(18,0), … ,
FOREIGN KEY (PROJ_ID, LINK_PRED_UID)
REFERENCES MSP_TASKS (PROJ_ID, TASK_UID) …
);
```

Pozor na role ve vztazích

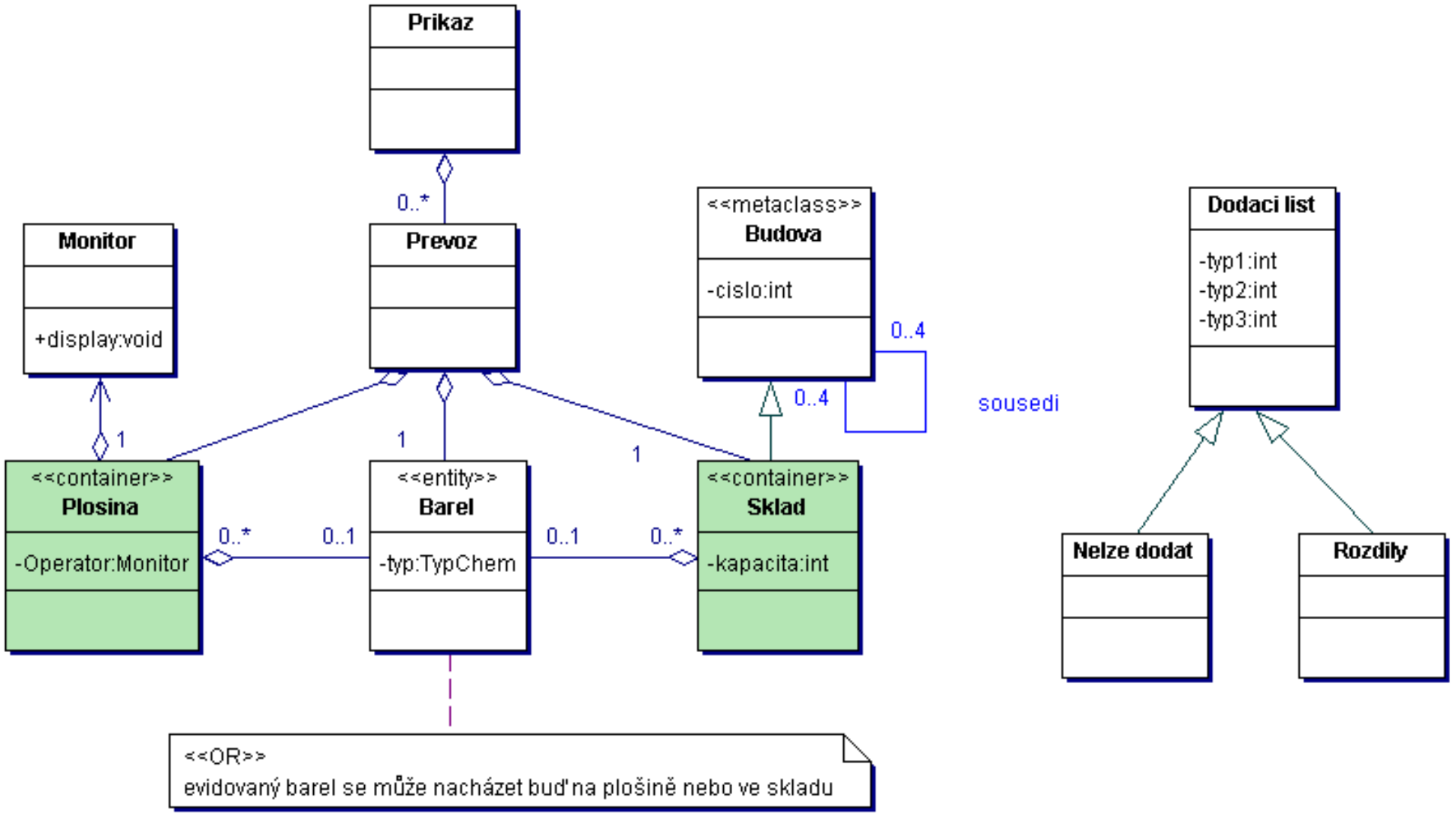


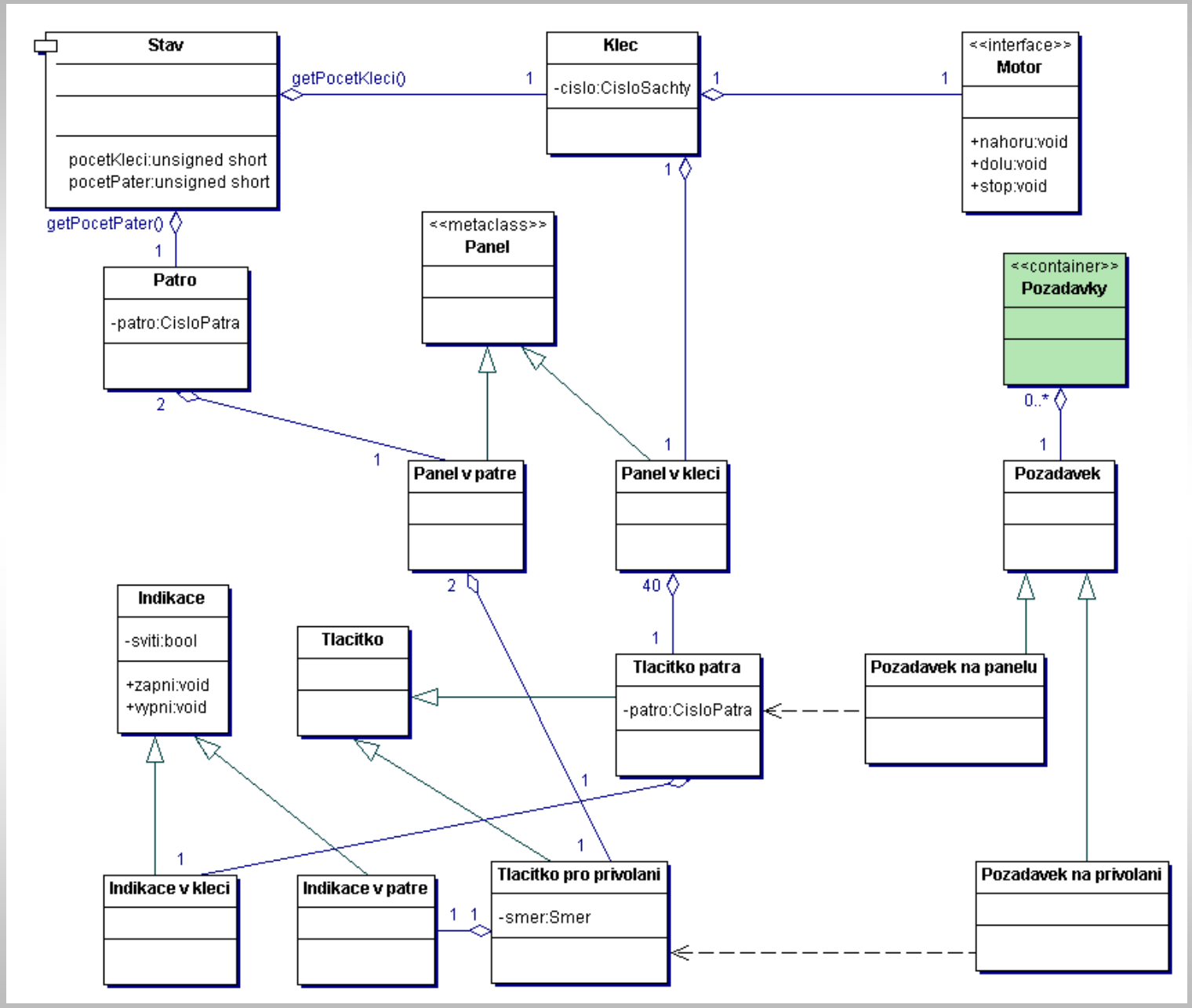
Předmět je právě jednoho typu

Typ předmětu má 0..* instancí

Generovaný kód

```
#ifndef Predmet_h
#define Predmet_h
#include "Typ Predmetu.h"
class Predmet {
public:
        Predmet();
        ~Predmet();
        const Typ_Predmetu * get_je_typu () const;
         void set_je_typu (Typ_Predmetu * value);
private:
        const string get_nazev () const;
         void set_nazev (string value);
        string nazev;
        Typ_Predmetu *je_typu;
};
```

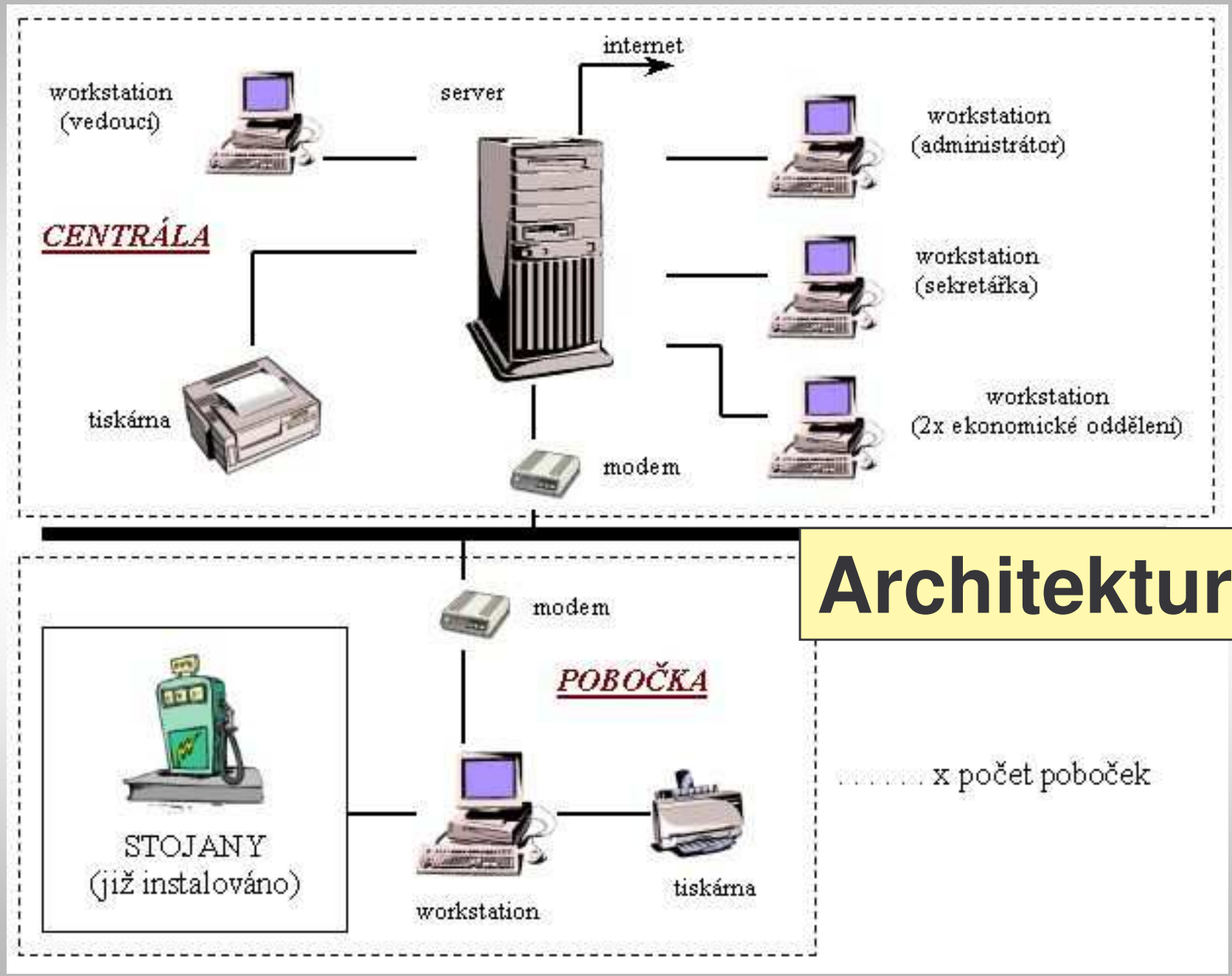

Prikaz
0..*
Prevoz
Monitor
+display:void
1
<<container>>
Plosina
-Operator:Monitor
0..*
0..1
1
<<entity>>
Barel
-typ:TypChem
0..1
0..*
1
<<container>>
Sklad
-kapacita:int
<<metaclass>>
Budova
-cislo:int
0..4
0..4
sousedi
Dodaci list
-typ1:int
-typ2:int
-typ3:int
Nelze dodat
Rozdily
<<OR>>
evidovaný barel se může nacházet buď na plošině nebo ve skladu


Stav
pocetKleci:unsigned short
pocetPater:unsigned short
getPocetKleci()
getPocetPater()
Klec
-cislo:CisloSachty
<<interface>>
Motor
+nahoru:void
+dolu:void
+stop:void
Patro
-patro:CisloPatra
<<metaclass>>
Panel
<<container>>
Pozadavky
Panel v patre
Panel v kleci
Pozadavek
Indikace
-sviti:bool
+zapni:void
+vypni:void
Tlacitko
Tlacitko patra
-patro:CisloPatra
Pozadavek na panelu
Indikace v kleci
Indikace v patre
Tlacitko pro privolani
-smer:Smer
Pozadavek na privolani

CASE nástroje

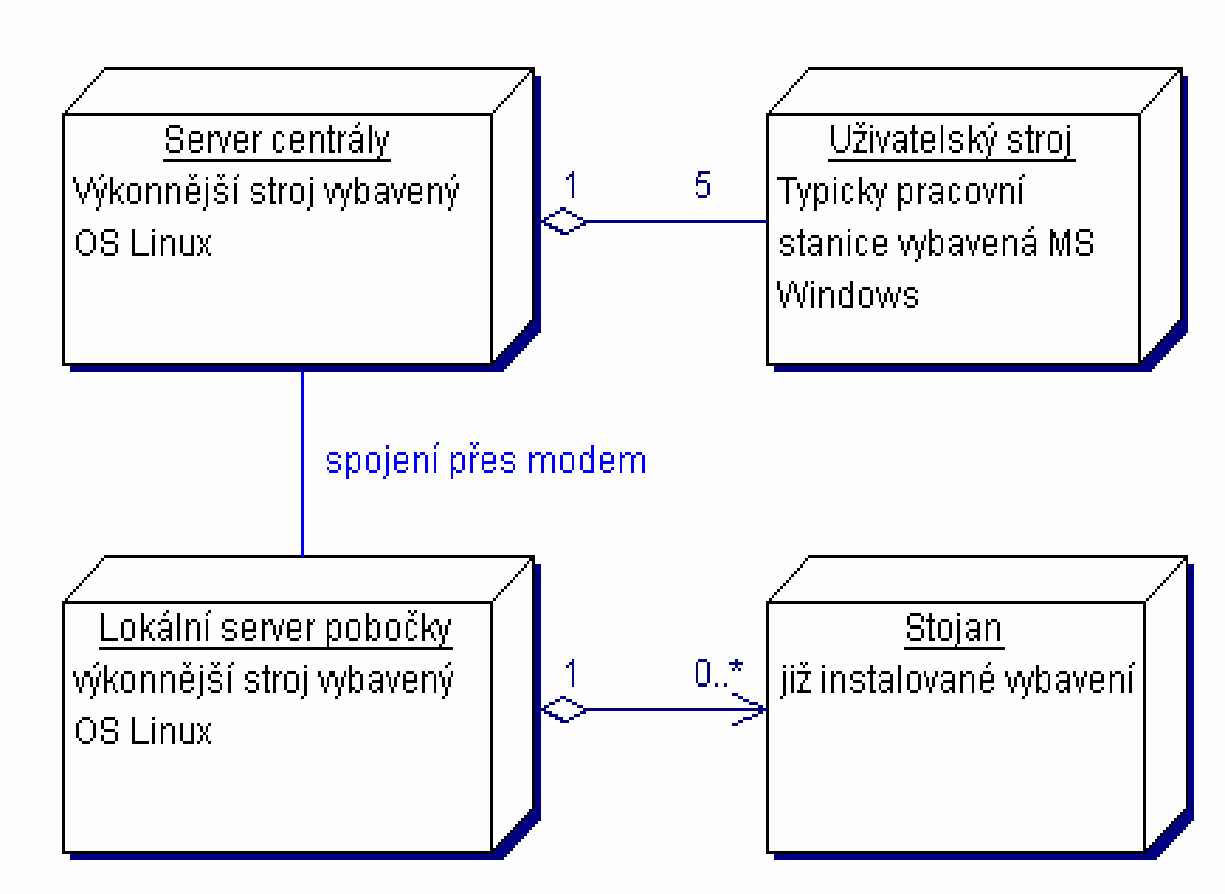
- http://www.uml.org

Definice architektury systému

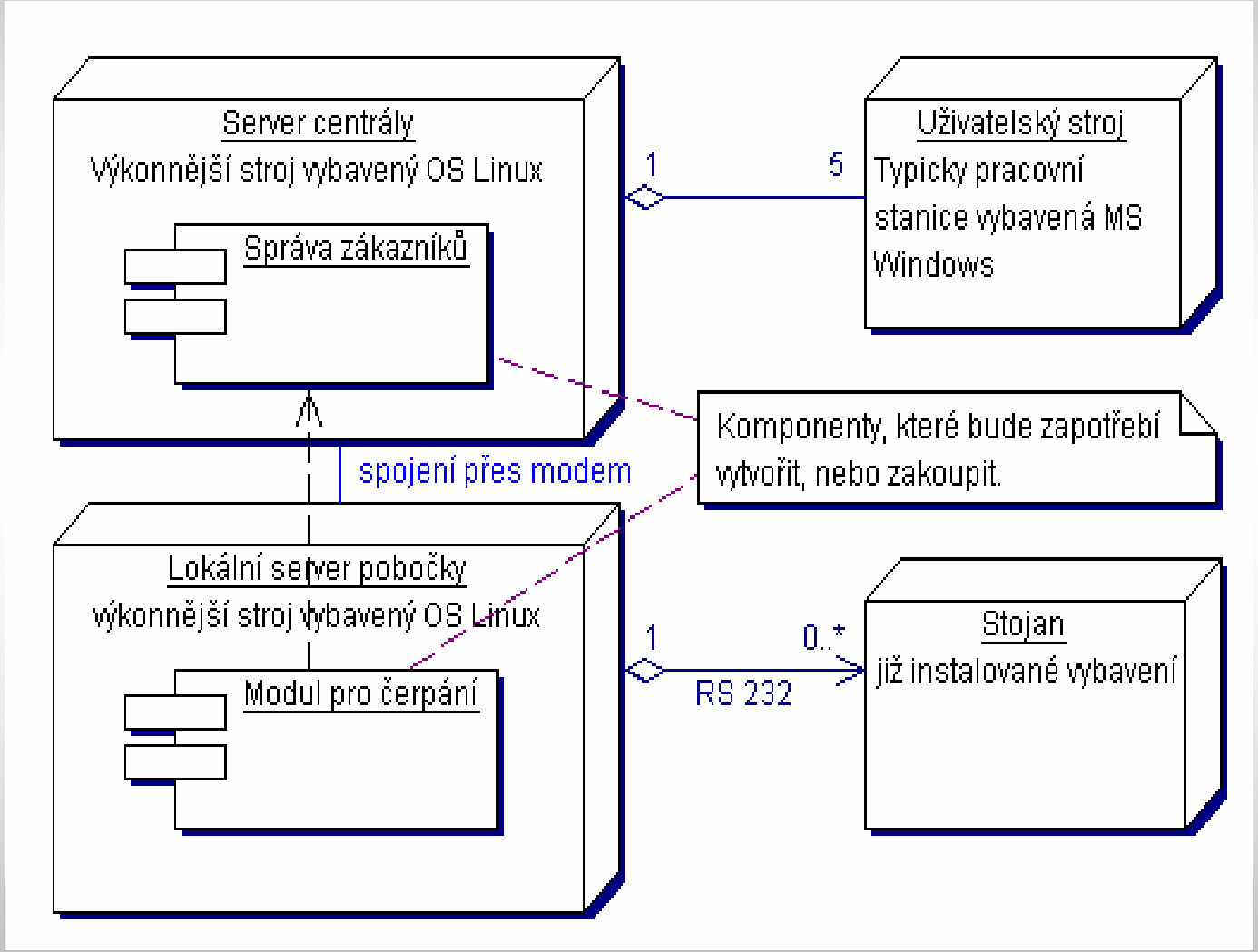


Architektura

První představa o rozmístění v UML

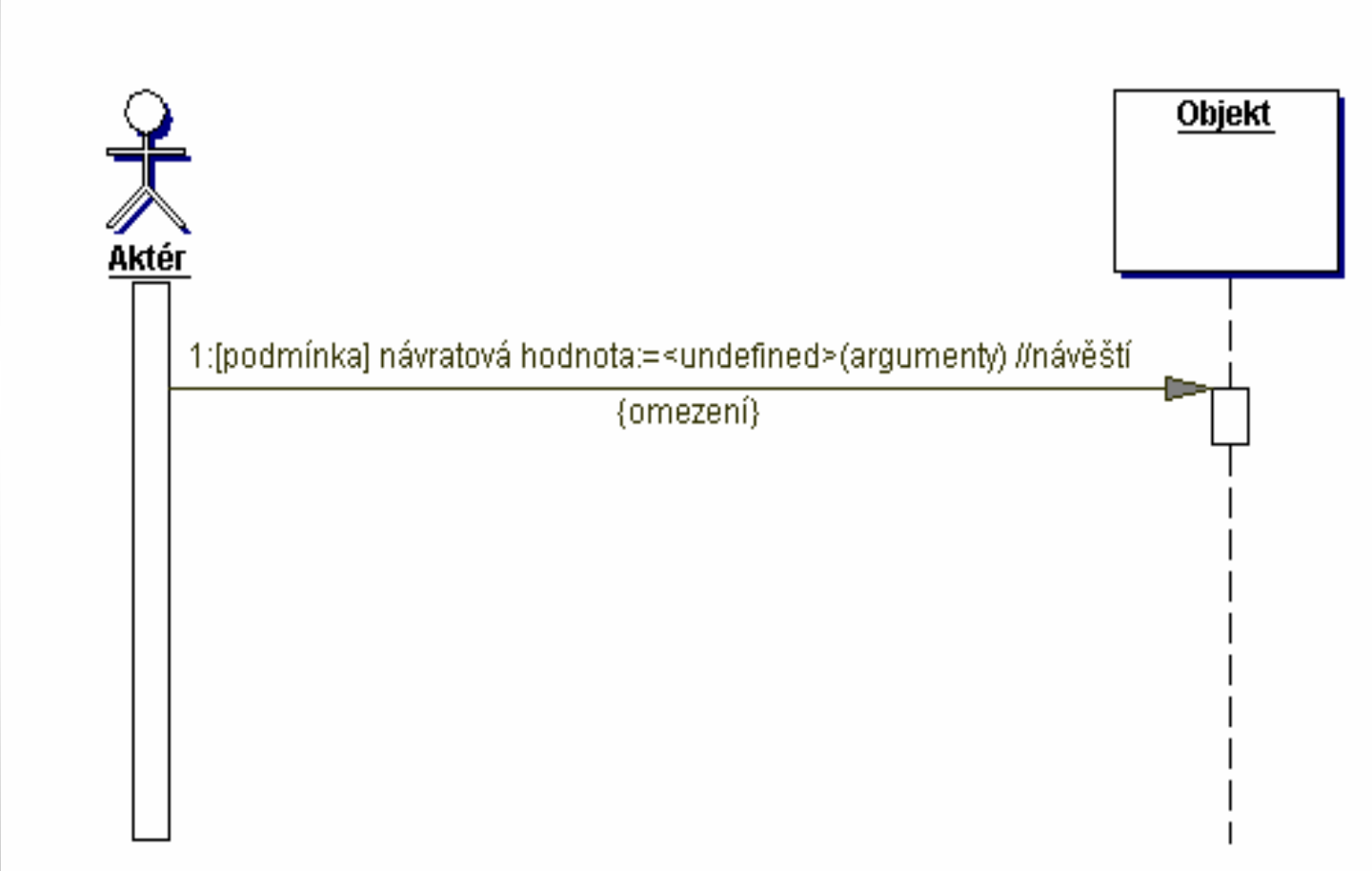


Druhá představa o rozmístění v UML

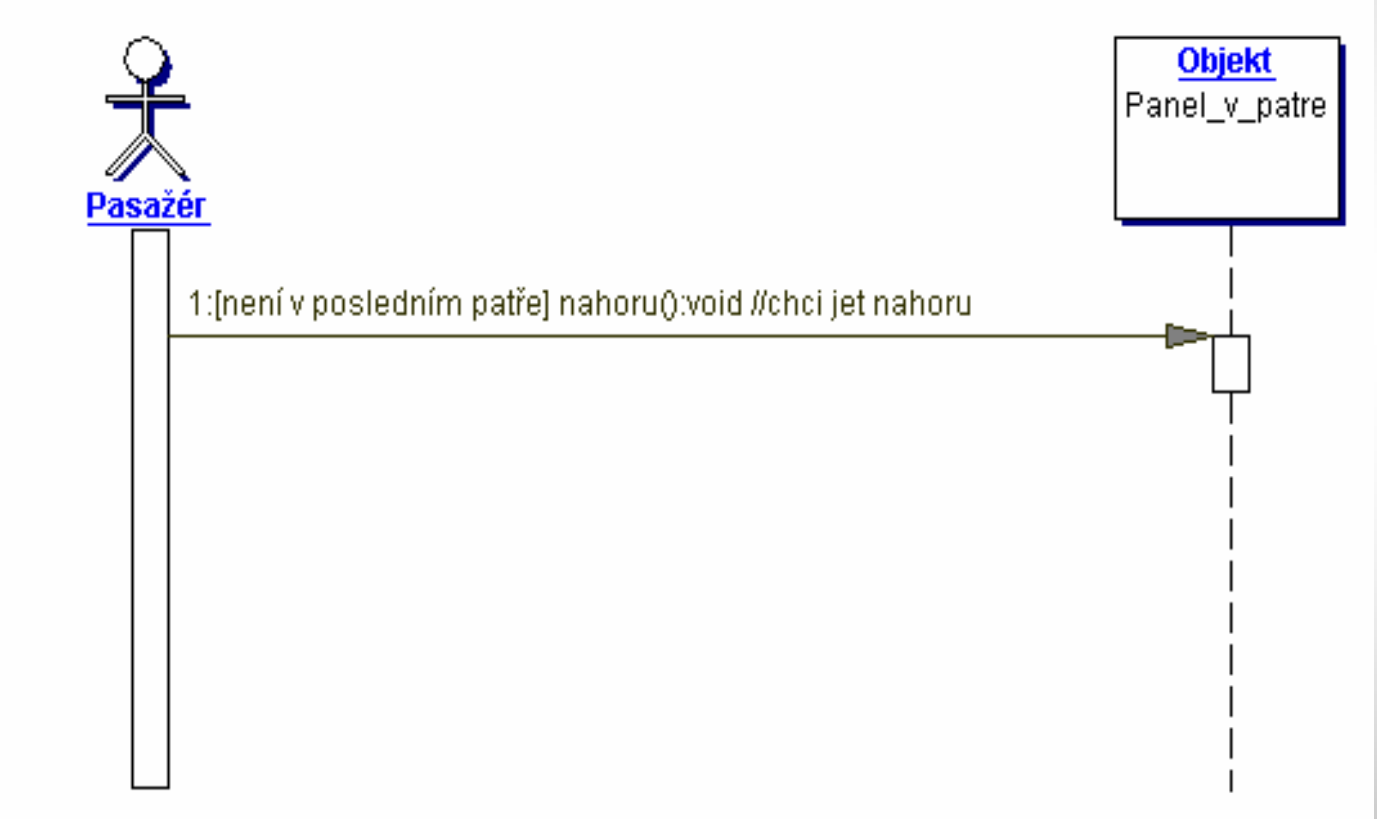


Funkční model

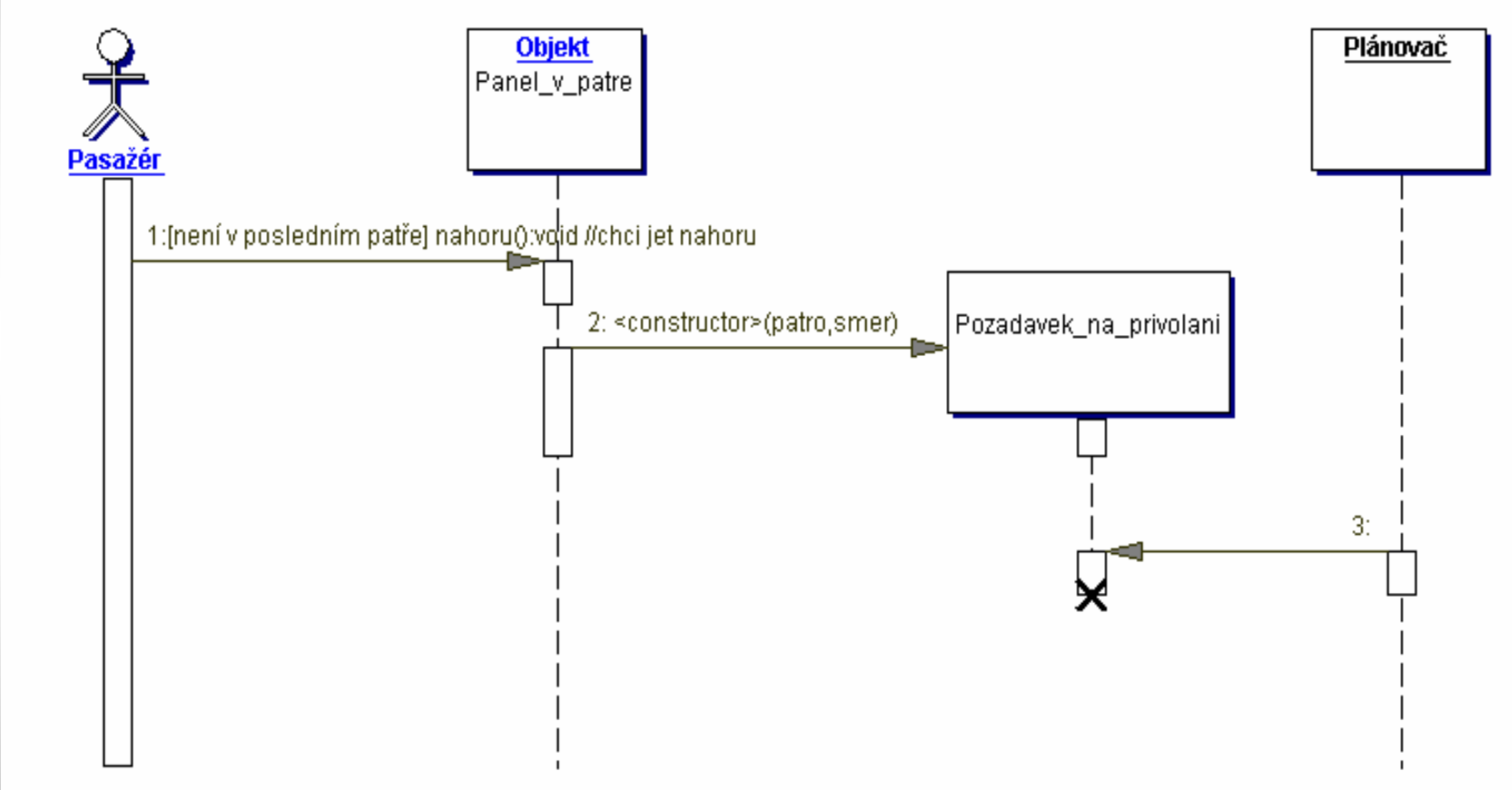
Základní princip scénáře



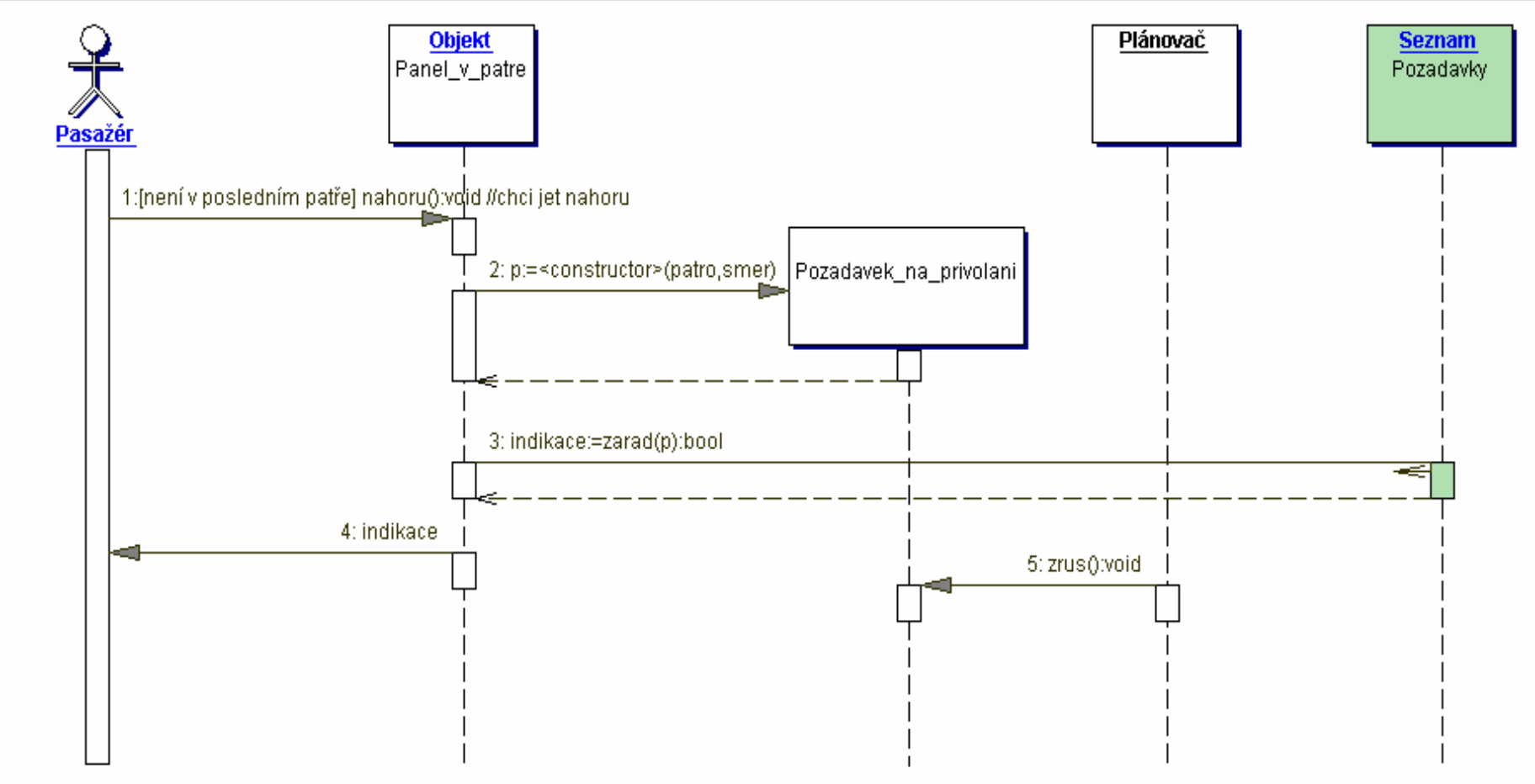
Zvolíme-li konkrétní metodu



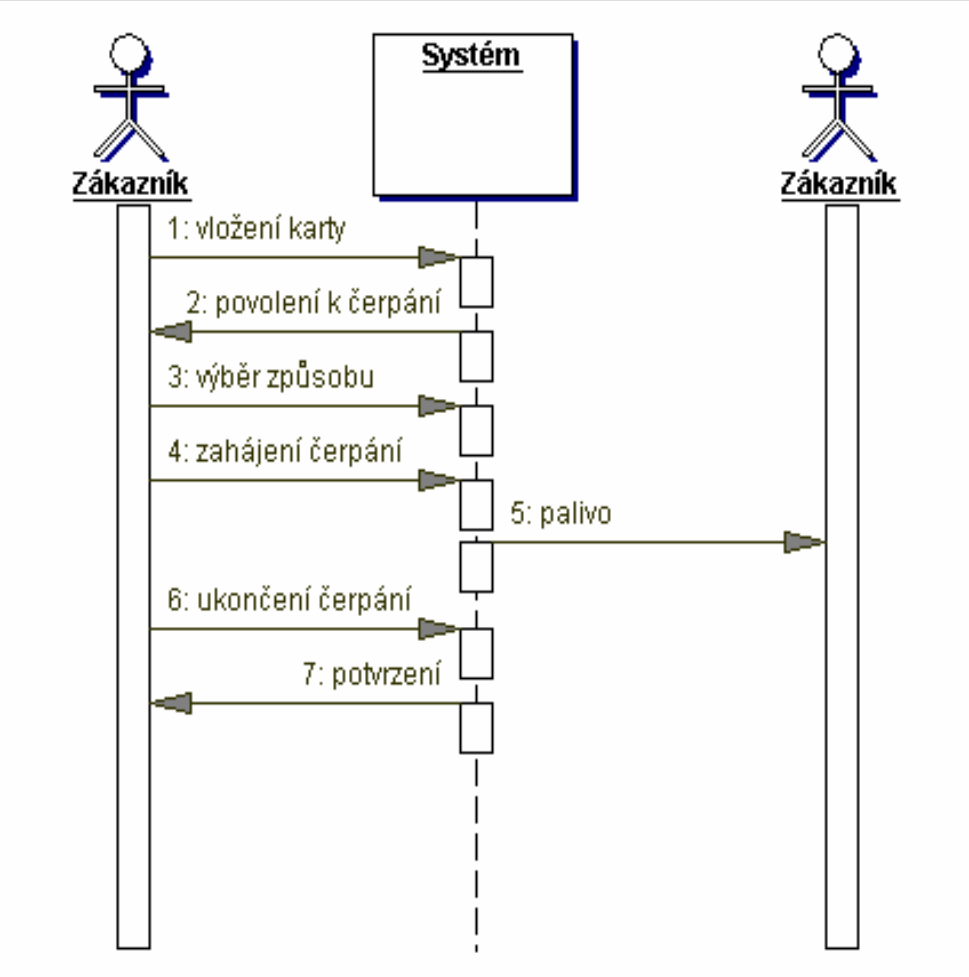
Konstrukce a destrukce



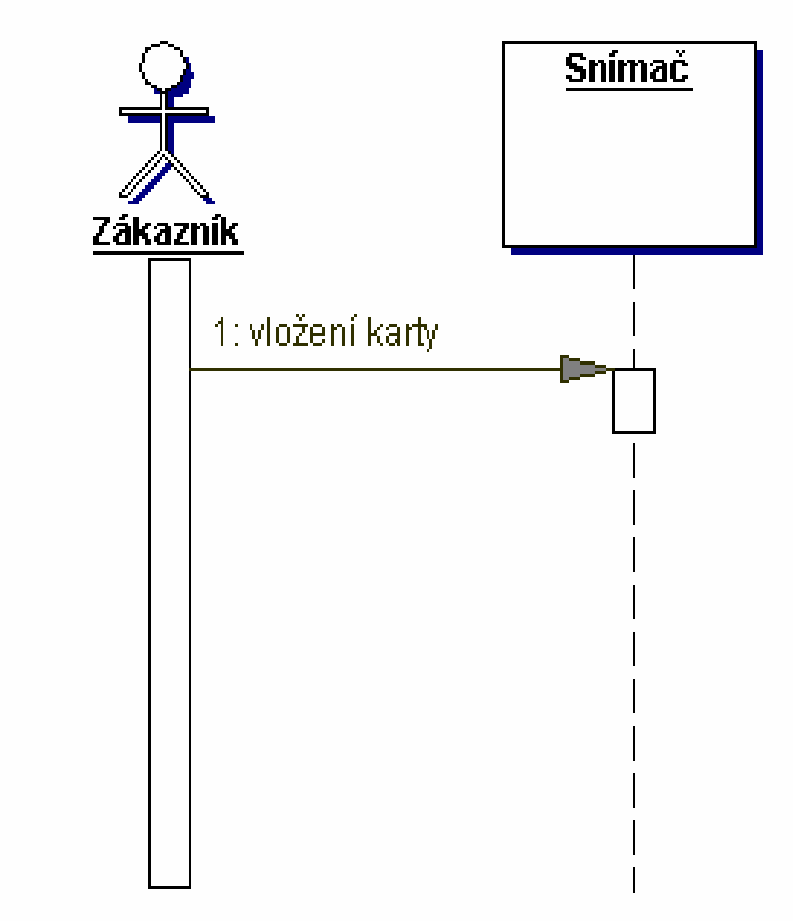
Reakce a návratové hodnoty



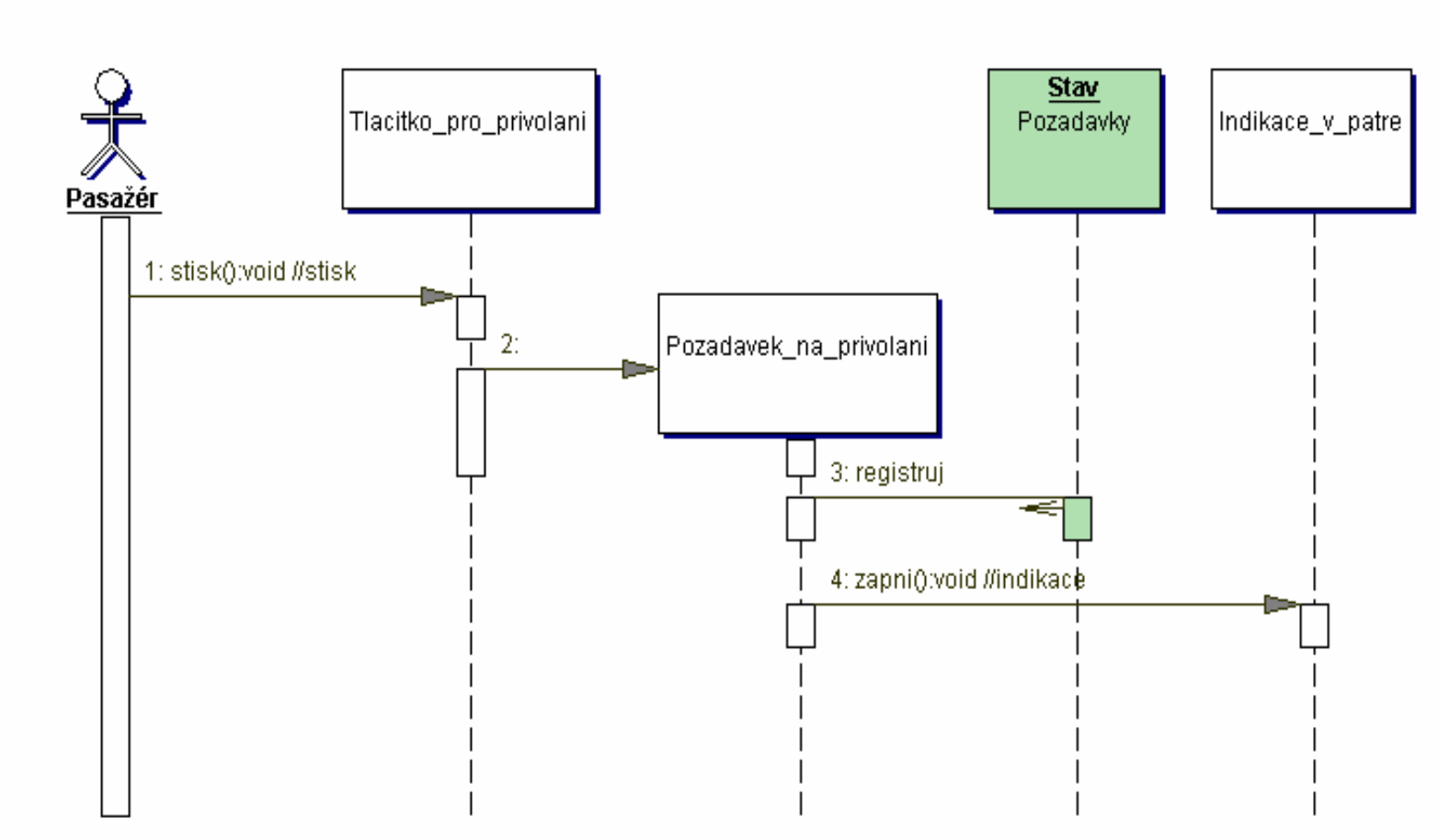
Hrubý scénář pro „čerpání“



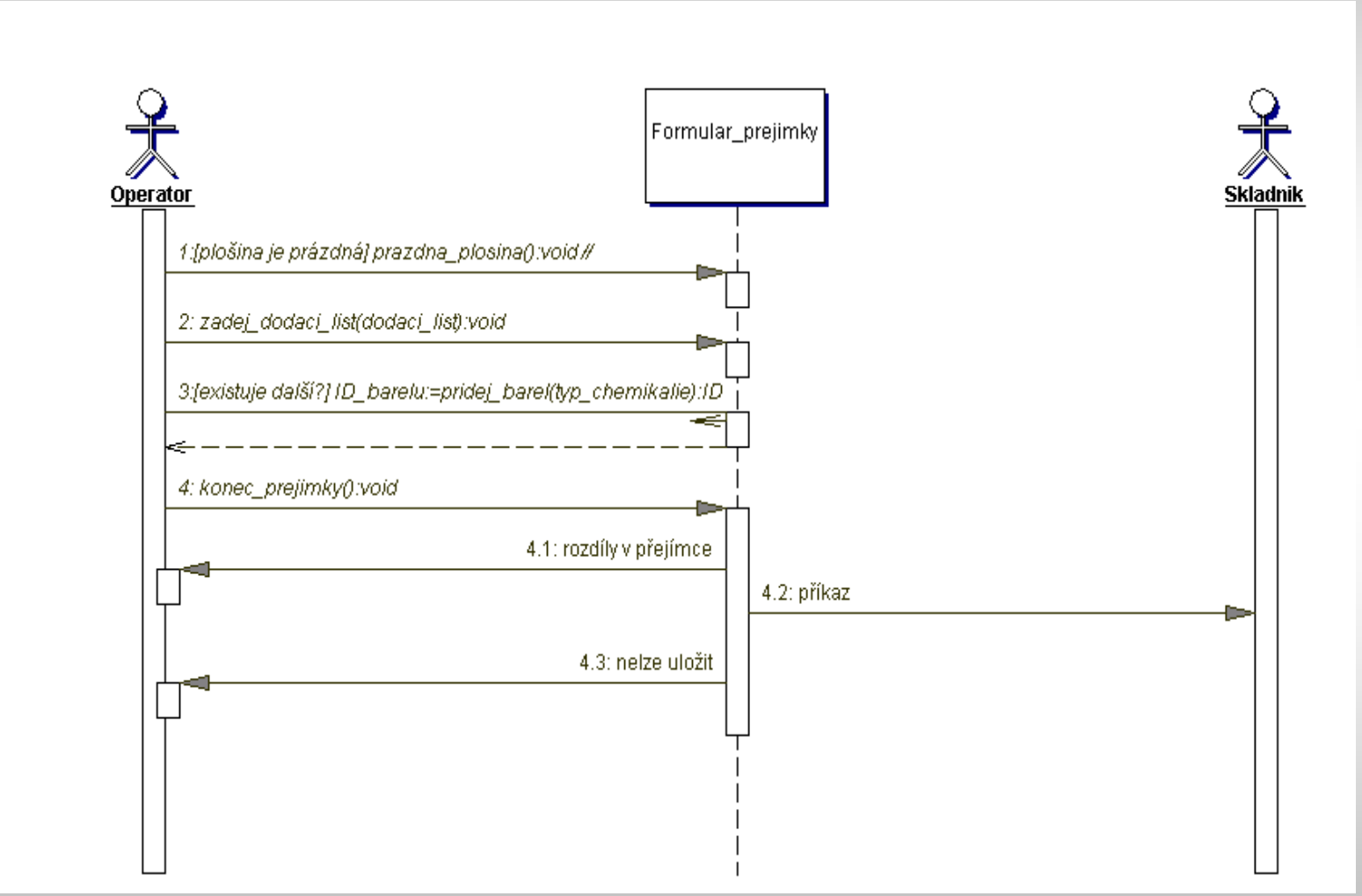
Zákazník se „autentizuje“



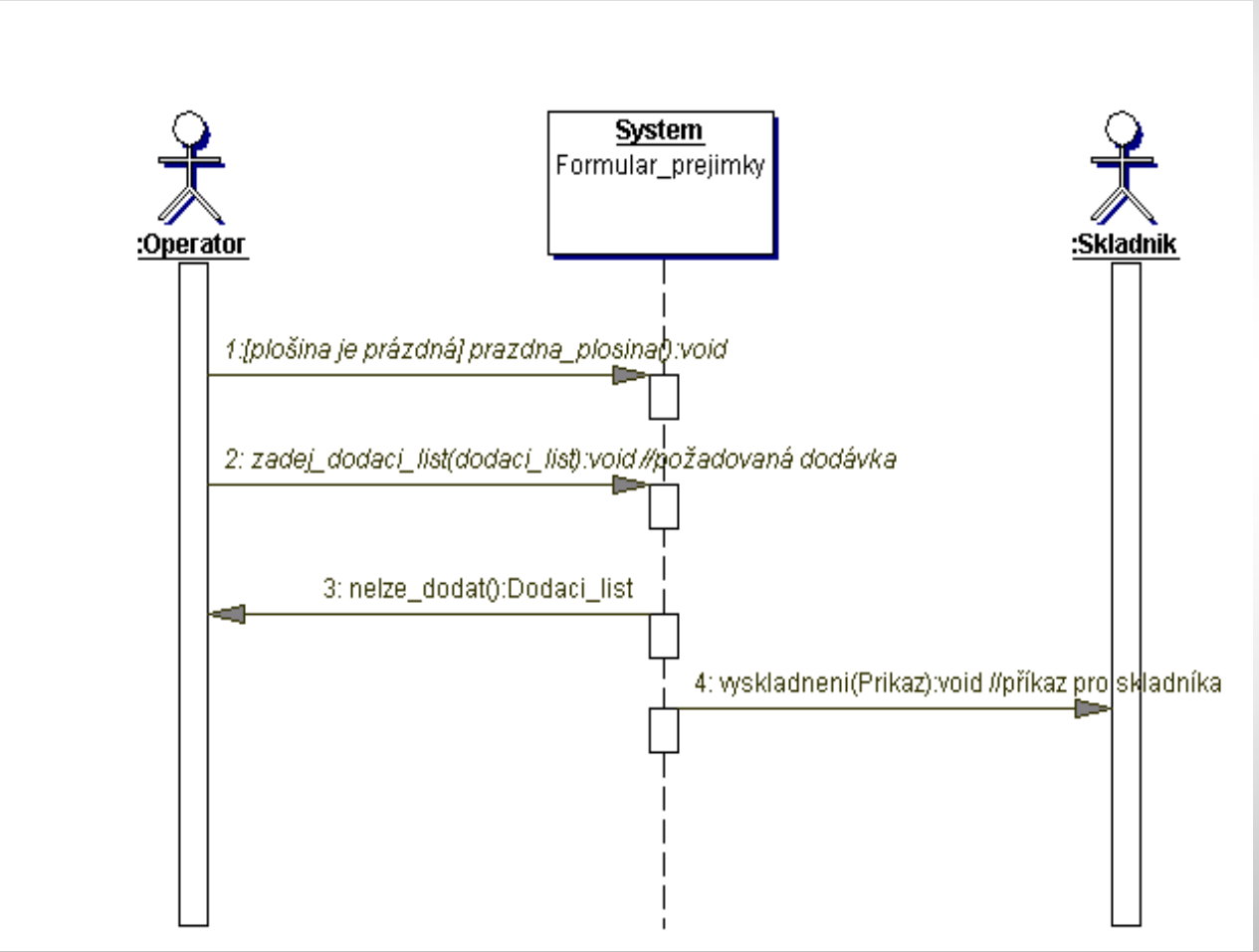
Scénář pro „přivolání“



Scénář pro “přejímku”



Scénář pro “dodávku”



Popis akce (operace, funkce)

Operation: název

Description: textový popis

Reads: jaká data jen čte

Changes: jaká data mění nebo vytváří

Sends: jaké reakce vyvolává (jaké zprávy posílá)

Assumes: co předpokládá

Results: co zajišťuje (zaručuje)

Popis pro "prázdná plošina"

Operation: prázdná plošina

Description: informuje systém, že nakládací plošina je prázdná

Reads:

Changes: plošina

Sends:

Assumes:

Results:

- vyprázdní v modelu nakládací **plošinu**
- uvolní identifikátory barelů, které jsou na **plošině**

Popis pro "zadej dodací list"

Operation: dodací list

Description: zahájí přejímku a uloží informace z dodacího listu

Reads: *supplied* dodací_list

Changes: zadaný_dodací_list

Sends:

Assumes:

Results:

- vnitřní objekt **zadaný_dodací_list** je inicializován hodnotami z fyzického **dodacího_listu**

Popis pro "barel k zařazení"

Operation: barel k zařazení

Description: každý vyložený barel je jednoznačně identifikován

Reads: *supplied* typ_chemikálie

Changes: plošina, *new* b: Barel

Sends: operátor:{ID barelu}

Assumes:

Results:

- nakládací plošina obsahuje barel b
- operátor dostane identifikaci **ID barelu**
- atribut **b.typ** je nastaven na **typ_chemikálie**
- atribut **b.ID** je nastaven na identifikaci **ID barelu**

Popis pro "konec přejímky"

Operation: konec přejímky

Description: operátor informuje systém, že již byly vyloženy všechny barely

Reads: zadaný_dodací_list

Changes: plošina, budovy ve skladu

Sends: operátor:{rozdíly v přejímce, nelze uložit}, skladník:{příkaz pro skladníka}

Assumes:

- **sklad** je bezpečný

Popis pro "konec přejímky" (pokrač.)

Results:

- pro všechny barely, které lze do **skladu** umístit, přesune v modelu jejich umístění do vhodné **budovy** a vytvoří **príkaz pro skladníka(kam: alokační seznam)**
- pokud existují rozdíly mezi **zadaným_dodacím_listem** a skutečnou dodávkou, vytvoří se **rozdíly v přejímce(navíc, chybí: seznam barelů)**
- pro všechny barely, které nelze do skladu umístit vytvoří **nelze uložit(co: seznam barelů)**
- **sklad** je bezpečný

Další postup

- Z datového modelu se snažíme odvodit funkce:
  - Vytvoříme matici CRUD (Create, Read, Update, Delete)
  - Zkoumáme, zda pro každý typ dat existuje odpovídající funkce
- Z datového modelu se snažíme odvodit dynamiku:
  - Pro každý typ dat zkoumáme, zda objekty nevykazují změny stavu

Matice CRUD

- Řádky odpovídají typům objektů.
- Sloupce odpovídají funkcím.
- V průsečíku je zapsáno zda funkce C,R,U a/nebo D odpovídající data.
- V každém řádku by mělo někde být vše (některá funkce musí objekt vytvářet, jiná využívat, či rušit).

Matice CRUD pro ECO sklad

	Prázdná plošina	Zadej dodací list	Zařaď barel	Konec přejímky	Dodávka	Zahájení práce systému ECO sklad	Ukončení práce systému ECO sklad
Plošina	U		U		U	C	D
Sklad				U	U	C,Get	D,Save
Monitor				U,Print	U,Print	C	D
Barel			C				
Dodací list		C		R,D			
Příkaz				C,Print	C,Print		

Co jsme zjistili?

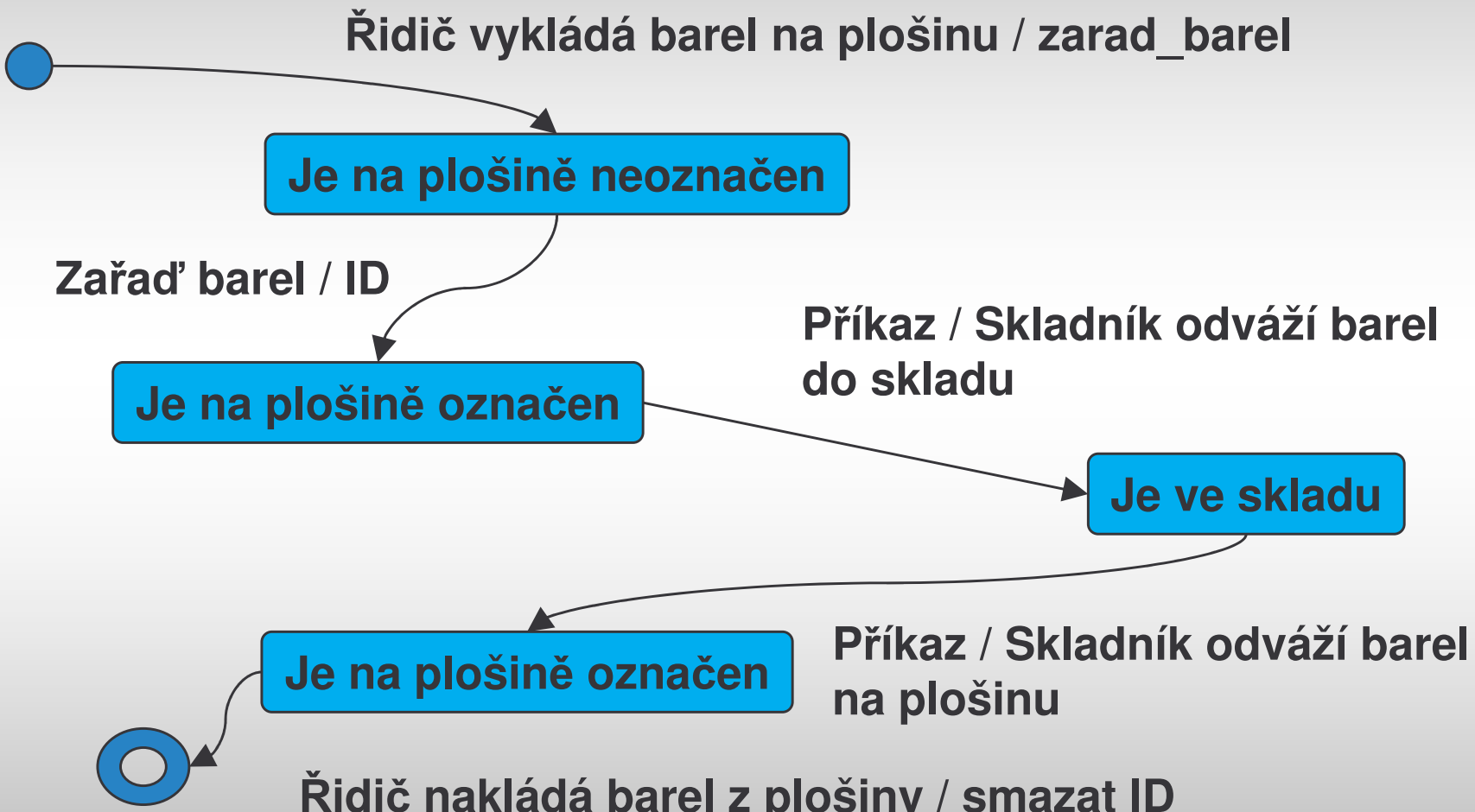
- Potřebujeme ještě v rámci nějaké funkce reprezentaci barelu zrušit.
- Mohla by to udělat funkce „dodávka“, neboť po vyskladnění barelu jeho životní cyklus končí.
- Doplníme tedy do popisu funkce dodávka požadavek „pokud v rámci dodávky využijeme některý barel, vymažeme jeho reprezentaci z obsahu skladu a zrušíme ji“.
- Do matice CRUD přidáme odpovídající D.

Dynamický model

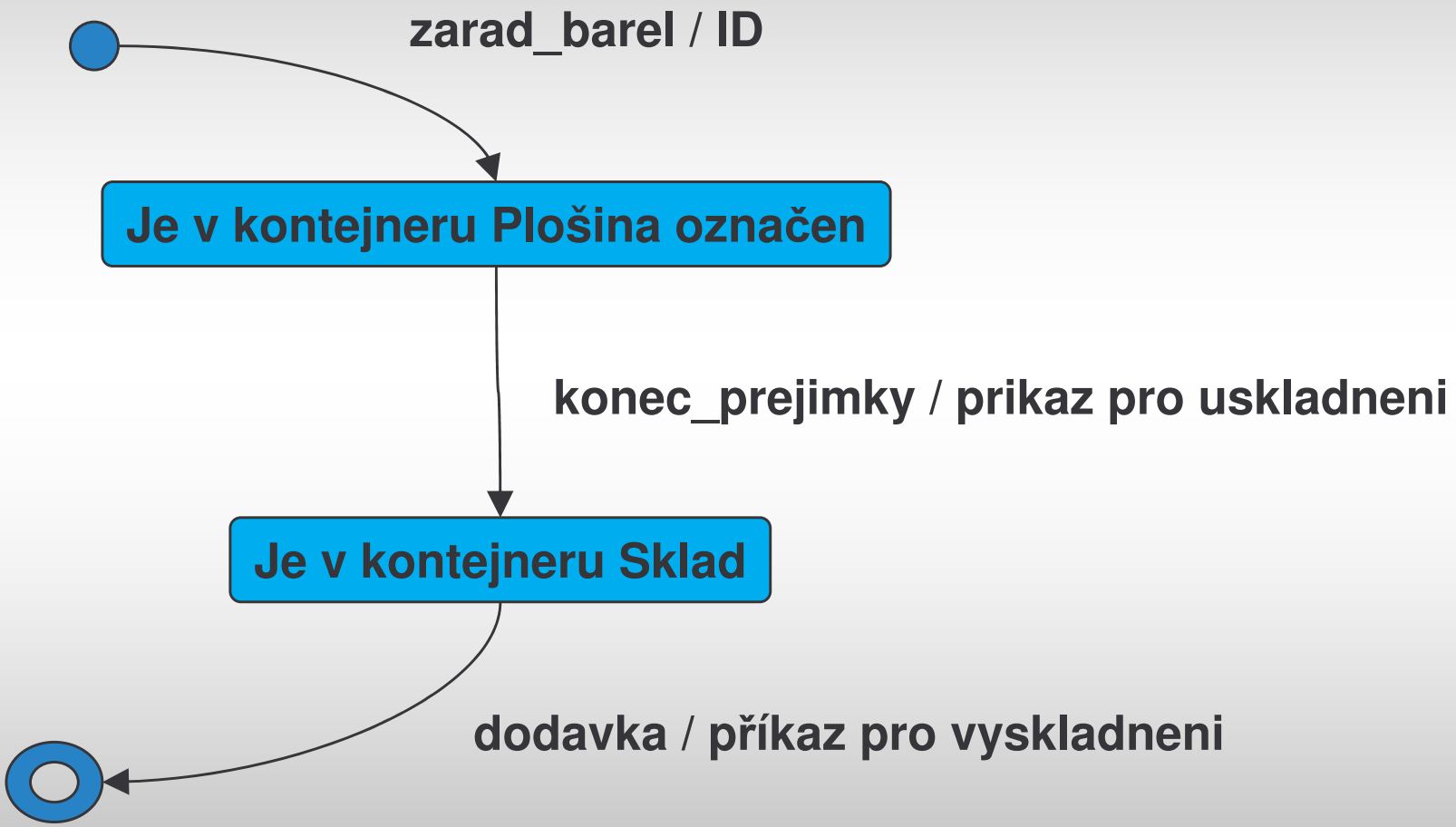
Stavové diagramy

- Slouží k popisu dynamiky systému
- Stavový diagram definuje možné **stavy**, možné **přechody** mezi stavy, **události**, které přechody iniciují, **podmínky** přechodů a **akce**, které s přechody souvisí
- Stavový diagram lze použít pro popis dynamiky objektu (pokud má rozpoznatelné stavy), pro popis metody (pokud známe algoritmus), či pro popis protokolu (včetně protokolu o styku uživatele se systémem)

Životní cyklus skutečného „barelu”

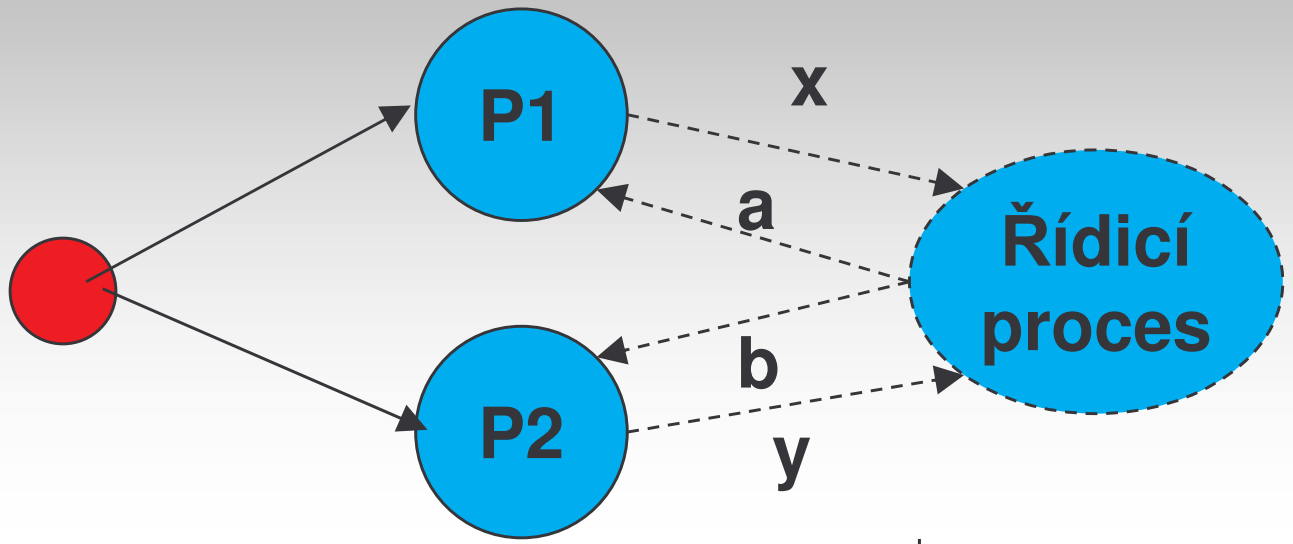


Životní cyklus entity „barel”

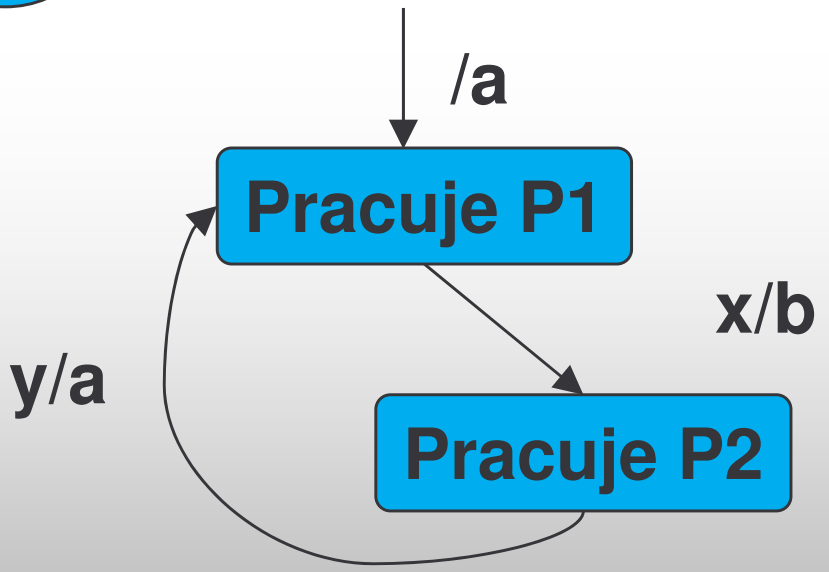


Popis řídicích procesů pomocí stavových diagramů

- Vstupy řídicího procesu lze modelovat pomocí událostí stavového diagramu.
- Výstupy řídicího procesu lze modelovat pomocí akcí stavového diagramu.
- Pak lze řídicí procesy modelovat stavovými diagramy.



střídavě spouštíme P1 a P2



Životní cyklus systému

- Vyjádření souhrné dynamiky systému, která je zachycena ve scénářích
- Definuje povolené návaznosti akcí a reakcí
- Představuje hrubou “uživatelskou příručku” pro systém
- Definice systému jako “konečného automatu”

Životní cyklus jako regulární výraz

<Životní cyklus> = Lifecycle <jméno objektu> : <regulární výraz>

<regulární výraz> =

<akce>	
| #<reakce>	
| <regulární výraz>. <regulární výraz>	sekvence
| [<regulární výraz>]	volitelně
| <regulární výraz>*	iterace
| (<regulární výraz> | <regulární výraz>)	selekce
| (<regulární výraz> || <regulární výraz>)	paralelně

<akce> = jméno události

<reakce> = jméno reakce

Životní cyklus “ECO-skladu”

Lifecycle ECO-sklad:
(dodávka | přejímka)* || (dotaz na stav | je bezpečný?)*
přejímka = prázdná plošina. dodací list.
(barel k zařazení. #ID barelu)*.
konec přejímky. [#rozdíly v přejímce].
#příkaz pro skladníka . [#nelze uložit]
dodávka = prázdná plošina.požadovaná dodávka.
#skutečná dodávka. #příkaz pro skladníka
dotaz na stav = …
je bezpečný? = ...

Životní cyklus entity „barel”

Lifecycle BAREL:

zarad_barel . #ID barelu . #příkaz pro uskladnění . dodávka . #příkaz pro vyskladnění

Kontroly analytických modelů

Výstup analýzy

Konceptuální model:

- **datový model** popisuje entity, atributy, vztahy, integritní omezení,
- **funkční model** popisuje služby, které systém poskytuje pro záznam, údržbu a využití dat,
- **dynamický model** popisuje možné stavy dat a jejich změny.

Kontrola výstupů analýzy:

- kontrola jednotlivých modelů (pohledů)
- kontrola vzájemné konzistence modelů

Kontrola datového modelu

- je datový model úplný?
  - existuje entita pro každý typ objektu?
  - nejsou zde nadbytečné entity (entity tvořené pouze identifikací, entity s jedinou instancí, apod.)?
  - jsou zde zaneseny všechny vztahy (včetně generalizací a agregací)?
  - nejsou zde odvoditelné vztahy?
  - je model v normální formě?
  - jsou zanesena všechna integritní omezení?

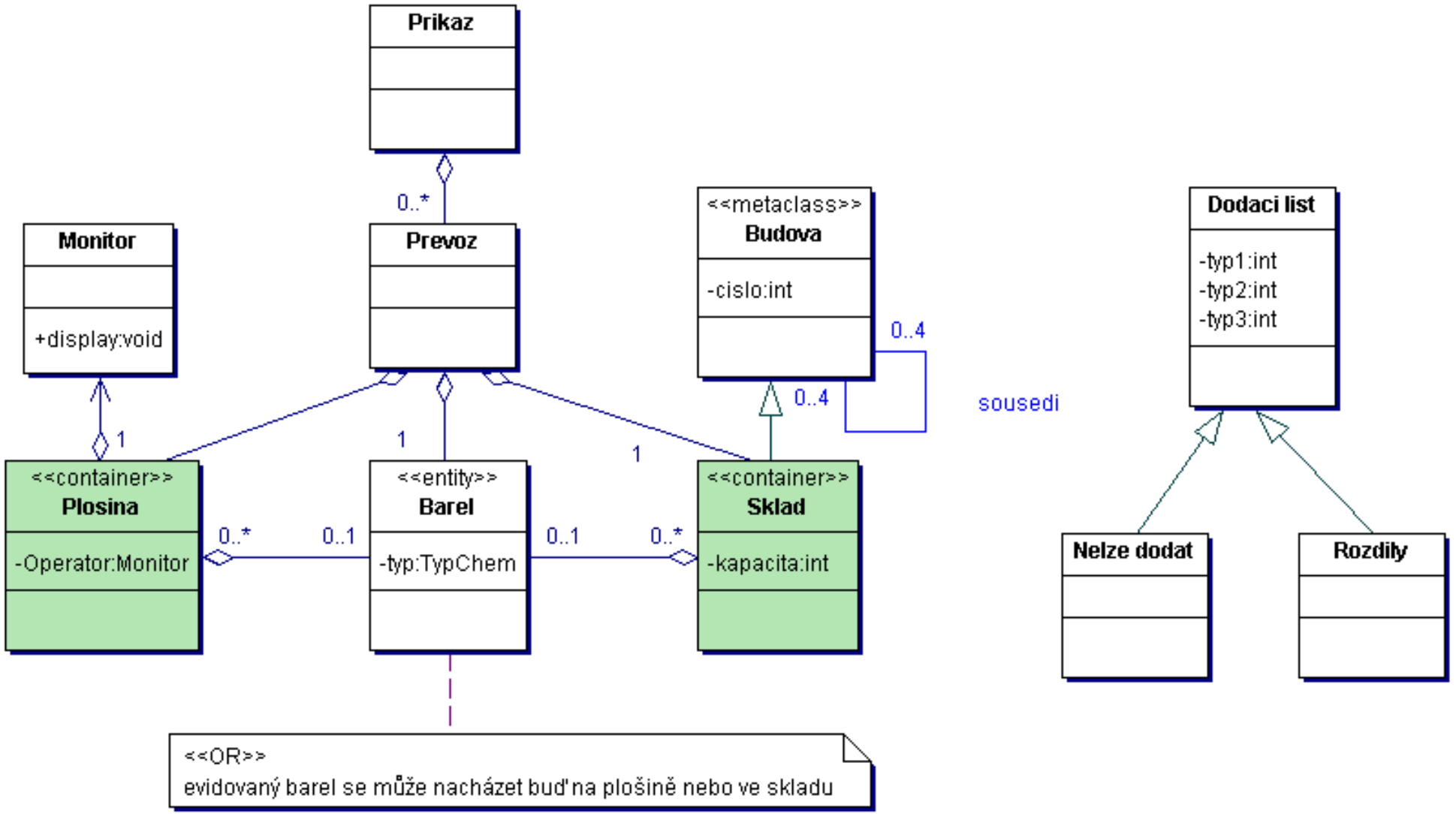
Nadbytečné entity

- entity tvořené pouze identifikací
- entity s jedinou instancí
- entity s vazbou typy 1:1
- apod.
  - Dobrou technikou je představit si příklady entit a objektů?

Jsou zaneseny všechny vztahy?

- Nelze doplnit generalizace?
- Nelze doplnit agregace?
- Nelze model vylepšit?
- Příklad: Pro entitu „dodací list“ lze vymyslet pružnější model, který usnadní případné úpravy v budoucnosti

Datový model pro ECO-sklad



Nejsou zde odvoditelné vztahy?

- Zákazník si objednává zboží
- Zákazníkovi je vystavena faktura.
- Odebrané zboží je předmětem fakturace.
- ? Nejsou zde odvoditelné vztahy?
- Pozn.: Odvoditelné vztahy mohou v modelu být, ale musí být jako odvoditelné předznačeny znakem „/“ a doplněny způsobem odvození (formulí, popisem v OCL).

Jsou zanesena všechna integritní omezení?

Řadu vlastností dat nelze do diagramu zanést:

- Šéf musí mít větší plat než jeho podřízení.
- V jednom skladu nelze umístit chemikálie typu „1“ a „2“.

```
context s:Sklad inv : forAll(Barel x,y |
   s.obsahuje(x) and s.obsahuje(y)
   implies x.typ != 1 or y.typ != 2)
```

Vyvážení datového modelu

- datový model versus datový slovník
  - každá entita, atribut a vztah v DD
- datový model versus funkční dekompozice
  - každá paměť a datový tok obsahuje entitu, atribut nebo vztah (nebo jejich kombinaci)
- datový model versus minispecifikace
  - něco musí entity a vztahy vytvářet/rušit, číst/modifikovat (matice CRUD)

Kontrola funkčního modelu

- je funkční model úplný?
  - existuje funkce/metoda pro každou událost?
  - každá funkce/metoda musí být popsána dekompozicí, nebo mít minispecifikaci (vstupy a výstupy musí odpovídat)
  - nejsou zde nadbytečné funkce/metody?

Vyvážení funkčního modelu

- funkční model versus datový slovník
  - každá paměť a datový tok v DD
  - každý prvek DD se někde vyskytuje (jinak je zbytečný)
- funkční model versus datový model
  - každá data zmíněná ve funkce/metodě musí být popsána v datovém modelu
- funkční model versus dynamický model
  - každý řídicí proces má dynamický model (vstupy = podmínky, výstupy = akce)

Kontrola dynamického modelu

- je dynamický model úplný?
  - existuje model pro každou entitu, která může mít různé stavy?
  - existuje model pro každý řídicí proces?
  - existuje popis životního cyklu systému?

The End