X36SIN: Softwarové inženýrství

Karel Richta

2+2, kl.z.

Co to je „softwarové inženýrství“ ?

(definice IEEE 1993)

„Softwarové inženýrství je systematický, disciplinovaný a kvalifikovaný přístup k vývoji, tvorbě a údržbě softwaru.“

Proč se SI na FEL učí?

- Protože „softwarové inženýrství“ patří ke standardní výbavě absolventů universit.
- Absolventi FEL by neměli být pozadu a měli by se umět domluvit s absolventy jiných škol.
- Zdá se, že tato profese bude ještě dlouho žádaná.

Jak se SI na FEL učí?

- Projektová forma výuky
- Projekty do výuky patří (kromě klasické výuky)

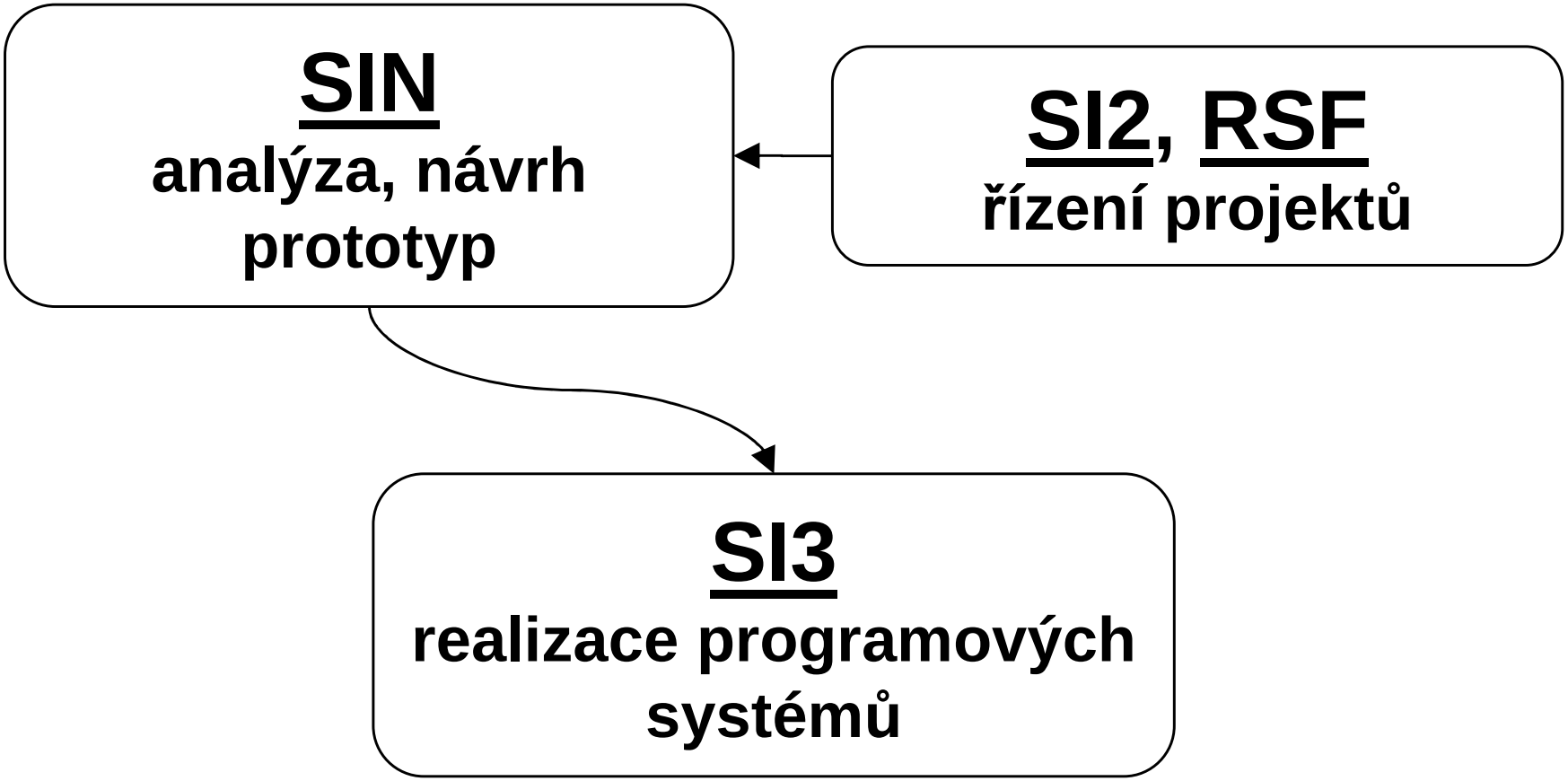
„Tradiční učení by nemělo být dominantní způsob výuky na universitě, která se chce zabývat výzkumem.“

MIT Educational Design Project, 1999

Smysl předmětů řady SI

- SIN je základní kurz softwarového inženýrství, který je určen pro pochopení discipliny, získání základních dovedností v analýze a návrhu, seznámení s používanými technikami a nástroji.
- V rámci cvičení se řeší menší projekty v týmech. Smyslem je vyzkoušet si práci na projektu řešeném v týmu. Výstupem projektů je předepsaná projektová dokumentace.
- Na předmět SIN (Softwarové inženýrství), který se zabývá zejména modelováním, navazuje předmět SI3 (Realizace programových systémů), který se zabývá realizací.
- Paralelně s SIN běží předmět SI2, resp. RSF (Řízení softwarových projektů), který se zabývá řízením projektů. Studenti SI2 řídí studenty SIN, studenti RSF provádějí audit projektů.

Návaznost předmětů SI



Co je to projekt?

- *„Projekt je dobře definovaná posloupnost činností, která je zaměřena na dosažení nejistého cíle, má určen začátek a konec, a je uskutečňována pomocí zdrojů – lidí a prostředků.“*
- *„Projekt je specifická nerutinní akce, která proto vyžaduje plánování.“*
- *„Čím složitější je projekt, tím více vyžaduje plánování.“*

Co je to softwarový projekt?

- *„Softwarový projekt je projekt, jehož cílem je vytvoření nebo využití programového díla.“*
- *„Při uskutečňování SW projektů se uplatní SW inženýři, kteří nabyli znalostí v předmětech SI (mimo jiné také v předmětu SIN).“*

Jaké jsou softwarové projekty v předmětu SIN?

- Větší projekt se za semestr nestihne (zkušenost).
- Projekt představuje hodně práce – projekty je třeba řešit v týmech (navíc se učíme týmovou práci a zodpovědnost).
- Projekt je třeba ukončit prezentací a obhajobou (prezentace práce představuje důležitou praktiku).
- Také metodika posouzení jiného projektu přináší nové poznatky.

Co z toho vyplývá?

- Studenti jsou rozděleni do týmů. Studenti SIN představují řešitele. Studenti předmětu SI2 fungují jako manažeři projektů, testéři kvality a pod. Studenti RSF provádějí audit.
- Vedoucí projektu (cvičící) může na základě osobnostního testu definovat strukturu týmu. Případně podle dohody volí pouze šéfa týmu a ostatní role v týmech definuje šéf.
- Tým řeší projekt, který si zvolí a dohodne, nebo který mu byl přidělen.
- Cvičení jsou projektová – konají se tak, jak to odpovídá potřebě projektů, příp. tak, jak stanoví cvičící dle potřeby projektů!

SIN – Jak získat známku

- Předmět SIN je zakončen klasifikovaným zápočtem.
- Zápočet uděluje cvičící - vedoucí projektu, příp. přednášející na základě doporučení vedoucího, pokud vedoucí nemá příslušné oprávnění.
- Zápočet se uděluje na základě akceptovaného projektu a písemných testů.
- Známka je určena kvalitou projektu (0 až 100 bodů), výsledky testů (20 až 50 bodů) a osobním hodnocením (-20 až 20 bodů).

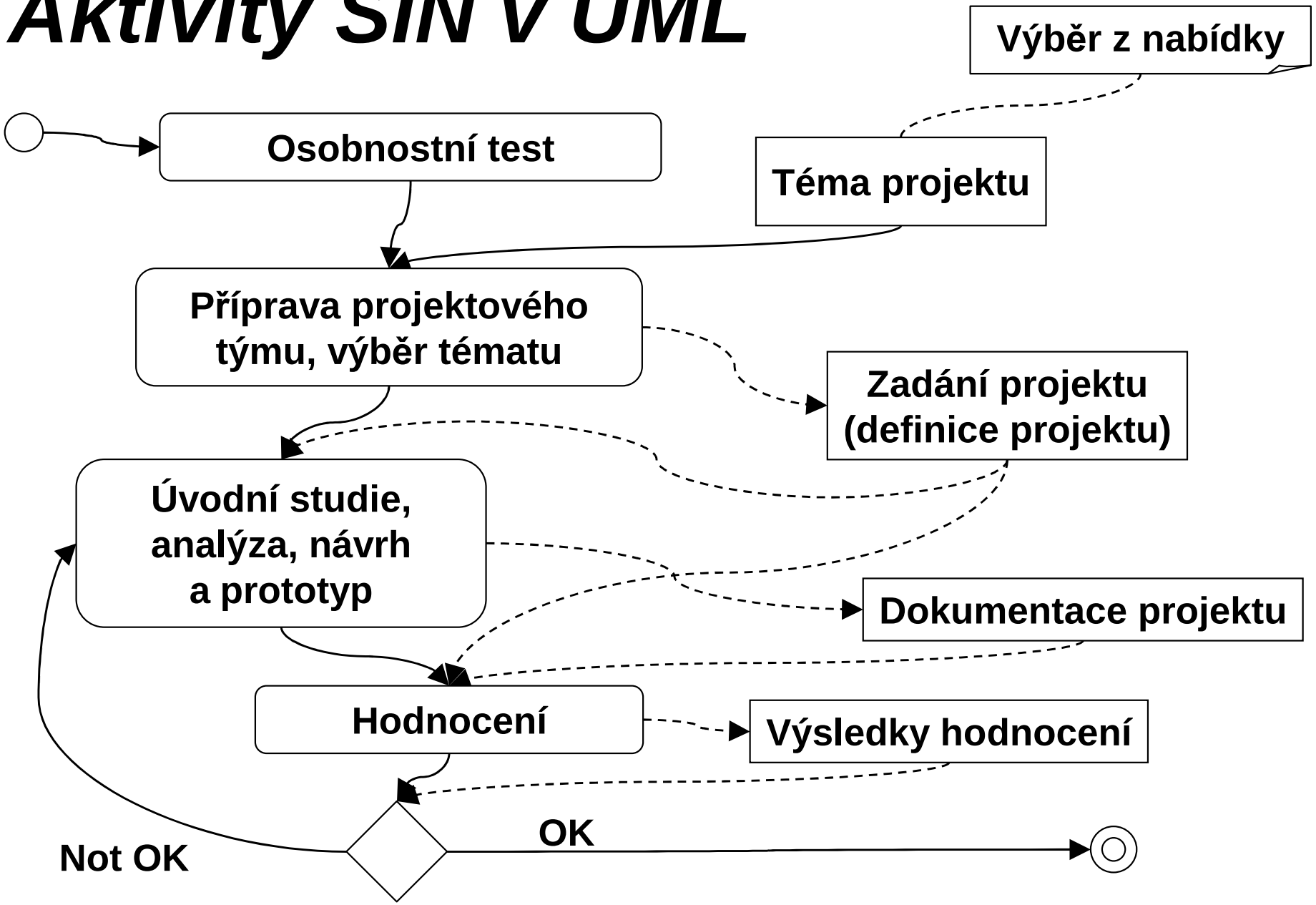
Pro získání zápočtu je třeba:

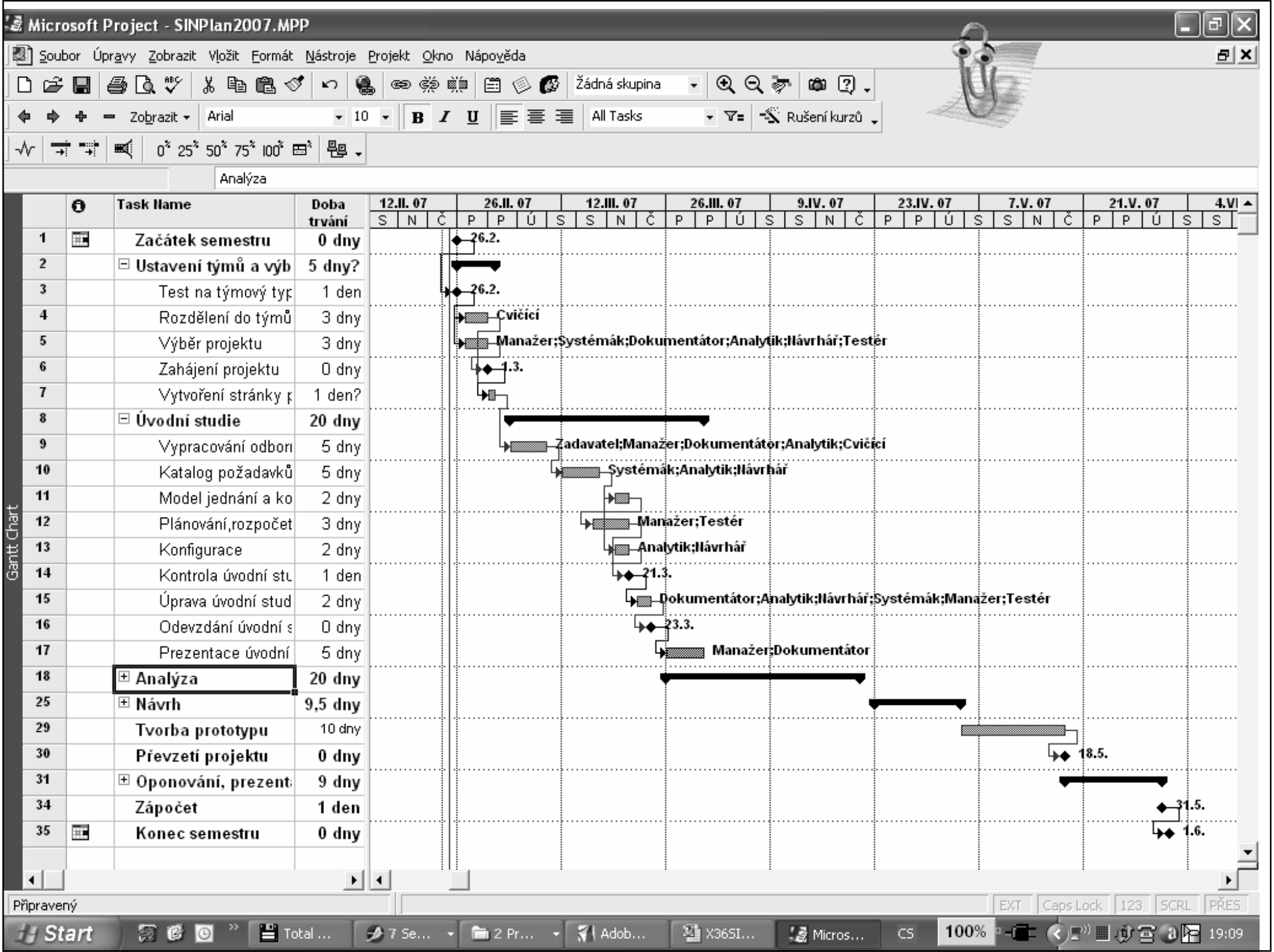
- Vytvořit projekt, který projde hodnocením (s kladným výsledkem). Projekty hodnotí:
  - Cvičící
  - Zadavatel
  - Nezávislý hodnotitel
  - Kdokoliv
- Ohodnotit jiný projekt
- Absolvovat 2 testy alespoň s minimálním hodnocením (10 bodů pro každý test).

Stupnice:

Body od	Body do	Známka
-20	69	-
70	89	3
90	109	2
110	160	1

Aktivity SIN v UML

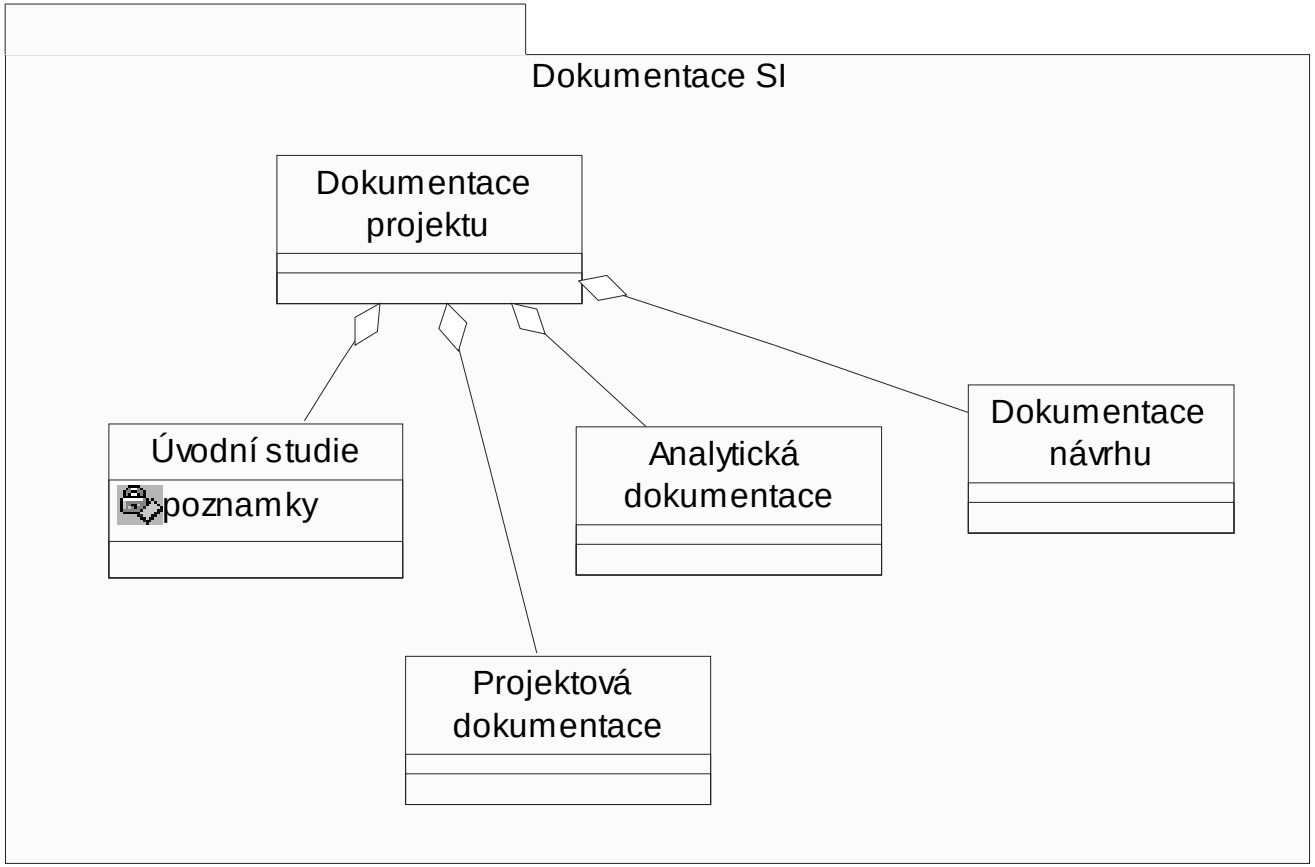



Microsoft Project - SINPlan2007.MPP
Soubor Úpravy Zobrazit Vložit Formát Nástroje Projekt Okno Nápověda
Žádná skupina
Zobrazit Arial 10 All Tasks Rušení kurzů
Analýza
Gantt Chart
Task Name
Doba trvání
1 Začátek semestru 0 dny
2 Ustavení týmů a výb 5 dny?
3 Test na týmový typ 1 den
4 Rozdělení do týmů 3 dny
5 Výběr projektu 3 dny
6 Zahájení projektu 0 dny
7 Vytvoření stránky p 1 den?
8 Úvodní studie 20 dny
9 Vypracování odbor 5 dny
10 Katalog požadavků 5 dny
11 Model jednání a ko 2 dny
12 Plánování,rozpočet 3 dny
13 Konfigurace 2 dny
14 Kontrola úvodní stu 1 den
15 Úprava úvodní stud 2 dny
16 Odevzdání úvodní s 0 dny
17 Prezentace úvodní 5 dny
18 Analýza 20 dny
25 Návrh 9,5 dny
29 Tvorba prototypu 10 dny
30 Převzetí projektu 0 dny
31 Oponování, prezent 9 dny
34 Zápočet 1 den
35 Konec semestru 0 dny
12.II. 07
26.II. 07
12.III. 07
26.III. 07
9.IV. 07
23.IV. 07
7.V. 07
21.V. 07
26.2.
Cvičící
Manažer;Systémák;Dokumentátor;Analytik;Návrhář;Tester
1.3.
Zadavatel;Manažer;Dokumentátor;Analytik;Cvičící
Systémák;Analytik;Návrhář
Manažer;Tester
Analytik;Návrhář
21.3.
Dokumentátor;Analytik;Návrhář;Systémák;Manažer;Tester
23.3.
Manažer;Dokumentátor
18.5.
31.5.
1.6.
Připravený
EXT Caps Lock 123 SCRL PŘES
Start
Total ...
7 Se...
2 Pr...
Adob...
X36SI...
Micros...
CS
100%
19:09

Co to znamená:

- Realizovat projekt znamená:
  - ***prostudovat téma***
  - ***vypracovat popis problému***
  - ***vytvořit potřebnou dokumentaci***
  - ***vytvořit prototyp produktu***
  - ***absolvovat požadované inspekce***
  - ***prezentovat projekt***
- Ohodnotit jiný projekt znamená:
  - ***být přítomen při inspekcích cizího projektu***
  - ***prostudovat dokumentaci cizího projektu***
  - ***sepsat posudek***

Obsah dokumentace SIN



Přibližná osnova cvičení:

- **Osobnostní test, rozdělení do týmů, témata projektů, výběr projektu**
- **Vypracování zadání projektu - odborný článek**
- **Vypracování úvodní studie (CIM), plán řešení**
- **Inspekce úvodní studie, příp. přepracování**
- **Vypracování analytické specifikace (PIM)**
- **Inspekce analytické specifikace**
- **Návrh (architektura, uživatelský vzhled, dekompozice na komponenty - PSM, plán realizace projektu)**
- **Vytvoření prototypu**
- **Prezentace**
- **Posouzení cizího projektu**

Důležité termíny

- 1.3. – zahájení projektu
- 23.3. – úvodní studie (CIM)
- 20.4. – analýza (PIM)
- 1.5. – návrh (PSM)
- 18.5 - prototyp, odevzdání projektu
- 31.5. - odevzdání posudku, zápočet

Stránky předmětu SIN:

https://service.felk.cvut.cz/courses/X36SIN

http://ocw.cvut.cz/moodle/

Literatura

- Tyto přednášky
- Richta, Sochor: Softwarové inženýrství I. Skripta FEL ČVUT, Praha 1998 (bohužel již vyprodána).
- Vrana, I., Richta, K.: Zásady a postupy při zavádění podnikových informačních systémů. Grada, Praha 2004.
- Arlow, Neustadt: UML a unifikovaný proces vývoje aplikací. Computer Press, Praha 2003.
- Schmuller: Myslíme v jazyce UML. Grada, Praha 2001.
- Kanisová, Müller: UML srozumitelně. Computer Press, Brno 2004.
- Šešera, Mičovský, Červeň: Datové modelování v příkladech. Grada 2001.

Další zdroje

- Bieliková: Softvérové inžinierstvo - Princípy a manažment. Skripta STU, Bratislava 2000.
- Král: Informační systémy. Science, Veletiny 1997.
- Sommerville: Software Engineering. Addison-Wesley, 2000.
- Pressman: Software Engineering. McGraw-Hill, 1994.
- Booch G., Rumbaugh J., Jacobson I.: The Unified Modeling Language User Guide, Addison Wesley Longman, 1999.
- Unified Modeling Language Specification, OMG, http://www.uml.org/
- http://www.rational.com/uml/
- http://interval.cz/

Osnova přednášek

- Úvod do softwarového inženýrství, plánování projektů
- Modelování požadavků (CIM)
- Analýza (PIM)
- Architektura SW, MDA
- Návrh (PSM)
- Návrhové vzory
- Metodiky
- Testování
- Další novinky pro vývoj: WS, SOA, EJB

Úvod do softwarového inženýrství

Úvodem trochu historie I.

- Termín „**software**“ zavedl v roce 1958 statistik John Tukey (také autor termínu „bit“).
- Za okamžik zrození termínu „**softwarové inženýrství**“ se obvykle považuje rok 1968, kdy NATO sponzoruje první konferenci s tímto názvem a na toto téma.
- V roce 1969 na ni navázala konference „Techniky softwarového inženýrství“.
- V roce 1972 vychází první časopis „Transactions on Software Engineering“ (IEEE Computer Society).
- V roce 1976 vytváří IEEE Computer Society první komisi, která by měla definovat obsah oboru „softwarové inženýrství“.

Proč to vůbec vzniklo?

- Něco bylo špatně ☹
- Počítačů přibývalo, přibývalo i softwarových projektů, ale ubývalo úspěšně dokončených projektů
- Někdy to došlo až na hranici únosnosti – software byl, nebo mohl být, příčinou havárií (např. Mariner I.)

Další historie II.

- Organizátoři první konference „**Softwarové inženýrství**“ v roce 1968 zvolili termín „softwarové inženýrství“úmyslně jako provokativní – naznačující, že produkce software musí přejít na jiné postupy a být podložena teoretickými disciplinami, podobně, jako je tomu u inženýrského přístupu v jiných oborech.
- Konala se ve Spolkové republice Německo ve známém středisku Garmisch-Partenkirchen a řídil ji profesor Bauer z Mnichovské techniky.
- Účastnilo se jí asi 50 odborníků z různých oblastí, z praxe i ze škol. Její účastníci formulovali směry, kterými by se výzkum v oboru SI měl ubírat.

Termín softwarové inženýrství

- ***„Softwarové inženýrství je disciplina, která se zabývá zavedením a používáním řádných inženýrských principů do tvorby software tak, abychom dosáhli ekonomické tvorby software, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.“***

Konference „Softwarové inženýrství 1968“

Termín softwarové inženýrství

- ***„Softwarové inženýrství je disciplina, která se zabývá zavedením a používáním řádných inženýrských principů do tvorby software tak, abychom dosáhli ekonomické tvorby software, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.“***

Konference „Softwarové inženýrství 1968“

Termín softwarové inženýrství

- ***„Softwarové inženýrství je disciplina, která se zabývá zavedením a používáním řádných inženýrských principů do tvorby software tak, abychom dosáhli ekonomické tvorby software, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.“***

Konference „Softwarové inženýrství 1968“

Další historie III.

- V roce 1979 vytváří Fletcher Buckley standard IEEE 370 pro vytváření plánů zajišťujících kvalitu software.
- V roce 1986 vzniká standard IEEE 1002, který definuje taxonomii softwarově inženýrských standardů.
- 1990 – 1995 vznikají standardy pro proces životního cyklu software (Standard for Software Life Cycle Processes) - standard ISO/IEC12207, vychází z DoD Std 498.
- V roce 1993 vznikají komise IEEE a ACM, které ústí do společného úsilí definovat softwarové inženýrství jako disciplinu.

Další historie IV.

- V roce 1998 společná komise IEEE a ACM definuje profesi softwarového inženýra.
- Nakonec v roce 2004 vzniká společný návrh „curricula“ pro výuku tohoto oboru, označovaného SE2004.
- Tím se završilo uznání softwarového inženýrství jako discipliny, podobně jako CS1991 završilo uznání informatiky.
- Softwarové inženýrství je definované jako standard IEEE 610.12.

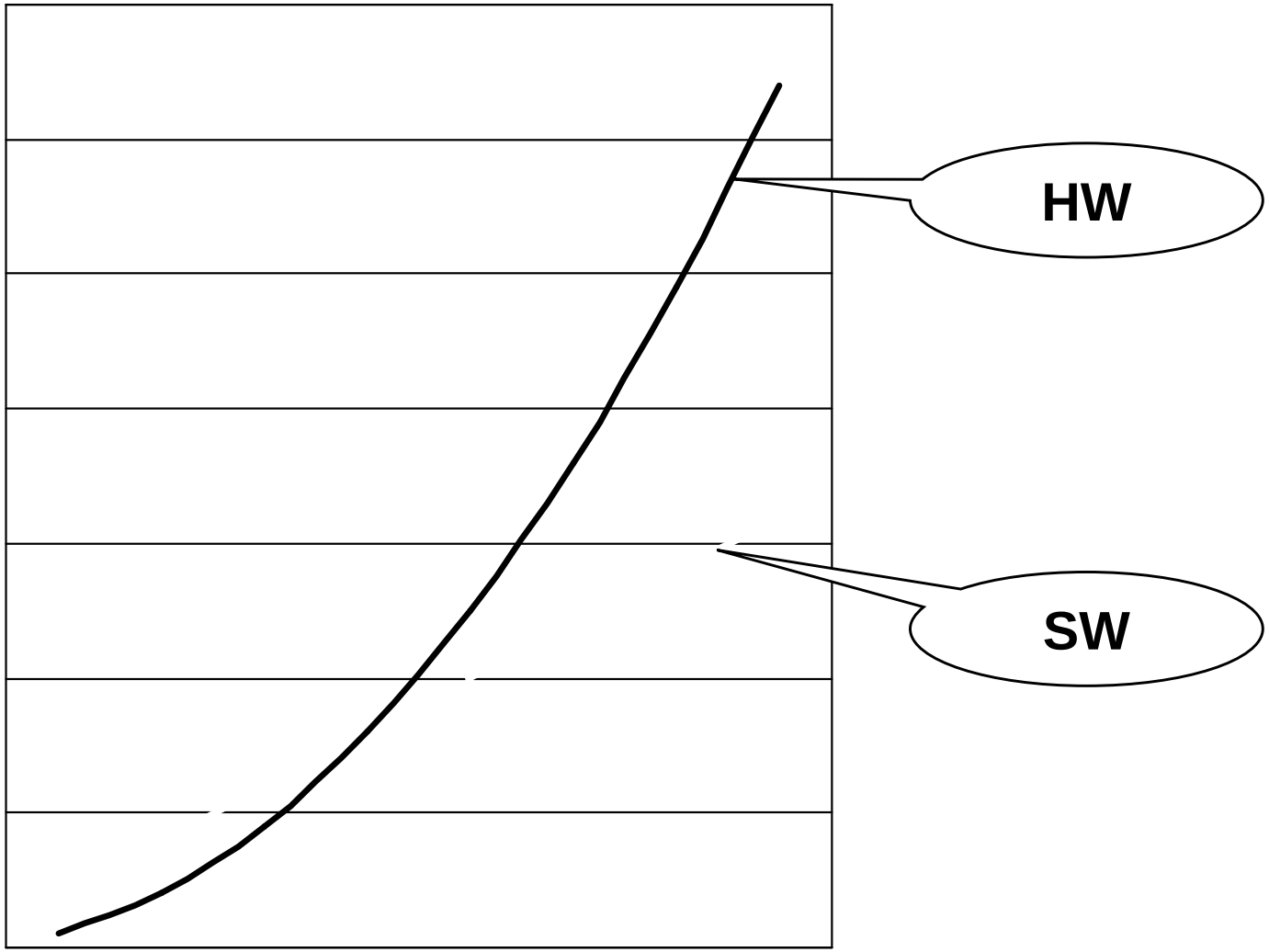
Příčina vzniku SI?

- Obvykle se říká, že to způsobil jev nazývaný „**softwarová krize**“.
- Dokud výkon počítačů nepřesáhl určitý rozměr, bylo možno se spolehnout na programátorské „hvězdy“.
- Často se počítače se využívaly pro vědecko-technické výpočty, kde záleželo spíše na preciznosti řešení, než na efektivitě tvorby programů.

Moorův zákon:

- ***„Výkon hardwaru vzrůstá zhruba dvakrát za dva roky“.***
  - Přestože sám autor prohlásil svou extrapolaci jako „pěkně divokou“, zákon zhruba platí dodnes.
  - Firma Intel nedávno zveřejnila výsledky výzkumné zprávy uvádějící, že Moorův zákon pravděpodobně přestane platit až kolem roku 2021 (křemík se dostane na hranici svých možností).

Softwarová krize



Edsger W. Dijkstra:

- ***„Hlavní příčinou softwarové krize byl nárůst výkonu hardware. Jinak řečeno, programování nemělo problémy, dokud neexistovaly počítače. Dokud jsme měli slabé počítače, mělo programování jen snesitelně těžké problémy. Nyní máme gigantické počítače a k nim gigantické problémy se softwarem“.***

Příčiny softwarové krize

- Příčinou softwarové krize vždy byl nesoulad mezi složitostí vytvářeného produktu a relativní nedostatečností a nezkušeností softwarové profese. Tento rozdíl způsobuje rozevírání nůžek a důsledkem pak jsou softwarové krize.

- *Pozn. Tento jev asi není omezen na software, ale zdá se mít univerzálnější platnost.*

Projevy softwarové krize

- Projekty překračují rozpočet.
- Projekty překračují čas.
- Software nemá dostatečnou kvalitu.
- Software neodpovídá požadavkům.
- Projekt není dobře řiditelný a software je obtížně udržovatelný.

Skončila softwarová krize?

- Při pohledu na předchozí body a současné projekty se zdá, že softwarová krize trvá stále.
- Svůj vliv zde mají též možnosti softwarových expertů, kteří jakkoli jsou geniální, mohou přímo mentálně obsáhnout jen určitý rozsah problémů.
- Řešení velkých projektů nelze proto nechat pouze na nich. Je nutno použít standardní techniku řešení obtížných problémů – „**rozděl a panuj**“ – velký problém je třeba rozdrobit na problémy menší.

Zkušenost (Brooks):

- ***„Přidání nových kapacit na zpožděný projekt prodlouží jeho řešení“.***

Tvorba software ↔ inženýrství

- Všechny tyto problémy související se softwarovou krizí vedly tedy nakonec k pokusu udělat z vývoje produkovaného nadšenci inženýrskou disciplinu.
- V 70-tých letech dochází k formulaci základních principů tohoto oboru.
- Vzniká také první generace nástrojů pro podporu této discipliny, které jsou označovány jako **CASE** (Computer Aided Software Engineering).

Co dělá inženýr?

- Než něco vyrábí, tak si to nejdříve nakreslí, namodeluje.
- K tomu potřebuje zavedenou notaci a vědecky podložené metody a postupy.
- Snaží se používat vhodné technologie tak, aby se dílo vytvořilo spolehlivě, efektivně a aby vydrželo.
- Srovnáme-li softwarového inženýra s inženýrem stavebním, pak stavební inženýr realizuje stavbu podle modelu, programátor programuje podle modelu.
- Model stavebnímu inženýrovi navrhl architekt (územní rozhodnutí, stavební povolení, realizace stavby), programátorovi softwarový architekt (úvodní studie, návrh architektury), analytik (konceptuální model) a návrhář (logický model).

První studijní programy

- Prvý inženýrský program byl vypsán již v roce 1979 na universitě v Seattlu, kde v roce 1982 udělili prvý titul v tomto oboru.
- Prvý bakalářský program v oboru SI vypsalo v roce 1996 vysoké učení technické v Rochesteru. Zpočátku byl odmítnut, ale později akreditaci získal v roce 2003 spolu s inženýrskou školou v Milwaukee.

Definice IEEE 610.12:

- ***„Softwarové inženýrství je aplikace systematického, disciplinovaného, kvantifikovatelného přístupu k vývoji, provozu a údržbě softwaru, tj. aplikace inženýrství na software. Také je to studium přístupů dle výše uvedeného.“***

Základní znalostní oblasti SI

- Správa požadavků (Software requirements)
- Softwarový návrh (Software design)
- Tvorba softwaru (Software construction)
- Testování softwaru (Software testing)
- Údržba softwaru (Software maintenance)
- Správa konfigurací (Software configuration management)
- Řízení vývoje (Software engineering management)
- Softwarový proces (Software engineering process)
- Nástroje a metody softwarového inženýrství (Software engineering tools and methods)
- Kvalita softwaru (Software quality)

Co nám SI přineslo?

Řadu novinek, namátkou některé:

- Softwarové profese a týmy.
- Modely životního cyklu.
- Oddělení správy dat od správy funkčnosti.
- Strukturované metody.
- Objektově-orientované metody.
- Komponentové metody.
- Nové programovací jazyky.
- Softwarové zápisy – unifikovaná notace UML.
- Metodiky tvorby softwaru.
- Plánování, metriky, organizace.

Co nám SI odneslo?

Pocit opravdových programátorů:

- Software vytvářejí specializovaní odborníci, kteří jediní znají postupy, nástroje, metody a techniky.
- Plánování a podobné hlouposti sem nepatří.
- Dokumentaci ať si pořizují ti, kteří neumějí číst programy přímo.

Neznámý programátor:

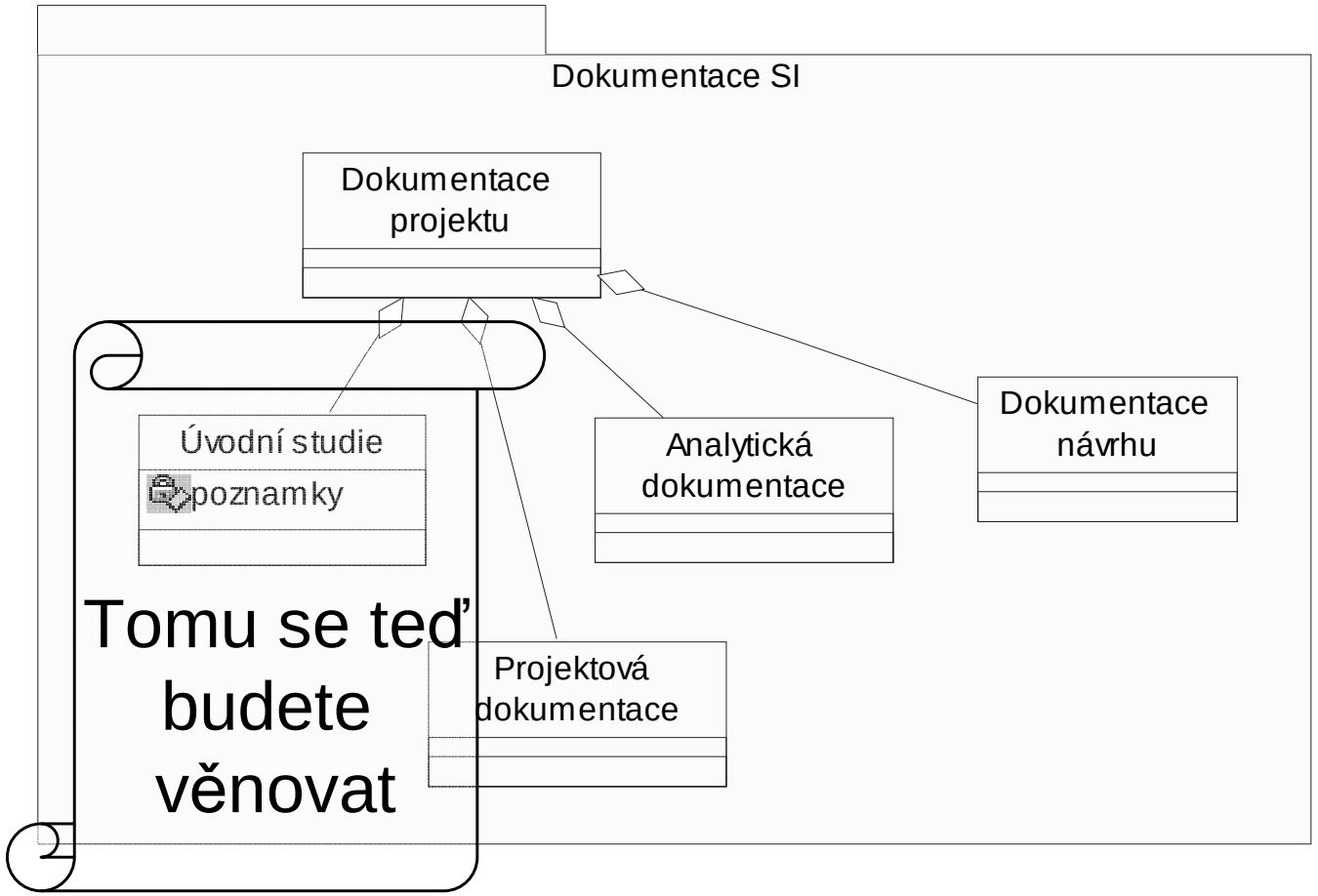
- *„Softwarové inženýrství znamená zavedení discipliny do volné tvorby software. Žádný opravdový programátor ho proto nemůže mít příliš v lásce, neboť jej nutí vytvářet nesmyslnou dokumentaci a další podobné artefakty.“*

Úvodní studie (feasibility study)

Odpověď na otázku ZDA a PROČ

Sběr požadavků na SW produkt

Obsah dokumentace SI



SIN: Zadávací dokumentace

- Měla by představovat zadání projektu pro řešitelský tým
- Měla by proto obsahovat deklaraci záměru a odborný článek
- Bude podkladem pro úvodní studii, kterou vypracují řešitelé – ta musí odpovědět na otázku “vyplatí se projekt řešit?”

Úvodní studie

Měla by odpovědět na otázku PROČ?

- Musí odpovědět na otázku: “vyplatí se projekt řešit?”
- Musí odpovědět na otázku: “je projekt uskutečnitelný?” (feasibility study)
- Musí proto vymezit hranici projektu
- Musí odpovědět na otázku: “kdo a co bude k řešení zapotřebí?”

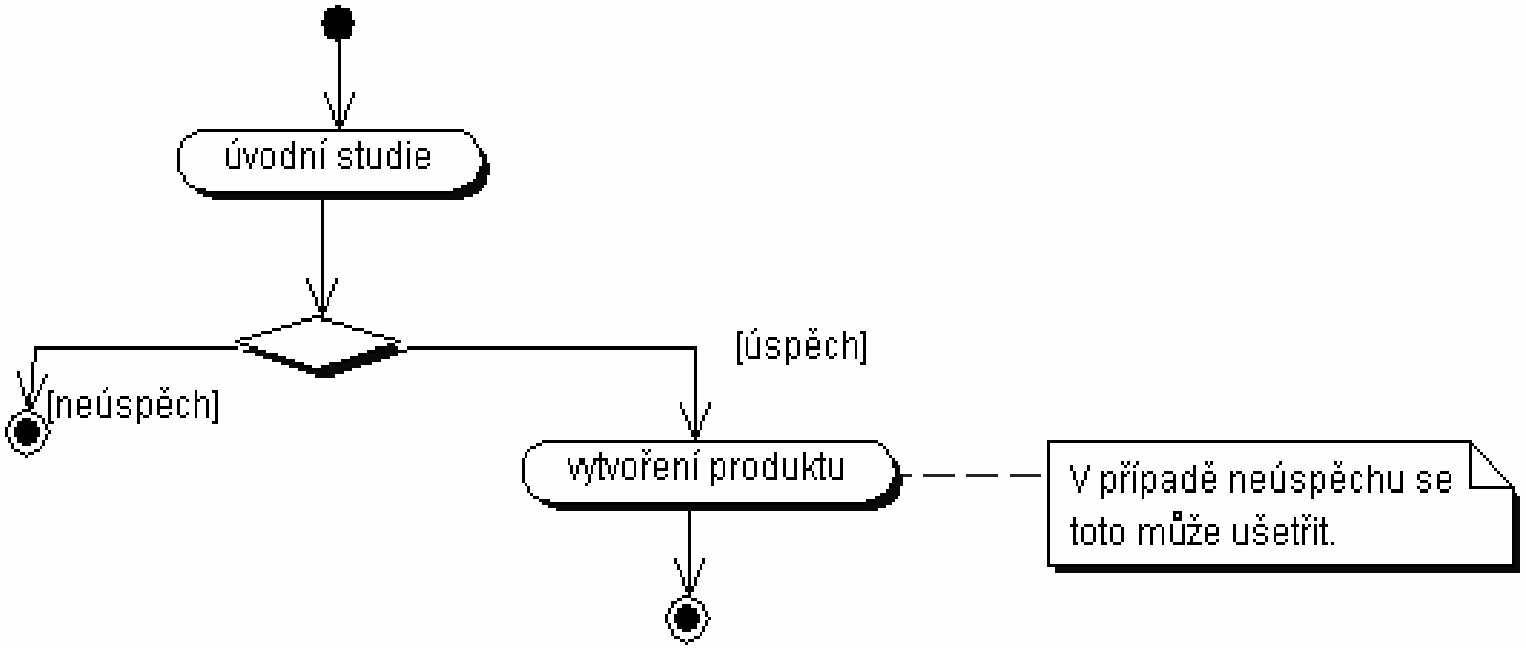
Vstupy úvodní studie

- Požadavky na systém
  - zadání projektu, deklarace záměru, vize projektu, odborný článek, tj. všechny dokumenty, které mají k řešenému problému nějaký vztah

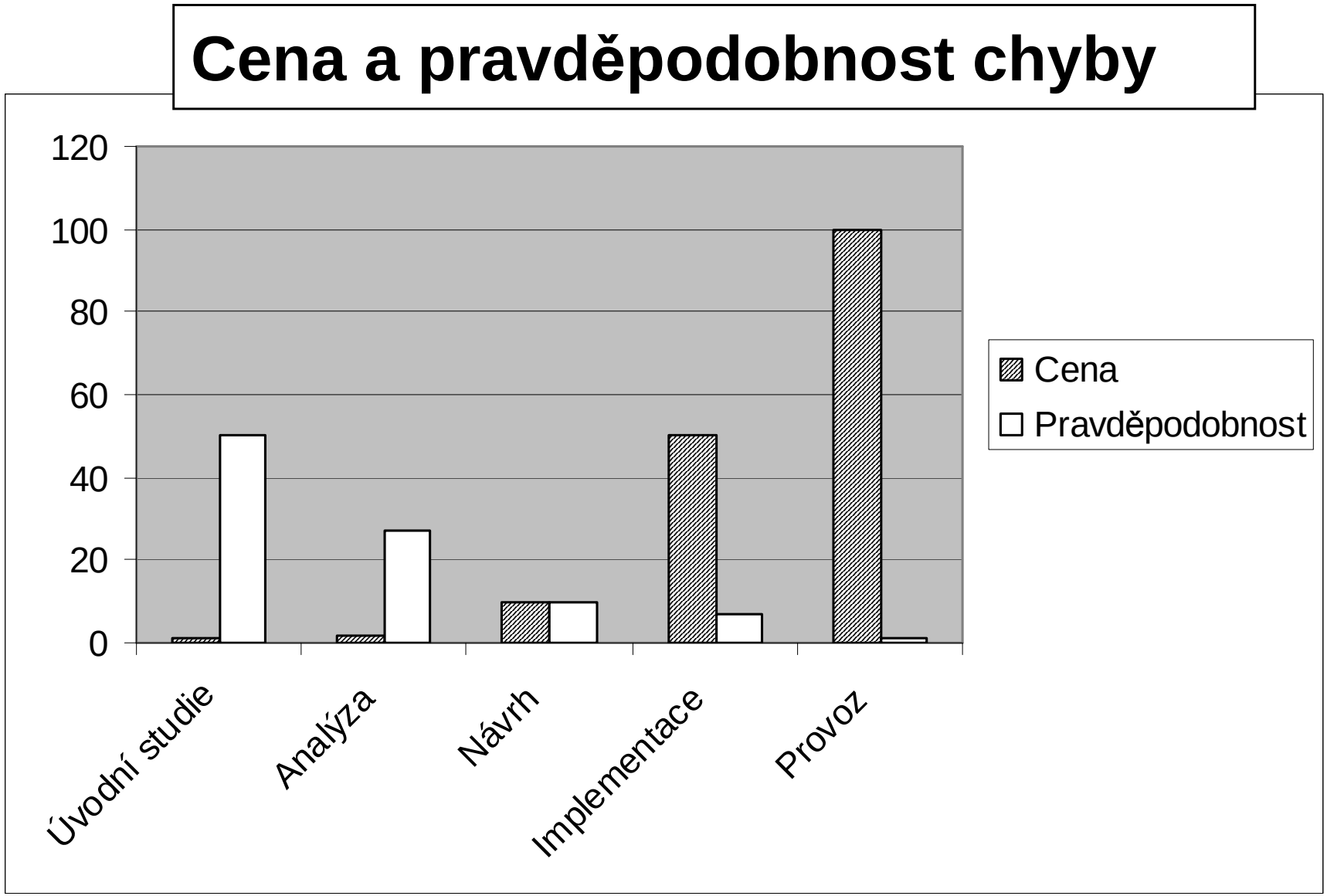
Výstupy úvodní studie

- Definice systému
  - katalog požadavků, definice hranice systému (diagram kontextu, model jednání), datový (pojmový) slovník, ...
- Projektová dokumentace
  - Řešitelský tým (funkce, zodpovědnosti).
  - Návrh řešení: HW, SW, komponenty.
  - Seznam úloh a harmonogram řešení.
  - Rozpočet: - cena HW, cena licencí na SW, cena vývoje SW a HW (COCOMO).

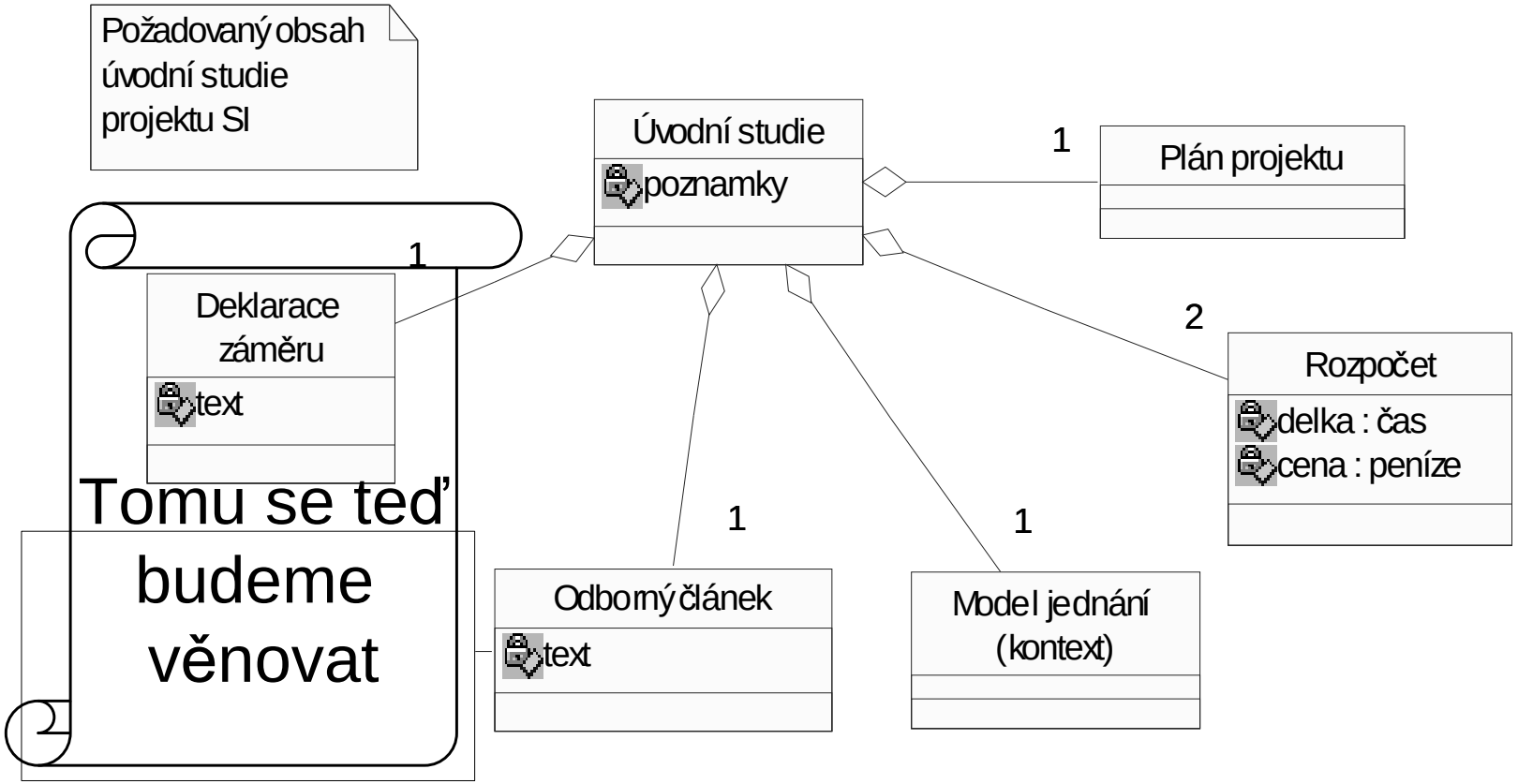
Úvodní studie může něco ušetřit



Má úvodní studie a analýza smysl?



Obsah úvodní studie



Deklarace záměru

- Krátký výstižný text se stručnými informacemi o projektu - jaké služby poskytuje, pro koho je určen a jaká předpokládá omezení.
- Měla by posloužit pro odpověď na otázku **“co ano, a co ne?”**.
- Je obvykle základem budoucího prospektu pro vytvořený produkt.

Deklarace záměru pro “Výtah”

(slouží pro odpověď na otázku “co ano, a co ne?”)

Systém “Výtah” slouží pro logické řízení obsluhy výtahu s jednou či více šachtami. Systém “Výtah” reaguje na požadavky uživatelů a dále registruje signalizaci ze spínačů v patrech a indikace ze senzorů přetížení. Systém “Výtah” ovládá klece výtahů pomocí povelů pro motory výtahů. Systém “Výtah” se nezabývá havarijním tlačítkem STOP, rovněž otevírání a zavírání dveří jde mimo systém (kvůli bezpečnosti).

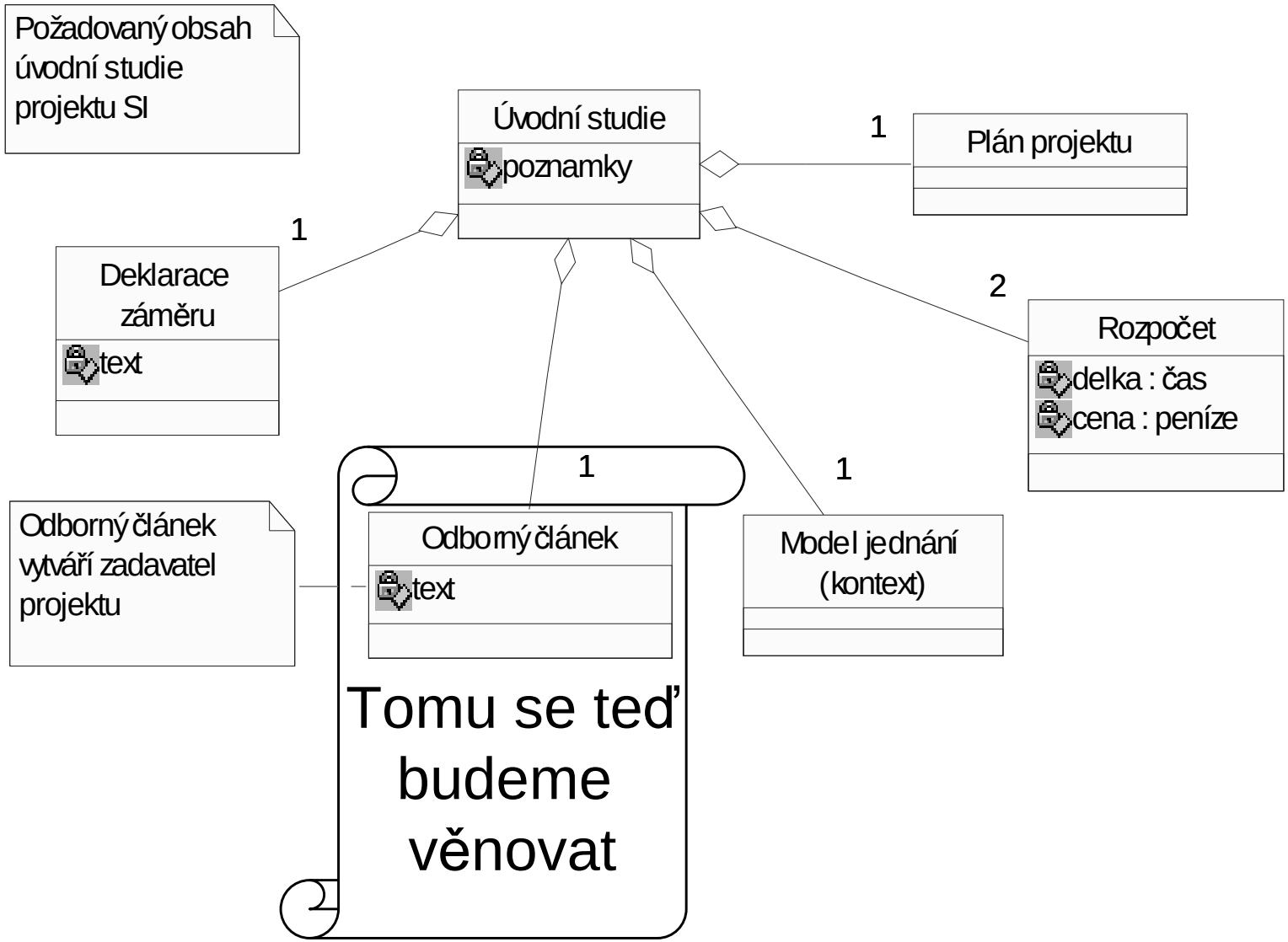
Příklad: Popis problematiky pro Systém pro přidělování úkolů (SPU)

- Ve firmě se přidělují úkoly
- Vykazuje se práce
- Je nutno vytvářet statistiky (kvůli rozhodování)
- Potřeba formalizace (kvůli automatizaci a spolehlivosti)
- Z toho plyne potřeba IS, který by to umožňoval
- Komerční systémy jsou ale drahé a složité

Deklarace záměru pro “SPU”

Systém SPU slouží pro zajištění podpory přidělování úkolů. Nadřízený pracovník přiděluje úkoly podřízeným pracovníkům a ti je plní. SPU poskytuje levné řešení tohoto problému, aby si je firmy s cca 100 zaměstnanci mohly dovolit. Jako komunikační médium využívá síť, což umožňuje kontrolu výkazů a zadávání úkolů i mimo firmu (např. ze služební cesty). Dále poskytuje různé statistiky činností.

Obsah úvodní studie



Odborný článek

- Všechny informace, které lze o projektu sehnat (články, interview, předpisy, …).
- Označení „**odborný článek**“ má vystihovat představu, že se jedná o texty v přirozeném jazyce, které sepsal odborník na řešenou problematiku. Informatik ji bude analyzovat a vytvoří popis přesnější. Někdy se nazývá **vize projektu**.
- Někdy se odborný článek nazývá „**katalog požadavků**“, ale my budeme takto označovat strukturovanou verzi odborného článku, kterou již tvoří informatik.

Odborný článek pro „Výtah“

(textový popis požadavků)

Systém “Výtah” slouží pro logické řízení obsluhy výtahu s jednou či více šachtami (předpokládají se 4 šachty a 40 úrovní). Systém zajišťuje efektivní plánování sběru a odvozu pasažérů mezi obsluhovanými patry podle požadavků (požadavek na přivolání výtahu pro jízdu směrem nahoru nebo dolů, požadavek na dopravení do určitého patra). Směr jízdy se nemění, dokud výtah nesplní objednávky v daném směru (výtah neví o pasažérech – neexistuje indikace prázdnosti klece). Přeplněný výtah nereaguje na výzvy (existuje indikace přetížení). Pro každou šachtu existuje samostatný motor ovládaný signály (povely UP, DOWN a STOP). Povel STOP způsobí zastavení výtahu v nejbližším patře v daném směru a otevření dveří výtahu (dveře se dají otevřít až v patře). Uvnitř klece je panel s tlačítky pater, indikace aktuální polohy a tlačítko STOP. Tlačítko STOP zabrání zavření dveří (jde mimo systém). Rovněž otevírání a zavírání dveří jde mimo systém (kvůli bezpečnosti). Příkazy pro systém jsou akceptovány až po zavření dveří. Operátor výtahu má k dispozici tlačítko ON/OFF, kterým zadává požadavek na zastavení pohybu výtahů.

Chyby v odborném článku

- Je příliš krátký a nepostihuje některé charakteristiky systému.
- Je příliš dlouhý a zabývá se problémy, které s popisem systému nesouvisí.
- Není z něj zřejmé, jaká data bude systém zpracovávat, jaké služby bude poskytovat, jak se budou vlastnosti systému měnit v čase či jako důsledek nějakých (popsaných) okolností.
- Neobsahuje některý požadavek.

Katalog požadavků

- Strukturovaná verze odborného článku.
- Odborný článek je předzpracován tak, aby tvořil strom požadavků.
- Požadavky jsou očíslovány a přes čísla se na ně lze odvolávat.

Katalog požadavků pro „Výtah“

(strukturovaný textový popis požadavků)

1. Systém “Výtah” slouží pro logické řízení obsluhy výtahu.

1.1 Výtah může mít jednu či více šachet (předpokládají se 4 šachty).

1.2 Výtah může mít dvě a více úrovní - pater (předpokládá se 40 úrovní).

2. Systém zajišťuje efektivní plánování sběru a odvozu pasažérů mezi obsluhovanými patry podle požadavků.

2.1 Požadavek na přivolání výtahu pro jízdu směrem nahoru nebo dolů (vzniká v patře).

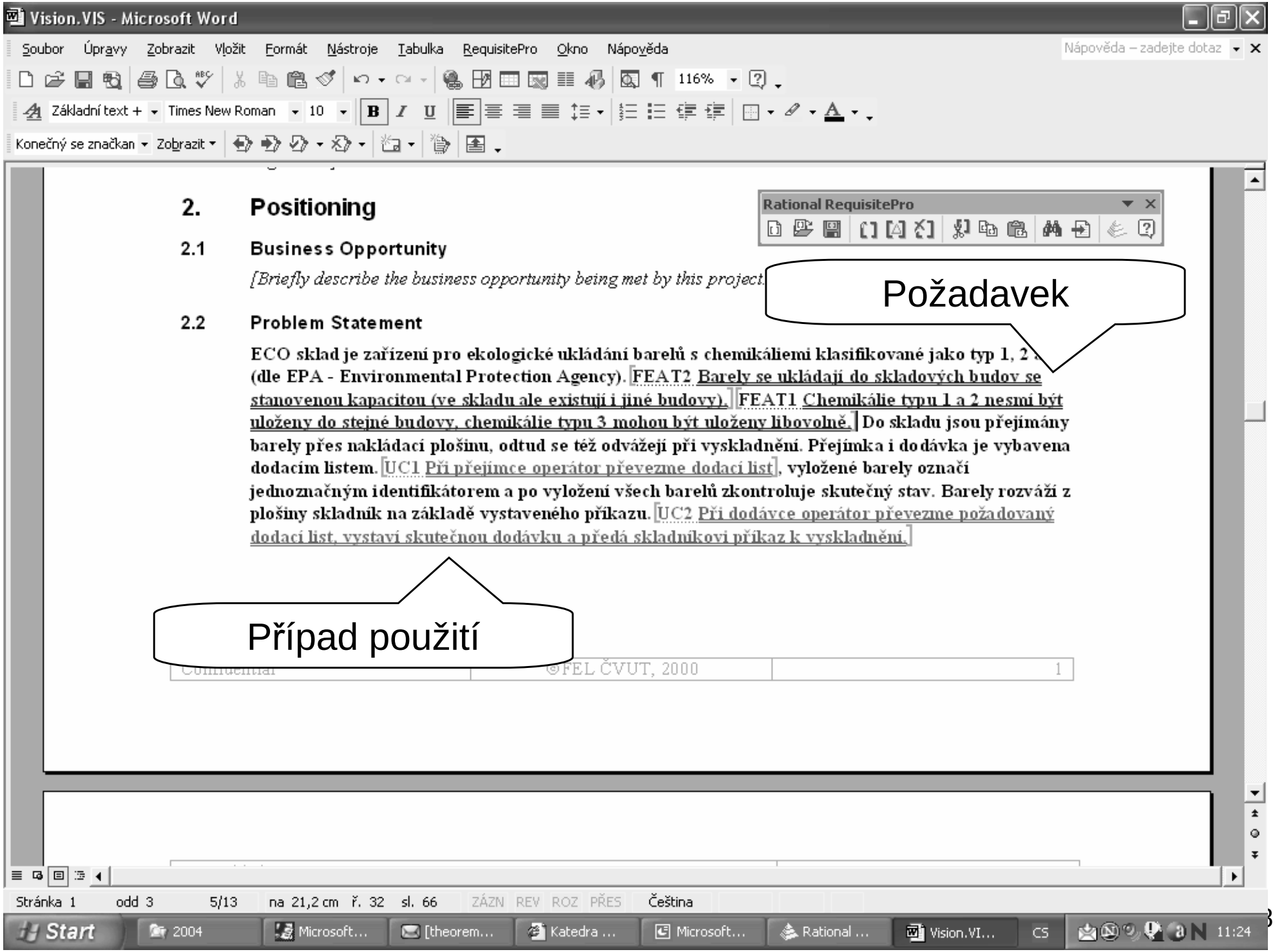
2.2 Požadavek na dopravení do určitého patra (vzniká v kleci výtahu).

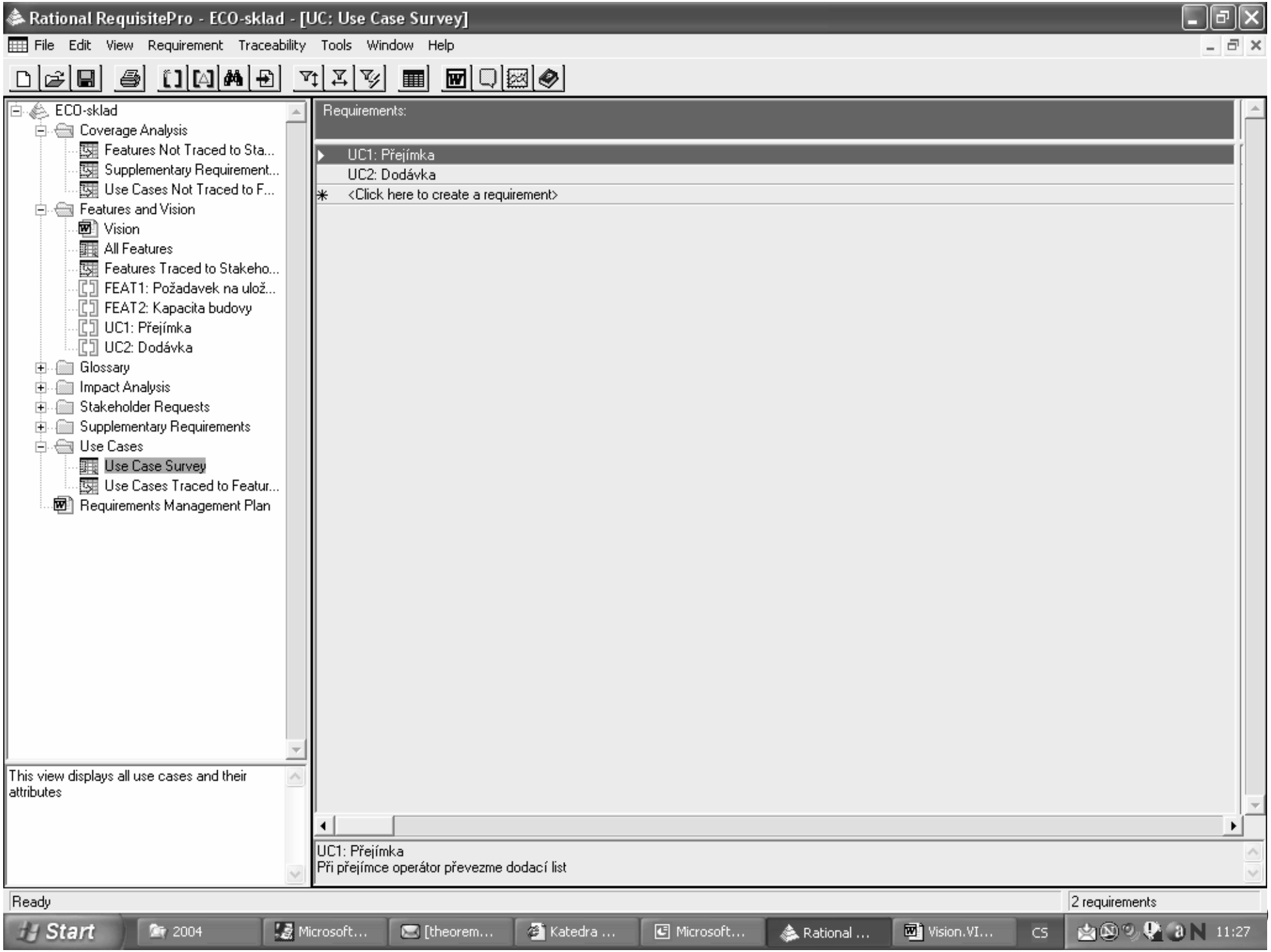
3. Směr jízdy se nemění, dokud výtah nesplní objednávky v daném směru (výtah neví o pasažérech – neexistuje indikace prázdnosti klece).

4. Přeplněný výtah nereaguje na výzvy (existuje indikace přetížení).

……

n. Pravděpodobnost chyby by měla být menší než 1 chyba za 10 let (příklad nefunkčního požadavku, který ale musíme též evidovat).


Vision.VIS - Microsoft Word
Soubor Úpravy Zobrazit Vložit Formát Nástroje Tabulka RequisitePro Okno Nápověda
Nápověda – zadejte dotaz
Základní text + Times New Roman 10
Konečný se značkan Zobrazit
Rational RequisitePro
2. Positioning
2.1 Business Opportunity
[Briefly describe the business opportunity being met by this project.
2.2 Problem Statement
ECO sklad je zařízení pro ekologické ukládání barelů s chemikáliemi klasifikované jako typ 1, 2 a 3 (dle EPA - Environmental Protection Agency). FEAT2 Barely se ukládají do skladových budov se stanovenou kapacitou (ve skladu ale existují i jiné budovy). FEAT1 Chemikálie typu 1 a 2 nesmí být uloženy do stejné budovy, chemikálie typu 3 mohou být uloženy libovolně. Do skladu jsou přejímány barely přes nakládací plošinu, odtud se též odvážejí při vyskladnění. Přejímka i dodávka je vybavena dodacím listem. UC1 Při přejímce operátor převezme dodací list, vyložené barely označí jednoznačným identifikátorem a po vyložení všech barelů zkontroluje skutečný stav. Barely rozváží z plošiny skladník na základě vystaveného příkazu. UC2 Při dodávce operátor převezme požadovaný dodací list, vystaví skutečnou dodávku a předá skladníkovi příkaz k vyskladnění.
Požadavek
Případ použití
Confidential
©FEL ČVUT, 2000
1
Stránka 1 odd 3 5/13 na 21,2 cm ř. 32 sl. 66 ZÁZN REV ROZ PŘES Čeština
Start
2004
Microsoft...
[theorem...
Katedra ...
Microsoft...
Rational ...
Vision.VI...
CS
11:24


Rational RequisitePro - ECO-sklad - [UC: Use Case Survey]
File Edit View Requirement Traceability Tools Window Help
ECO-sklad
Coverage Analysis
Features Not Traced to Sta...
Supplementary Requirement...
Use Cases Not Traced to F...
Features and Vision
Vision
All Features
Features Traced to Stakeho...
FEAT1: Požadavek na ulož...
FEAT2: Kapacita budovy
UC1: Přejímka
UC2: Dodávka
Glossary
Impact Analysis
Stakeholder Requests
Supplementary Requirements
Use Cases
Use Case Survey
Use Cases Traced to Featur...
Requirements Management Plan
This view displays all use cases and their attributes
Requirements:
UC1: Přejímka
UC2: Dodávka
<Click here to create a requirement>
UC1: Přejímka
Při přejímce operátor převezme dodací list
Ready
2 requirements
Start
2004
Microsoft...
[theorem...
Katedra ...
Microsoft...
Rational ...
Vision.VI...
CS
11:27

Význam termínů

- Všechny termíny v dokumentaci by měly být zaneseny ve **významovém slovníku** (technický termín je **datový slovník – Data Dictionary**).
- Je to proto, aby se termíny používané v dokumentaci interpretovaly stejně – např. „formulář 501“ může být termín běžný pro zadavatele, ale rozumět mu musí i řešitel - objednávka je obecně srozumitelný pojem, co ale má skutečně obsahovat?

Datový slovník (dle Yourdona)

Metaznak	Význam	Příklad	Jak se to čte
=	skládá se z	X =Y	X se skládá z Y
+	a	Z = X + Y	Z se skládá z X a Y
( )	může chybět	Z = X + (Y)	Z se skládá z X a příp. Z Y
{ }	opakování	Z = { X }	Z se skládá z několika X
[ ]	jeden z možných	Z = [X | Y]	Z se skládá buď z X nebo z Y (implicitní položku lze podtrhnout)
**	komentář	*toto je komentář*	
@	klíčová položka	Z = @X+Y	Z se skládá z X a Y, kde X je klíčová položka
@<číslo>	část složeného klíče	Z = @1X+@2Y	X a Y tvoří klíč (v tomto pořadí)

Datový slovník pro “Výtah”

(významy použitých termínů)

šachta = celé číslo *rozsah 1..4*
patro = celé číslo *rozsah 1..40*
tlačítko přivolání = patro + směr
směr = [UP | DOWN]
tlačítko patra = šachta + patro
stisk tlačítka = [tlačítko patra | tlačítko přivolání]
signalizace spínače patra = šachta + patro
signalizace přetížení = šachta
řídicí povel pro motor = šachta + povel
povel = [UP | DOWN | STOP]
indikace patra = šachta + patro
indikace přivolání = patro + směr
indikace = [indikace patra | indikace přivolání]

Př.: Rozhovor na téma „jméno“

- Člověk: My lidé se nazýváme jmény.
- Marťan: A co je to jméno?
- Člověk: Jméno je posloupnost znaků.
- Marťan: Takže „a1234“ je správné jméno?
- Člověk: Ve jménech používáme pouze písmena.
- Marťan: Takže „X“ je správné jméno?
- Člověk: Teoreticky ano, ale obvykle používáme jména, která obsahují nejméně dvě písmena. Navíc mají lidé většinou více jmen – jméno je rozděleno na části, kterým se říká „první jméno“, „příjmení“, apod.
- Marťan: …?

Datový slovník pro “Jméno”

celé jméno = { tituly před } + první jméno + { prostřední jméno } +
příjmení + { čárka + tituly za }

tituly před = [pan | paní | slečna | ing. | RNDr. | doc. | prof. | …]

první jméno = jméno

příjmení = jméno

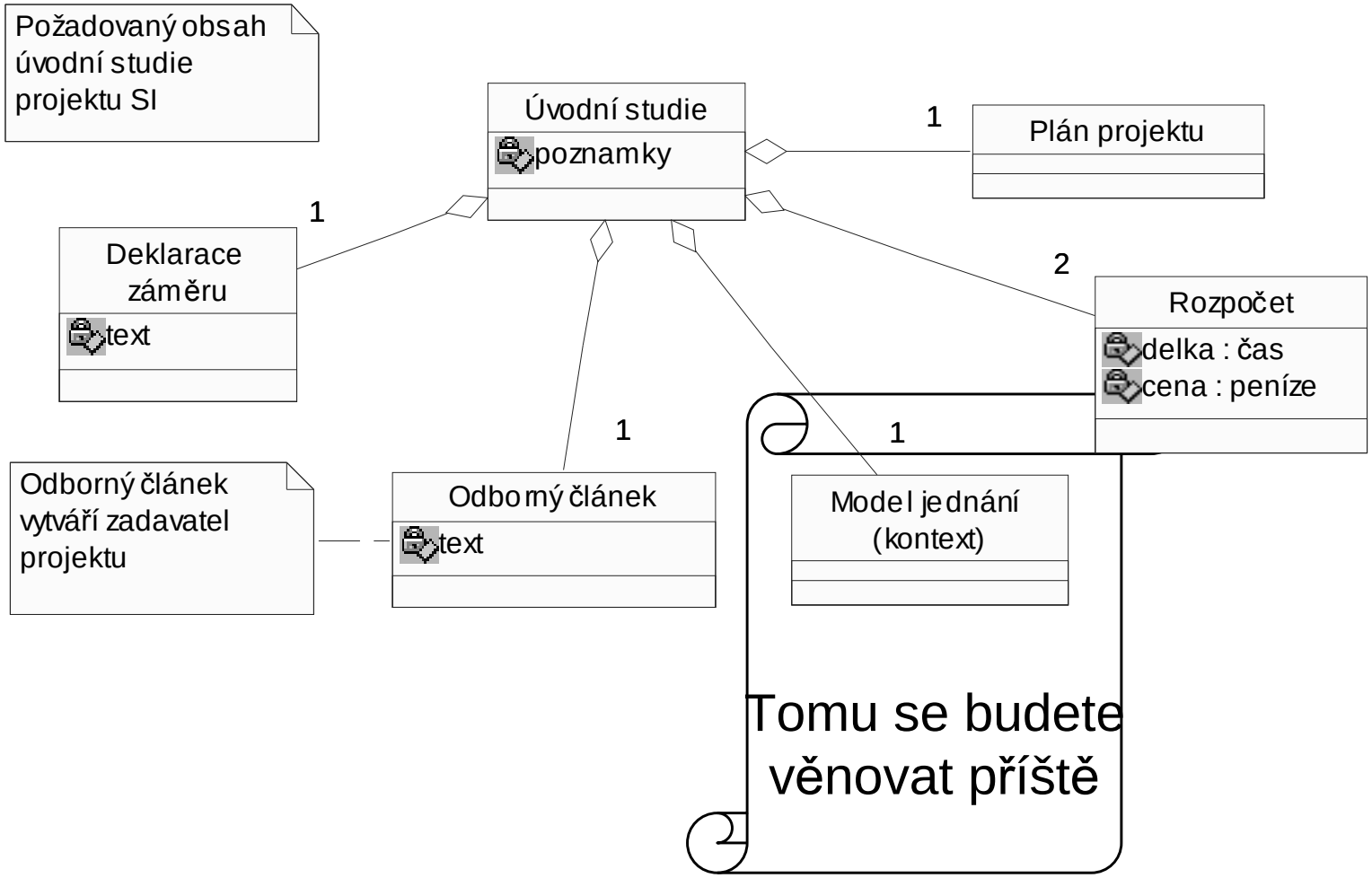
prostřední jméno = jméno

jméno = velké písmeno + 1{ malé písmeno }
písmeno = [malé písmeno | velké písmeno]
malé písmeno = [a | á | b | c | …] *písmena lokální abecedy*
velké písmeno = [A | Á | B | C | …] *písmena lokální abecedy*

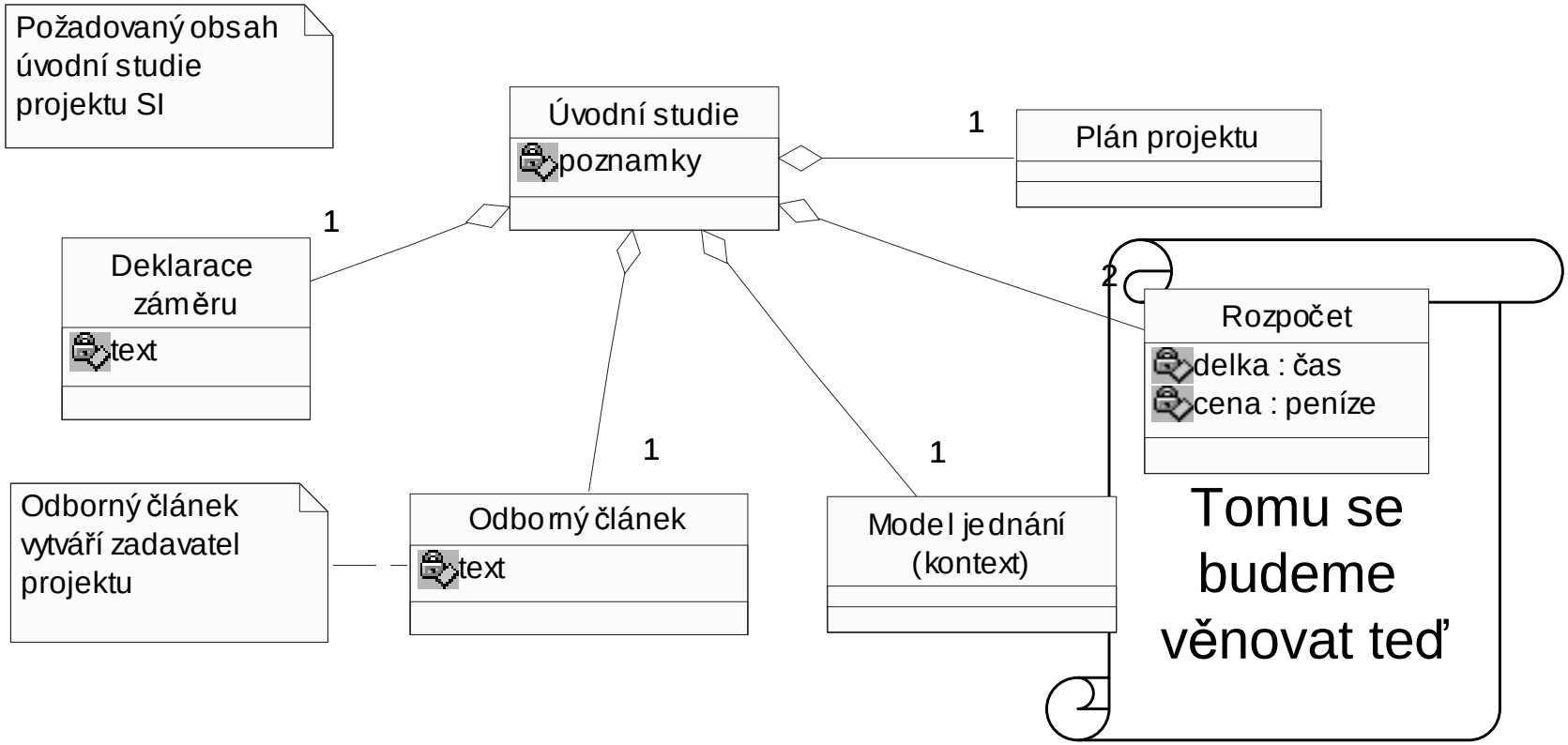
čárka = ,

tituly za = [CSc. | PhD. | DrSc. | prom.mat. | …]

Obsah úvodní studie



Obsah úvodní studie

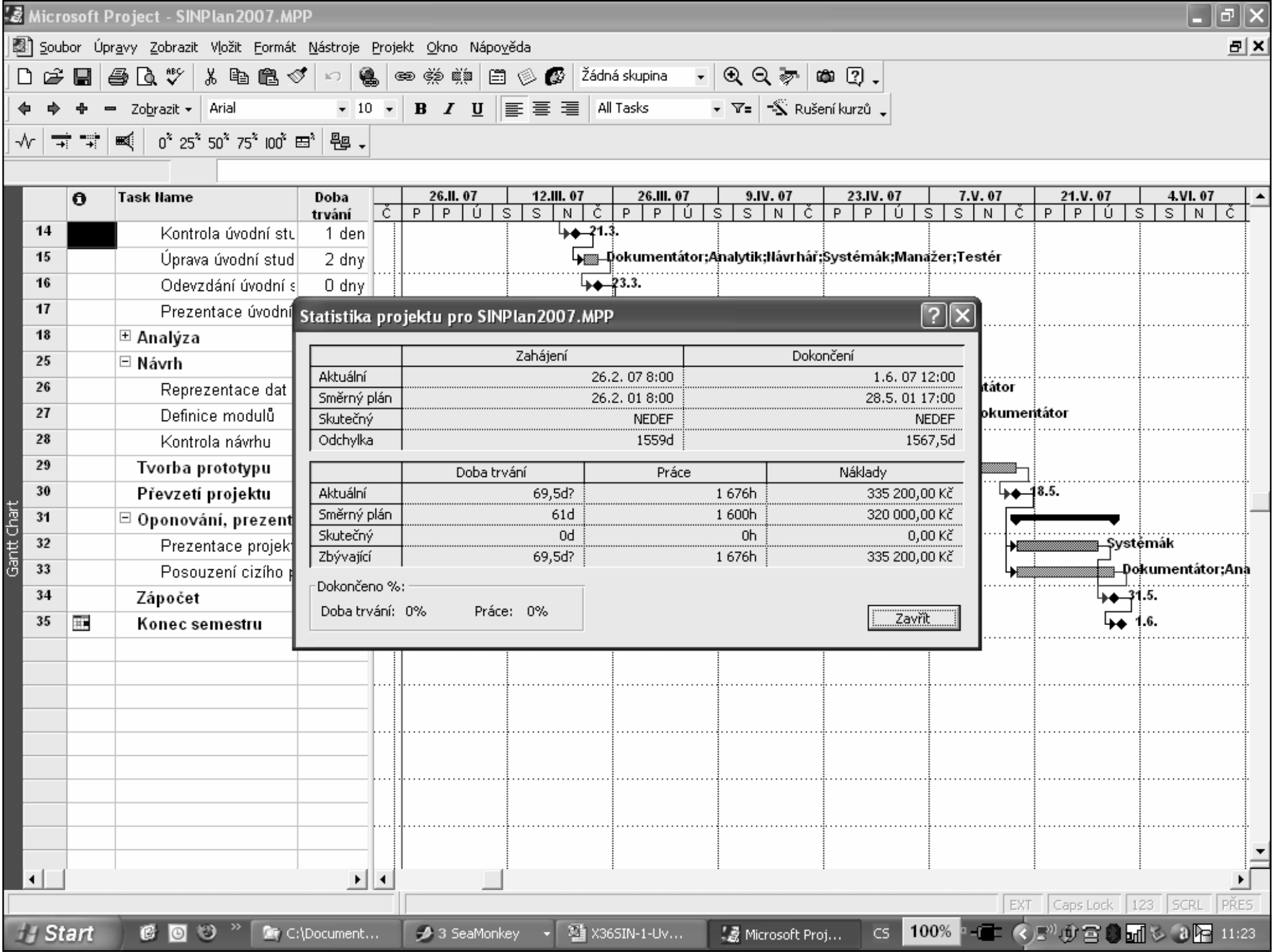


Odhad nákladů na projekt

- Odhad na základě zkušenosti z minula
  - již jsme něco podobného řešili a údaje o nákladech jsme si schovali
- Odhad na základě dekompozice problému na odhadnutelné složky
  - klasické řešení problému technikou „divide-et-impera“
- Odhady na základě výpočtu z odhadu rozsahu
  - odhad se obvykle řídí odhadem rozsahu kódu (LOC, KLOC, FP)

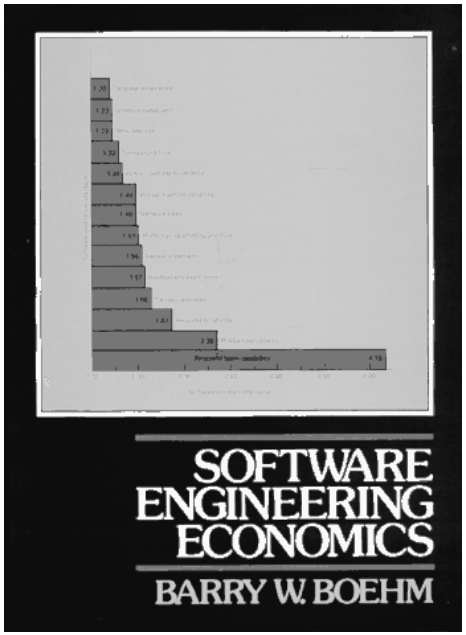
Náklady pomocí dekompozice na úlohy

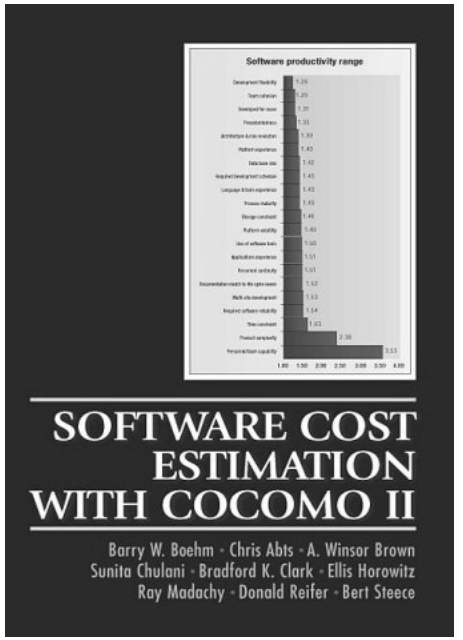
- Cena projektu = $\sum$ cen úloh
- Cena úlohy =
  - fixní náklady + cena za použití zdrojů
- Cena za použití zdroje =
  - odměna za práci v normální pracovní době +
  - odměna za práci přesčas +
  - fixní náklady na použití zdroje
- Práce = jednotky * délka
  - Práce je dána součinem počtu jednotek zdroje, které na úloze pracují a délky úlohy


Microsoft Project - SINPlan2007.MPP
Soubor Úpravy Zobrazit Vložit Formát Nástroje Projekt Okno Nápověda
Žádná skupina
Zobrazit Arial 10 All Tasks Rušení kurzů
Task Name
Doba trvání
Gantt Chart
14 Kontrola úvodní stu 1 den
15 Úprava úvodní stud 2 dny
16 Odevzdání úvodní s 0 dny
17 Prezentace úvodní
18 Analýza
25 Návrh
26 Reprezentace dat
27 Definice modulů
28 Kontrola návrhu
29 Tvorba prototypu
30 Převzetí projektu
31 Oponování, prezent
32 Prezentace projekt
33 Posouzení cizího p
34 Zápočet
35 Konec semestru
26.II. 07
12.III. 07
26.III. 07
9.IV. 07
23.IV. 07
7.V. 07
21.V. 07
4.VI. 07
21.3.
Dokumentátor;Analytik;Návrhář;Systémák;Manažer;Tester
23.3.
18.5.
Systémák
Dokumentátor;Ana
31.5.
1.6.
Statistika projektu pro SINPlan2007.MPP
Zahájení
Dokončení
Aktuální 26.2. 07 8:00 1.6. 07 12:00
Směrný plán 26.2. 01 8:00 28.5. 01 17:00
Skutečný NEDEF NEDEF
Odchylka 1559d 1567,5d
Doba trvání
Práce
Náklady
Aktuální 69,5d? 1 676h 335 200,00 Kč
Směrný plán 61d 1 600h 320 000,00 Kč
Skutečný 0d 0h 0,00 Kč
Zbývající 69,5d? 1 676h 335 200,00 Kč
Dokončeno %:
Doba trvání: 0%
Práce: 0%
Zavřít
EXT Caps Lock 123 SCRL PŘES
Start
C:\Document...
3 SeaMonkey
X36SIN-1-Uv...
Microsoft Proj...
CS
100%
11:23

Jiné metody odhadu

- COCOMO (Constructive Cost Model) - Barry Boehm





- http://sunset.usc.edu/research/COCOMOII/

Odhad rozpočtu dle COCOMO

- Vstup: Rozsah produktu v KLOC (KLines of Code)
- Náročnost = 2.94 * (Rozsah) $^{0.91}$
  - (udává se v člověko-měsících)
- Čas = 3.97 * (Náročnost) $^{0.28}$
- Cena = Čas * Plat
- Koeficienty se mění dle typu projektu a korekcí (cca 0.5 ÷ 2.0)

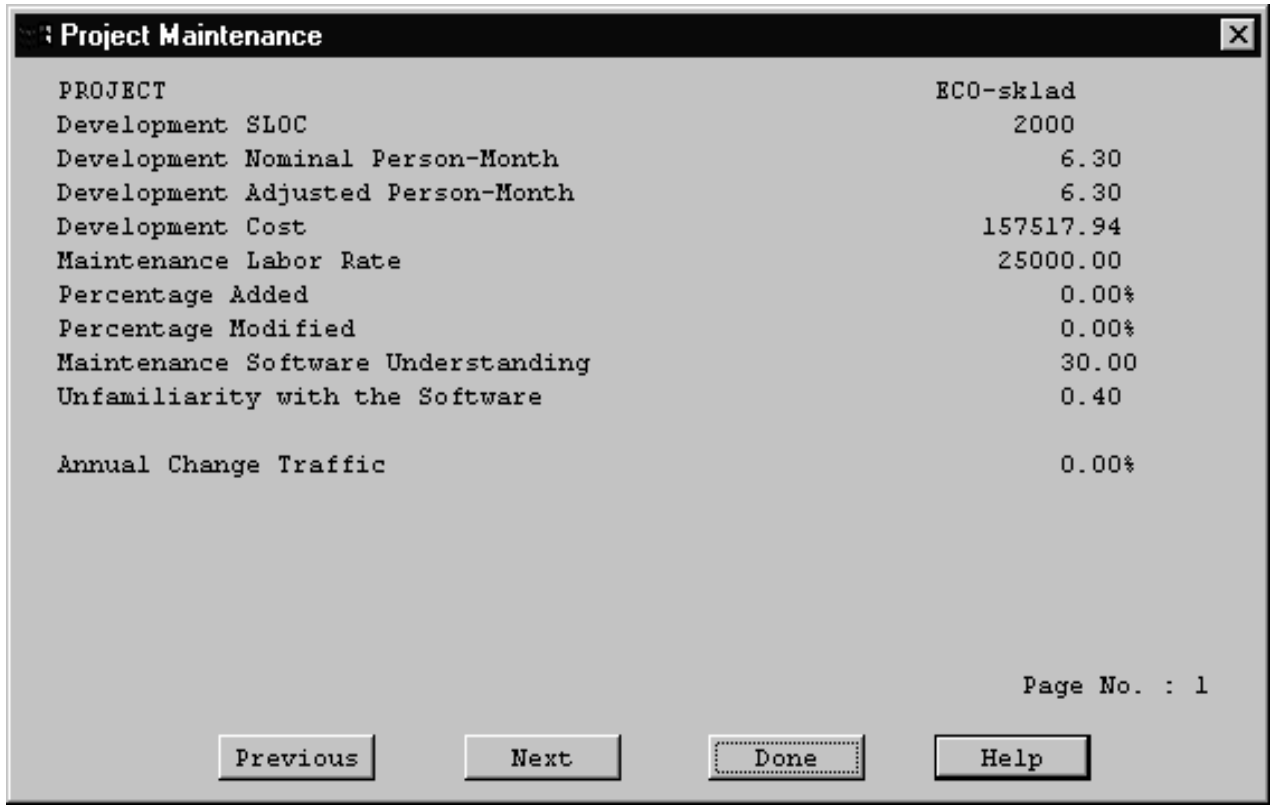
Příklad: Sestavovací program

- **Velikost: 32 KLOC (KDSI)**
- **Náročnost (Effort): 121.77 ČM**
- **Čas: 15.50 měsíců**
- **Lidí: 7.856**

(organic mode – jednodušší známé projekty, spočteno přes COCOMO kalkulátor)

http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/cocomo81.html

Příklad výpočtu (COCOMO II)



Př.: Náklady na projekt SPU

- Náklady dle MS-Project: 428.640,-
- Náklady dle COCOMO II: 443.000,-

  (člověkohodina je 200 Kč, parametry:

  SIZE = 5000, MODE = 1.05, DATA = 0.94, CPLX = 0.85, tj.

  náročnost = 13.86 člověko-měsíců,

  potřebný čas = 6.79 měsíce)

Zdroj: Projekt SPU

Výnosy pro projekt SPU

- Tento produkt by měl být distribuován jako krabicové řešení. Budeme-li předpokládat že se nám projekt podaří nasadit do 10ti firem (+ do jedné zdarma jako reklama), tak odhadovaná cena pro jednu kopii by měla být 50 000 Kč.
- Výnosy: 10 x 50.000,-

  tj. 450.000,-

Zdroj: Projekt SPU

Zhodnocení pro projekt SPU

- Obě metody odhadly cenu projektu na přibližně 450 000 Kč.
- Myslíme si, že cena produktu 50 000 Kč (+ DPH) by pro koncového zákazníka (firma s deseti až sto zákazníky) by mohla být přijatelná. Domníváme se, že nejméně 10 firem by si produkt zakoupilo, každý další prodej by znamenal zisk.
- Proto navrhujeme do projektu investovat.

Zdroj: Projekt SPU

Karnerova metoda odhadu

- **Jiná metoda odhadu nákladů, založená na modelu jednání**
- **Spočítejte aktéry, každého aktéra zařaďte do kategorie:**
  - **jednoduchý (např. jiný systém komunikující přes API) – váha 1,**
  - **střední (např. uživatel se znakovým terminálem nebo jiný systém komunikující přes TCP/IP) – váha 2, nebo**
  - **složitý (např. osoba komunikující přes GUI nebo Web) – váha 3.**
- **Sečtěte váhy všech aktérů a získáte neupravenou váhu aktérů - UAW (Unadjusted Actor Weights).**

Karnerova metoda odhadu (pokr.)

- **Rozdělte případy použití do kategorií podle odhadu počtu potřebných transakcí:**
  - **jednoduchý (méně než 4 transakce) – váha 5,**
  - **střední (4-7 transakcí) – váha 10, nebo**
  - **složitý (více než 7 transakcí) – váha 15.**
- **Sečtěte váhy všech případů užití a získáte neupravenou váhu případů užití - UUCW (Unadjusted Use Case Weight).**

Karnerova metoda odhadu (pokr.)

- **Sečtěte obě váhy a získáte neupravenou váhu modelu jednání – UUCP (Unadjusted Use Case Points)**
- **Adjustujte takto spočtenou váhu technickými faktory (TCF) a faktory prostředí (EF). Faktor má hodnotu 0 (žádný vliv) až 5 (silný vliv). Koeficient se spočítá:**

  TCF = 0.6+0.01*Technický faktor

  EF = 1.4-0.03*Faktor prostředí
- **Získáte tak upravenou váhu modelu jednání – UCP (Use Case Points)**

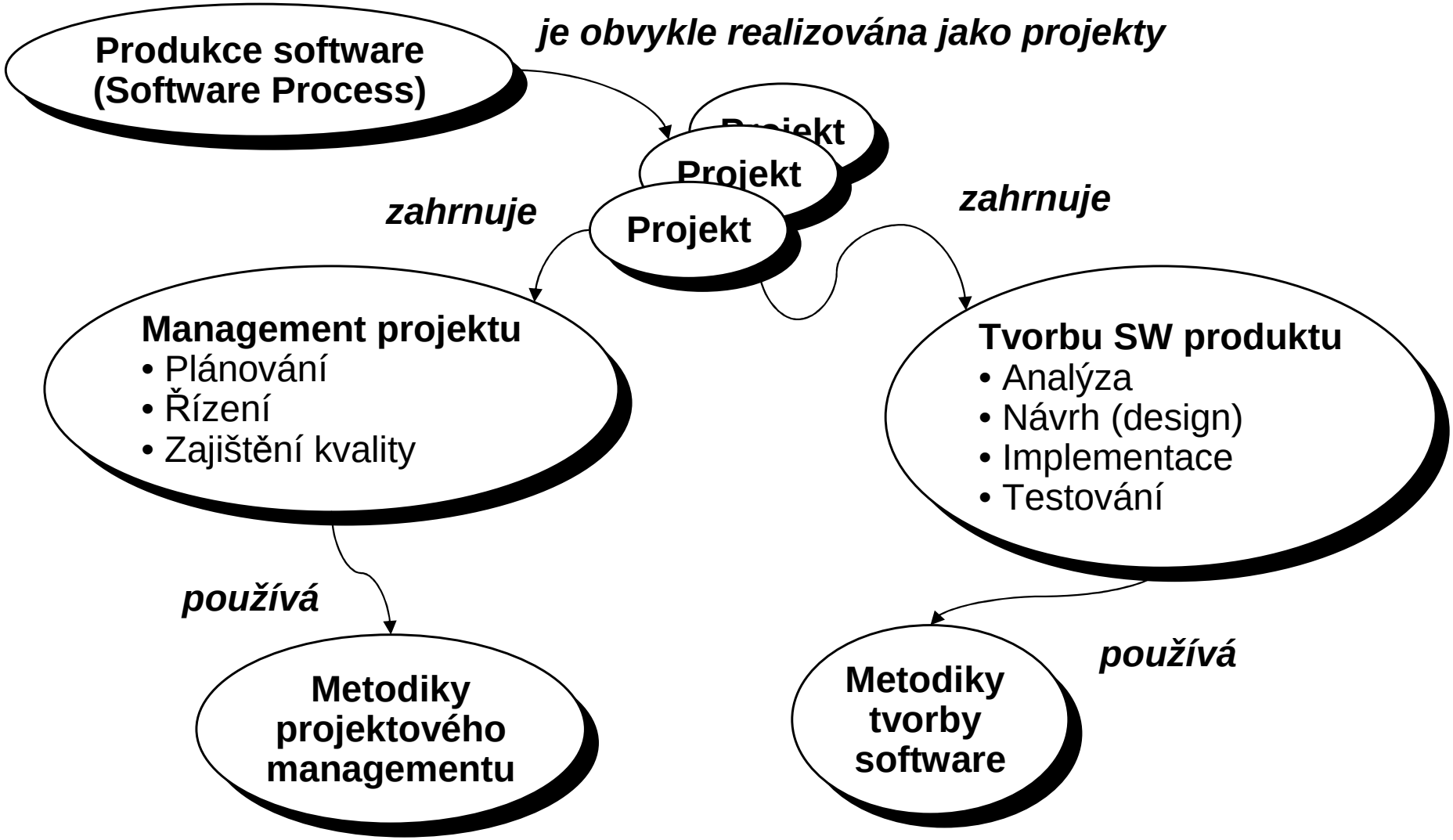
  UCP = (UAW+UUCP)*TCF*EF
- **Vynásobte UCP předpokládanou pracností jednoho případu užití (cca 15 – 30 hod, Karner doporučuje 20 hod). Získáte pracnost v člověko-hodinách.**

Příklad (Zdroj: www.komix.cz)

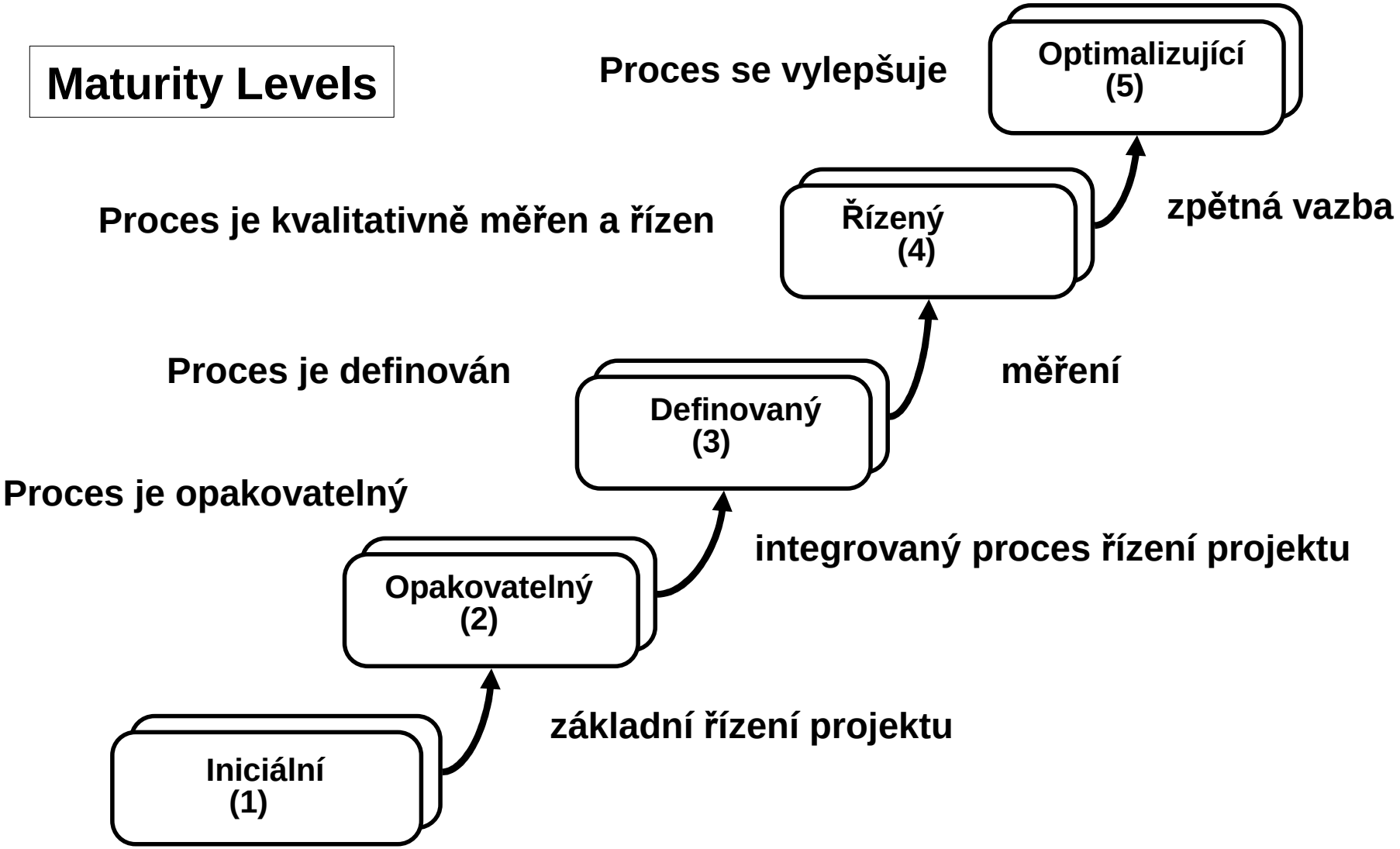
Co od Vás budeme chtít:

- Naučit se číst a vytvářet plány.
- Prostudovat si a upravit plán SINPlan2007.mpp (to je plán Vaší práce).
- Vytvořit plán práce pro Vaše následníky (pro ty, kteří budou projekt implementovat) a odhadnout z něj cenu.
- Pro ověření odhadnout cenu ještě jiným způsobem – např. pomocí COCOMO, nebo Karnerovou metodou.

Produkce software



Úroveň procesu tvorby software



The End

SOFTWAROVÉ INŽENÝRSTVÍ

ÚVODNÍ FÁZE VÝVOJE

Martin Komárek

MDD – Model Driven Development

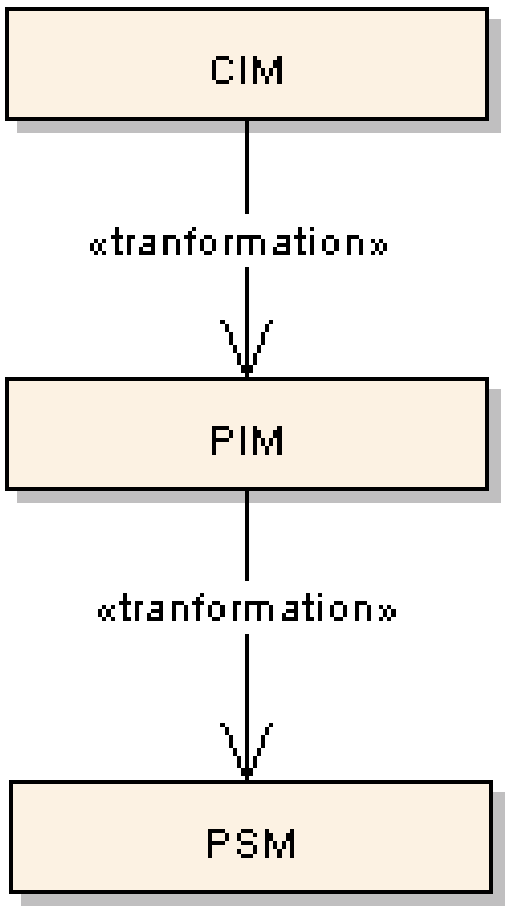
- Model vzniká vždy. Někdy bohužel jen v hlavách vývojářů.
- MDA Guide (www.omg.org)
- Model je kombinací textu a diagramů.
- Model by měl být „čitelný“ i bez diagramů.
- Diagramy „pouze“ usnadňují pochopení základní myšlenky popisovaného
- Každý digram by měl obsahovat pouze jednu myšlenku
- vytváření modelu je tedy převážně mentální úsilí vyúsťující v psaní textu

Proč modely vytvářet?

„Zaměstnanci přicházejí a odcházejí, modely zůstávají.“

- Znovupoužitelnost (firemní know-how)
- Sledovatelnost realizace požadavků
- Zvýšení udržovatelnosti produktu

MDD



- Computation Independent Model
- Platform Independent Model
- Platform Specific Model

CIM

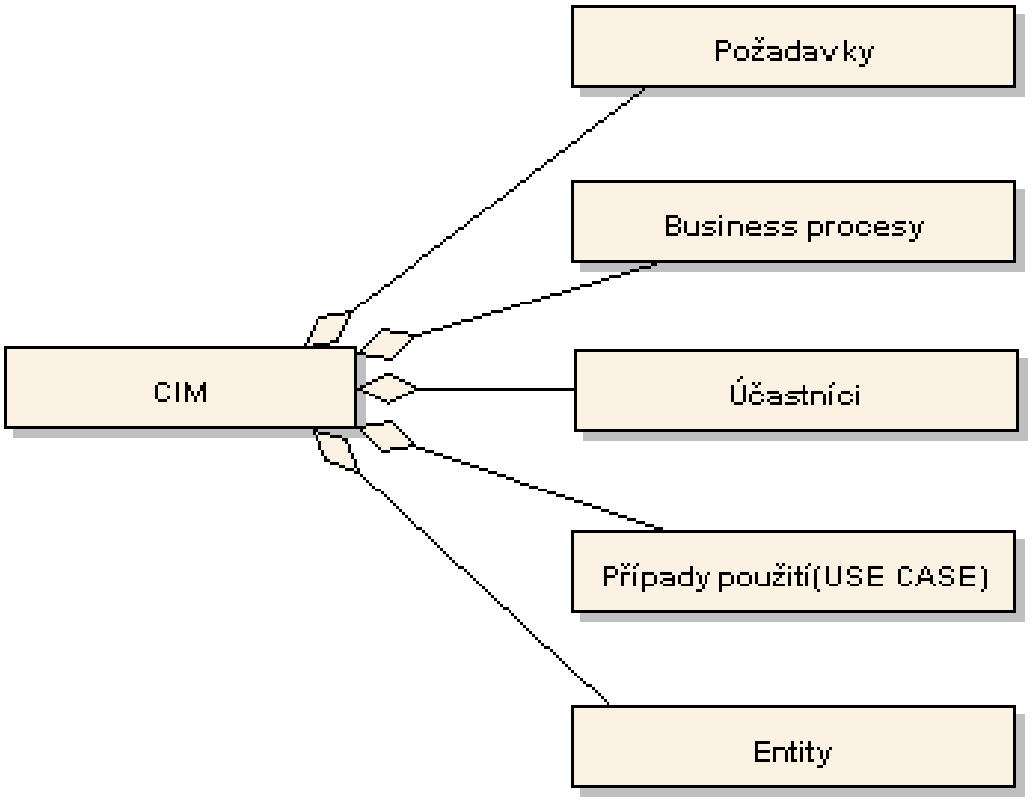
- Computation Independent Model ~ Domain Model ~ Conceptual Model
- strategický materiál pro odhad ceny
- strategický materiál pro vypracování nabídky
- podklad pro zavření smlouvy
- zákazníkovi srozumitelný a neobsahuje žádné implementační detaily

CIM

Podkladové materiály pro tvorbu jsou rozmanitého druhu:

- záznamy z jednání,
- normy, směrnice, zákony,
- stávající systémy,
- deklarace záměru,
- rešerše,
- .....

CIM


Požadavky
Business procesy
CIM
Účastníci
Případy použití(USE CASE)
Entity

UML – Unified Modeling Language

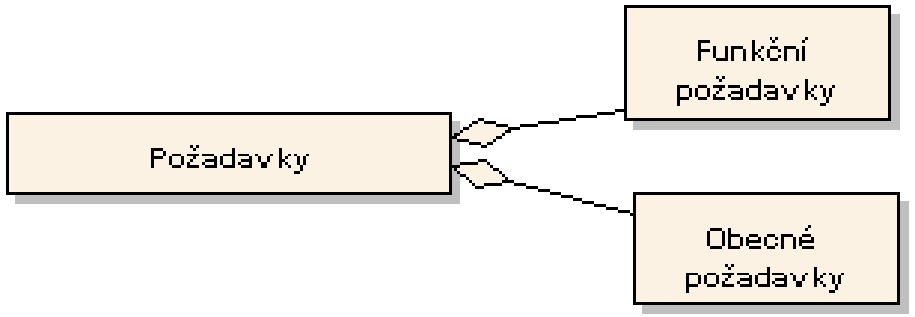
- Standard (www.omg.org)
- Syntaxe
- Sémantika

OOA/D - Objektově orientovaná analýza a návrh

- Co to je OO(A/D) ?
- Rozdíl mezi analýzou a návrhem
- Entity
- Business třídy
- Návrhové třídy

Požadavky

- Zachycují očekávání zákazníka
- mají jasné identifikátory (číslují se)
- hierarchicky děleny (zde na "funkční" a "obecné", ale možno i jinak)



- jeden požadavek=pouze jedno měřitelné "očekávání„
- obsahují (pokud možno) minimální množství informací o implementaci

Funkční požadavky

- definují co bude systém umožňovat
- měly by mít stanoveny priority

Př.:

1.1 Systém bude evidovat(minimálně po dobu pěti let) kdo a kdy měnil emailovou adresu na kterou je uživatelům zasíláno heslo.

1.2. Systém bude vždy při změně emailové adresy, na kterou je uživatelům zasíláno heslo, o této změně uživatele informovat a to tak, že mu na původní i novou emailovou adresu pošle zprávu o tom, kdo a kdy změnil jeho adresu na zasílání hesla.

Obecné požadavky

- vztahují se k celému systému
- spíše omezují způsob, jak bude systém navržen
- většinou se realizují vhodnou volbou jádra systému

Typové příklady obecných požadavků

- požadavky na parametry RAMS (relaibility, availability, maintainability, safety/security)
- požadavky na výkon
- požadavky na použitou platformu (operační systém, hardware, ...)
- požadavky na rozhraní s uživately i jinými systémy (vícejazyčnost, WEB/WAP/SMS brány/... , ...)
- požadavky na dokumentaci (analytická / návrhová / programátorská / uživatelská)
- požadavky související s právními aspekty (výhradní / nevýhradní / otevřená licence, otevřený kód, standardy atd.)

Příklad obecného požadavku

6.5. Systém bude spolehlivý.

Takovýto požadavek může být v závěru projektu oběma stranami interpretován velmi rozdílně. Proto je nutné požadavek upřesnit třeba takto:

6.5. Systém bude spolehlivý.

6.5.1. Střední doba do výpadku systému bude maximálně 20 dní.

6.5.2. Střední doba do opravy systému bude maximálně 12 hodin.

Business procesy - Co to je ?

- Zachycují i "akce", které se souvisejí s nasazením systému jen nepřímo

Business procesy – proč popisovat?

- Pro potřeby tvorby nabídky

  Model (business) procesů slouží jako podklad pro odsouhlasení rozsahu aplikace klientem.

- Pro potřeby nasazení systému

  a) pro testery - tester může testovat i business logiku

  b) pro tutory - školitelé musí vědět nejen co dělá které menu, ale i k čemu systém slouží

Business procesy – příklad

Př. "Přichází zákazník k přepážce banky" -> "Přijetí žádosti o hypotéku" -> atd... .

I bez znalosti bankovní problematiky dovodíme, že žádost bude pravděpodobně zadána do systému bankovní úřednicí.

Pokud bychom ale neměli v modelu zachyceno, že zákazník "Přichází k přepážce" mohli bychom si myslet, že zákazník může podat žádost o hypotéku i přes webové rozhraní nebo po telefonu.

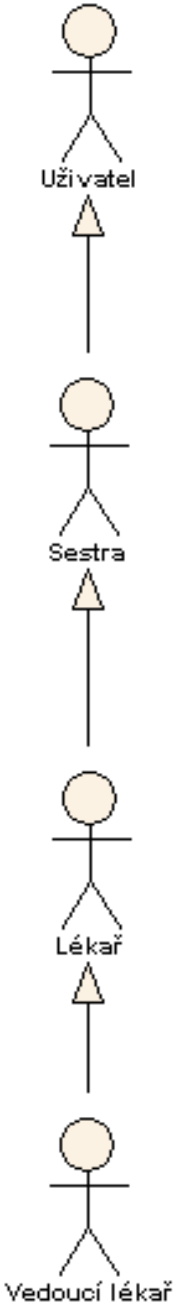
I z tohoto triviálního příkladu je zřejmá důležitost zachycení business procesů v CIM modelu.

Účastníci (=aktéři)

Jednotlivý účastník(actor)

- jsou vůči systému externími entitou, která systém využívá nebo ho ovlivňuje.
- většinou účastníkem reálná osoba(uživatel), ale může jím být například i "čas" (spouštění záloh atd..) nebo dokonce "blesk(=elektromagnetické rušení)"

Generalizace účastníků

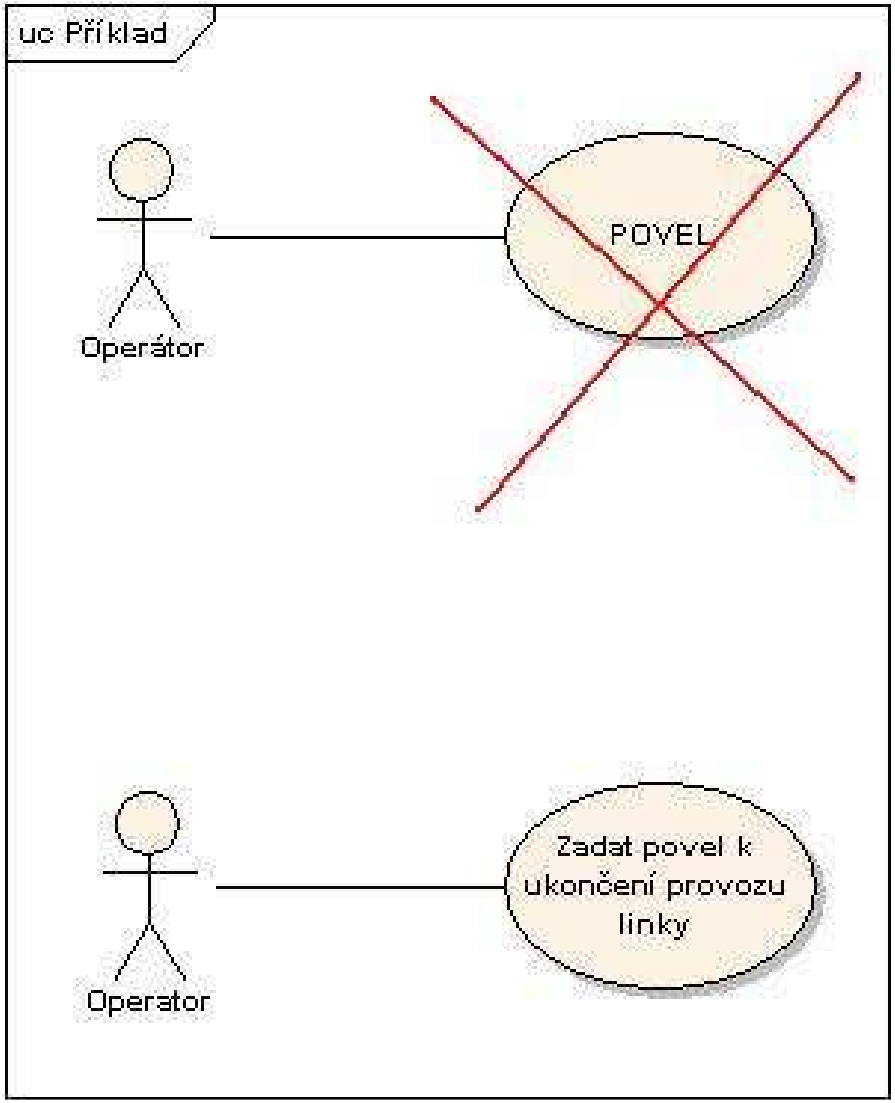


Případy použití systému (USE CASE)

“případ použití"~"případ užití"~"užitná činnost“~”USE CASE”

- Měl by popisovat jednu rutinní akci jednoho účastníka v jednu chvíli
- Je vždy iniciován účastníkem
- USE CASE diagram znázorňuje funkce systému z pohledu účastníků
- Název USE CASE má vždy *slovesnou vazbu*!!!

Název USE CASE !!! Slovesná vazba !!!



Vztahy mezi případy použití

- <<extend>>

  - pokud nějaký případ rozšiřuje chování (je zde možnost volby)

- <<include>>

  - pokud jeden případ zahrnuje případ jiný

  (např. přepočet ceny prodávaného zboží po změně kurzu EURO)

Entity systému

- většinou reálně existující „věci“, se kterými systém bude manipulovat
- seznam entit a jejich popis je „slovníčkem pojmů“
- ze seznamu "entit" se vychází při vytváření "analytických tříd" v PIM

Package (=balíček)

- Umožňují strukturovat modely
- Obdoba adresářů

Dotazy ?

X36SIN: Softwarové inženýrství

Modelování požadavků

Kontext

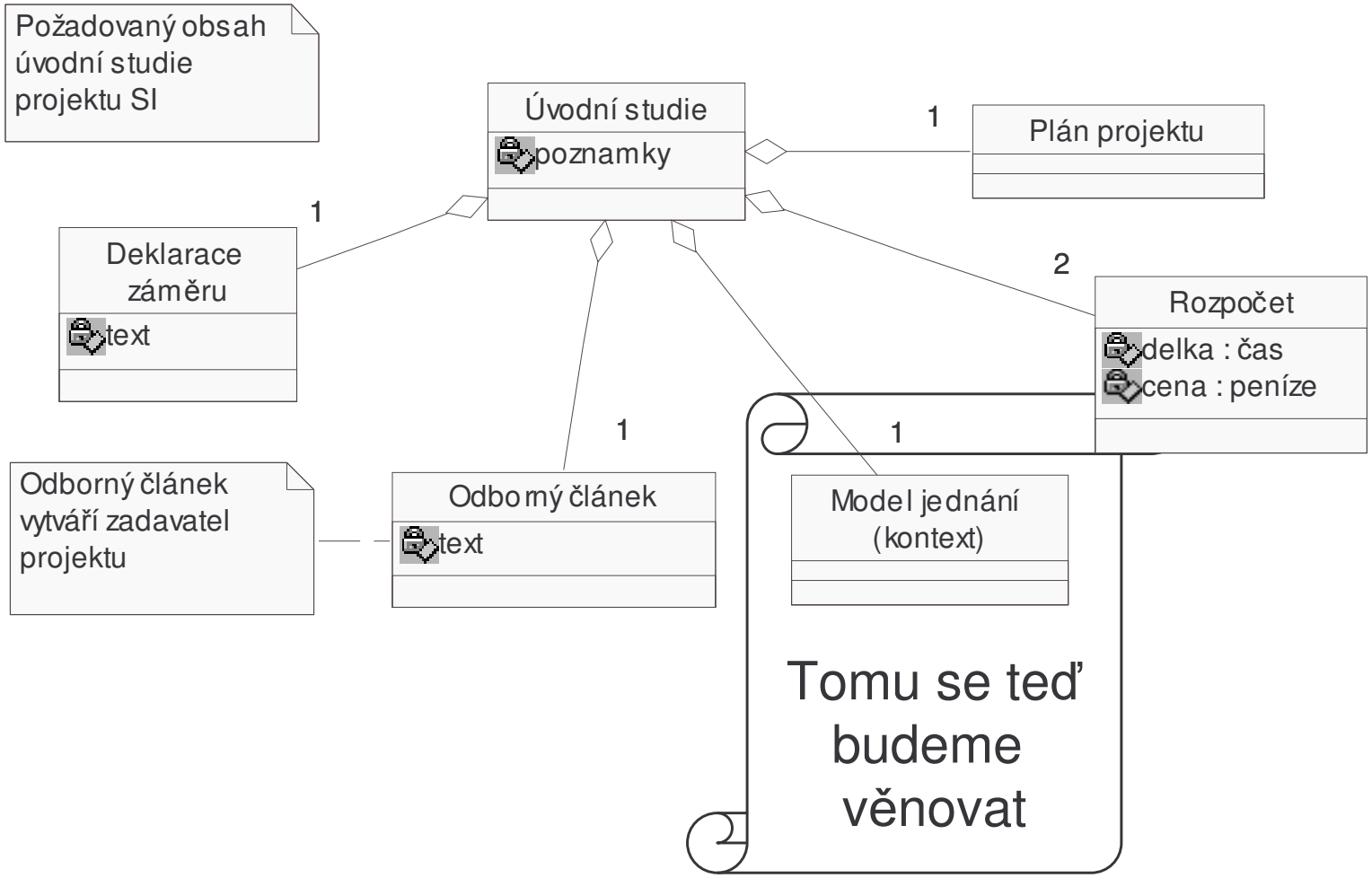
Dosud jsme probrali:

- Modely životního cyklu
- Co to je deklarace záměru, odborný článek, katalog požadavků
- Jak se plánuje a odhadují náklady

Dnes se budeme zabývat:

- Sběrem a modelováním požadavků
- Seznam aktérů a seznam událostí vyjadřujeme pomocí modelu jednání (use case model)
- Slovní popis případů užití a grafické vyjádření

Obsah úvodní studie

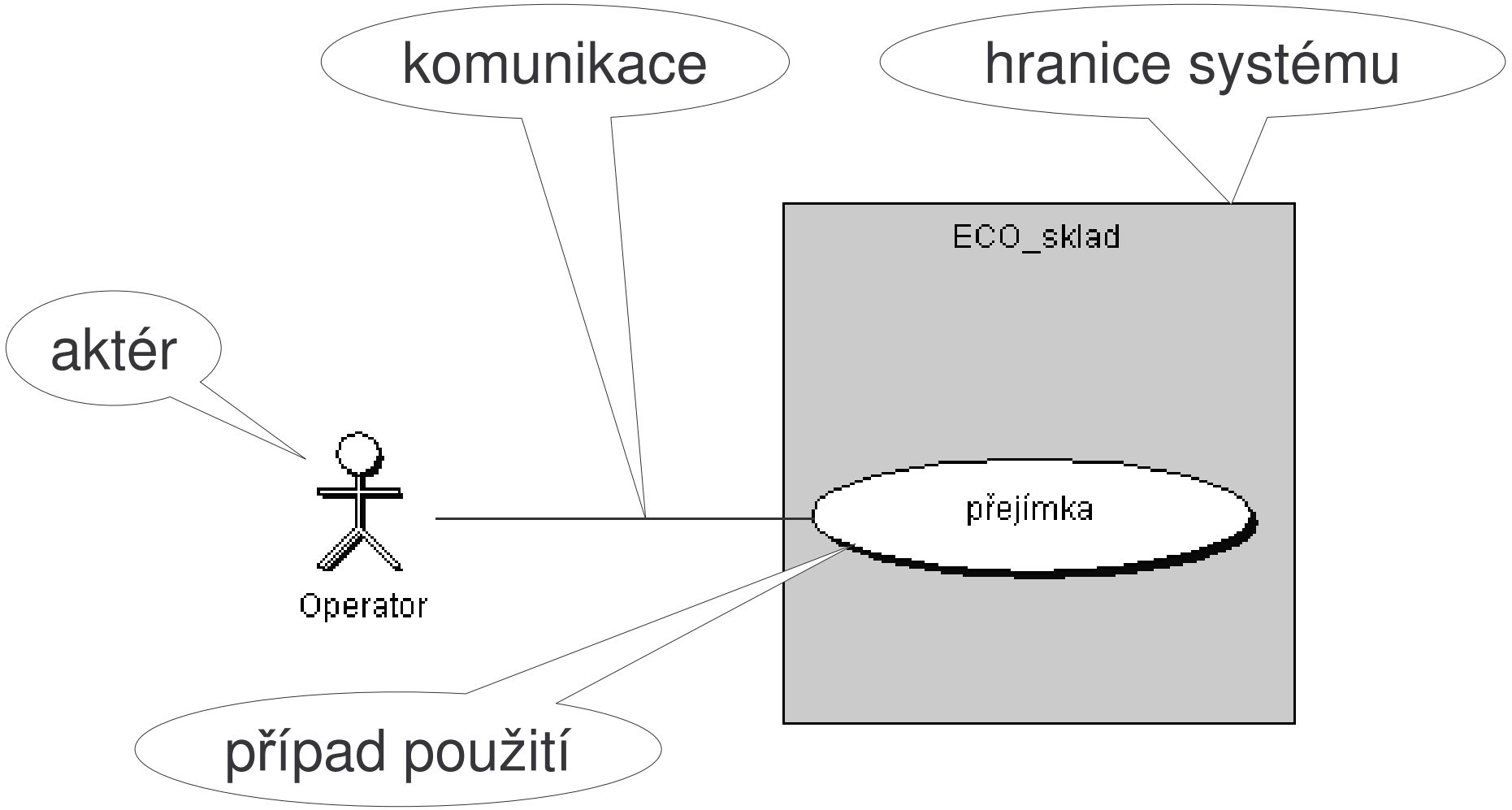


Model jednání (Use Case Model)

Prvky:

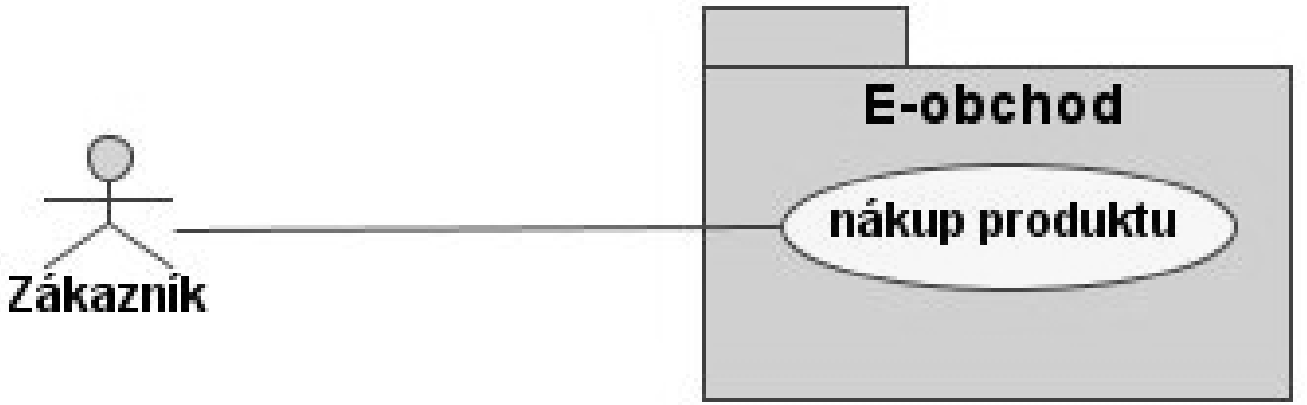
- **aktér** (actor) - uživatelská role nebo spolupracující systém
- **hranice systému** (systém boundary) - vymezení hranice systému
- **případ použití** (use case) - dokumentace události, na kterou musí systém reagovat
- **komunikace** - vazba mezi aktérem a případem použití (aktér komunikuje se systémem na daném případu)

Notace modelu jednání (UML)

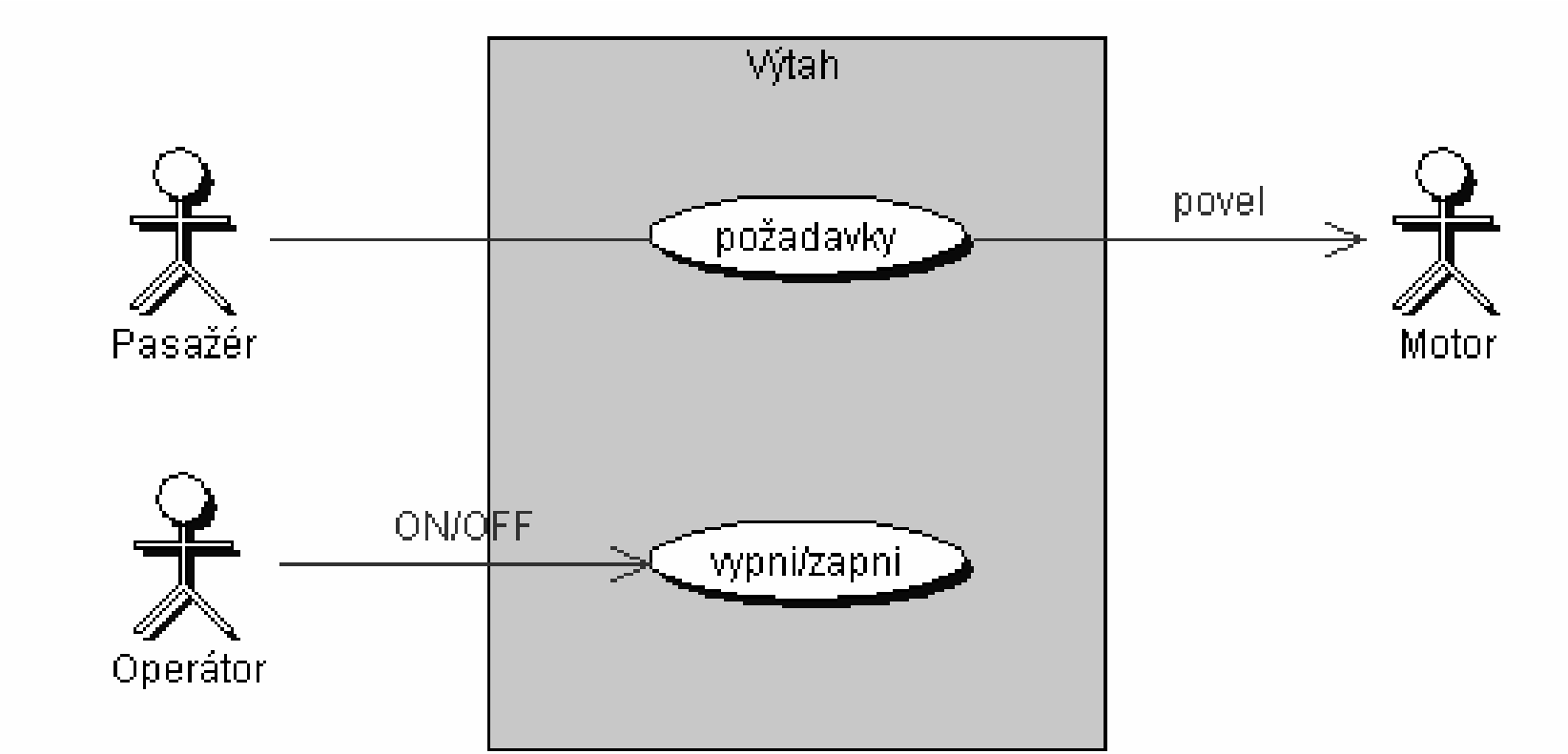


Příklad: „e-obchod“

- E-obchod poskytuje zákazníkům možnost nákupu produktů.



Příklad modelu jednání



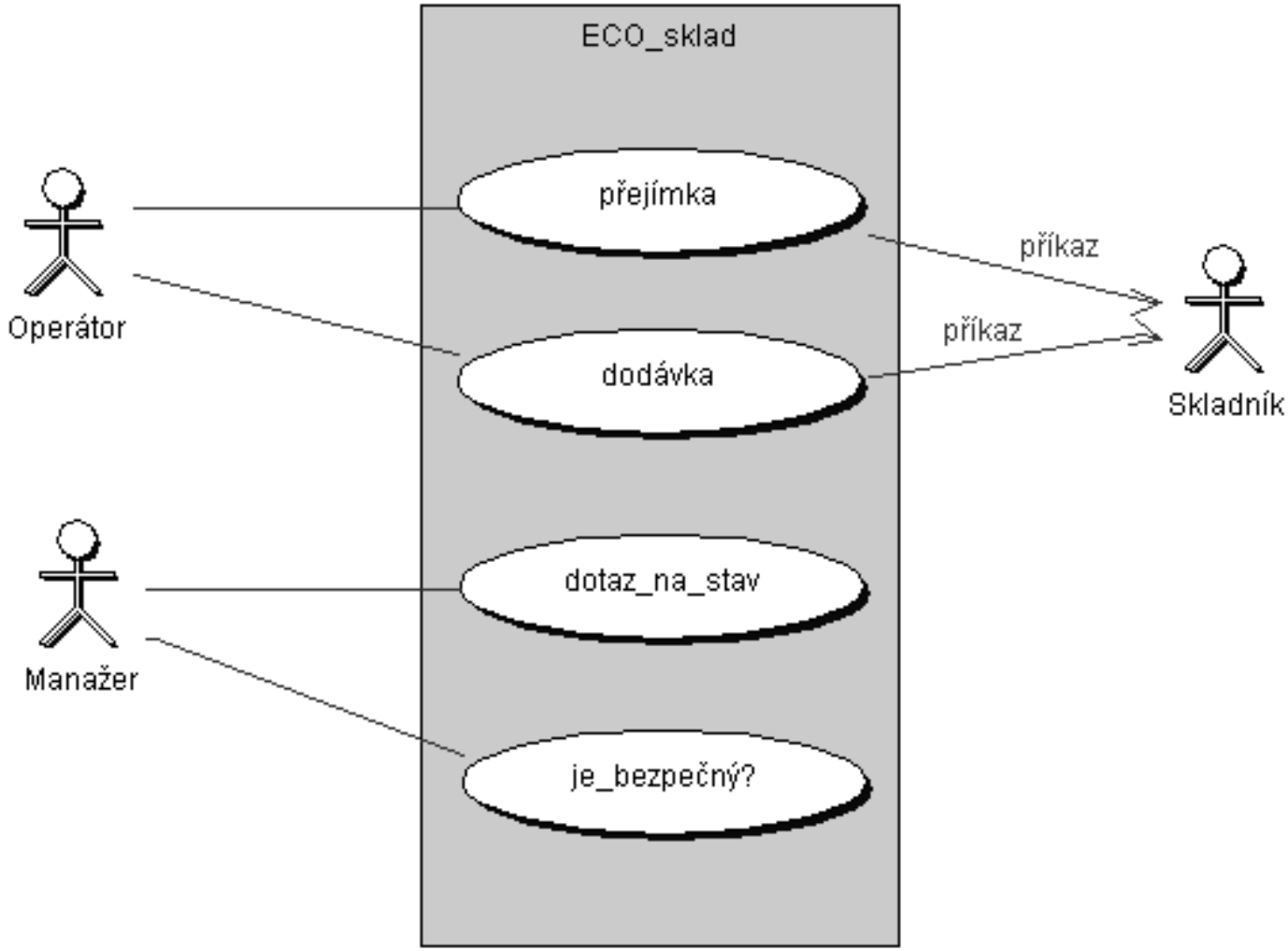
Chyby v modelu jednání

- Aktéři spolu komunikují mimo systém
- Není zdůrazněn dvojí výskyt aktéra
- Chybí případ použití (služba) pro některou událost
- Chybí některá reakce systému
- Případ použití není popsán v datovém slovníku
- Případ použití je popsán nevhodně (příliš obecně)
- Dva různí aktéři mají stejnou sadu událostí (pak to zřejmě nejsou různí aktéři)
- Za událost se považuje přihlášení do systému (zařazení do role jde mimo kontext)

Příklad: ECO-sklad

ECO sklad je zařízení pro ekologické ukládání barelů s chemikáliemi klasifikované jako typ 1, 2 a 3 (dle EPA - Environmental Protection Agency). Barely se ukládají do skladových budov se stanovenou kapacitou (ve skladu ale existují i jiné budovy). Chemikálie typu 1 a 2 nesmí být uloženy do stejné budovy, chemikálie typu 3 mohou být uloženy libovolně. Do skladu jsou přejímány barely přes nakládací plošinu, odtud se též odvážejí při vyskladnění. Přejímka i dodávka je vybavena dodacím listem. Při přejímce operátor převezme dodací list, vyložené barely označí jednoznačným identifikátorem a po vyložení všech barelů zkontroluje skutečný stav. Barely rozváží z plošiny skladník na základě vystaveného příkazu. Při dodávce operátor převezme požadovaný dodací list, vystaví skutečnou dodávku a předá skladníkovi příkaz k vyskladnění.

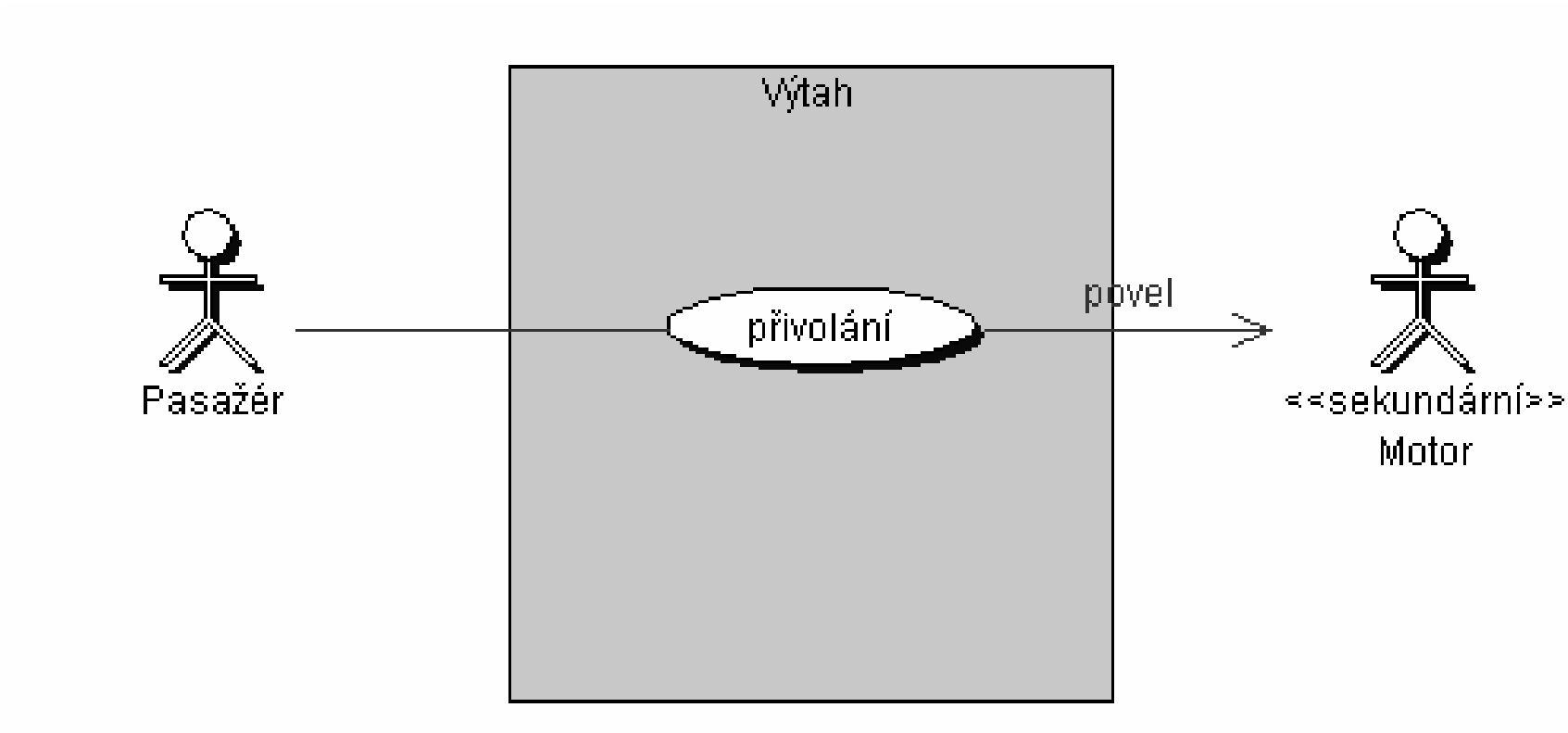
Model jednání pro ECO-sklad



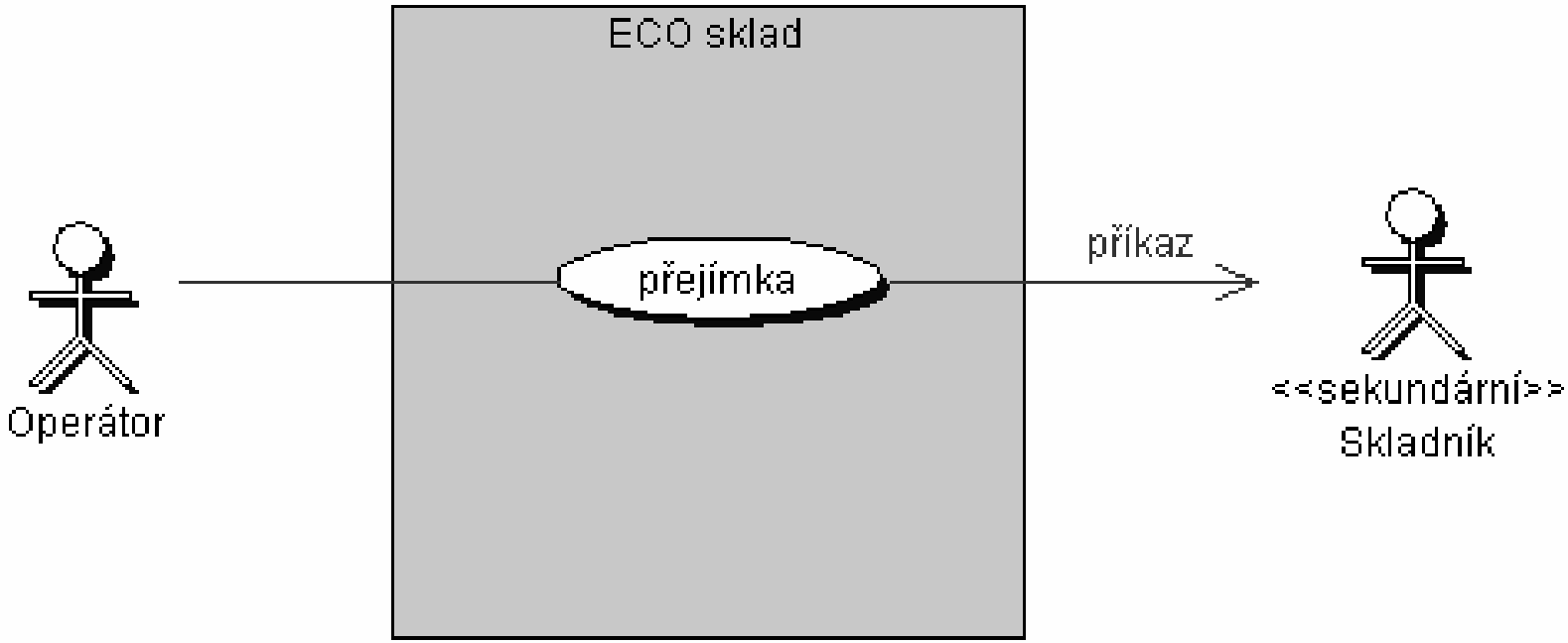
Doplňky k modelu jednání

- **sekundární aktér** - uživatelská role nebo spolupracující systém nutná pro činnost systému

Sekundární aktéři



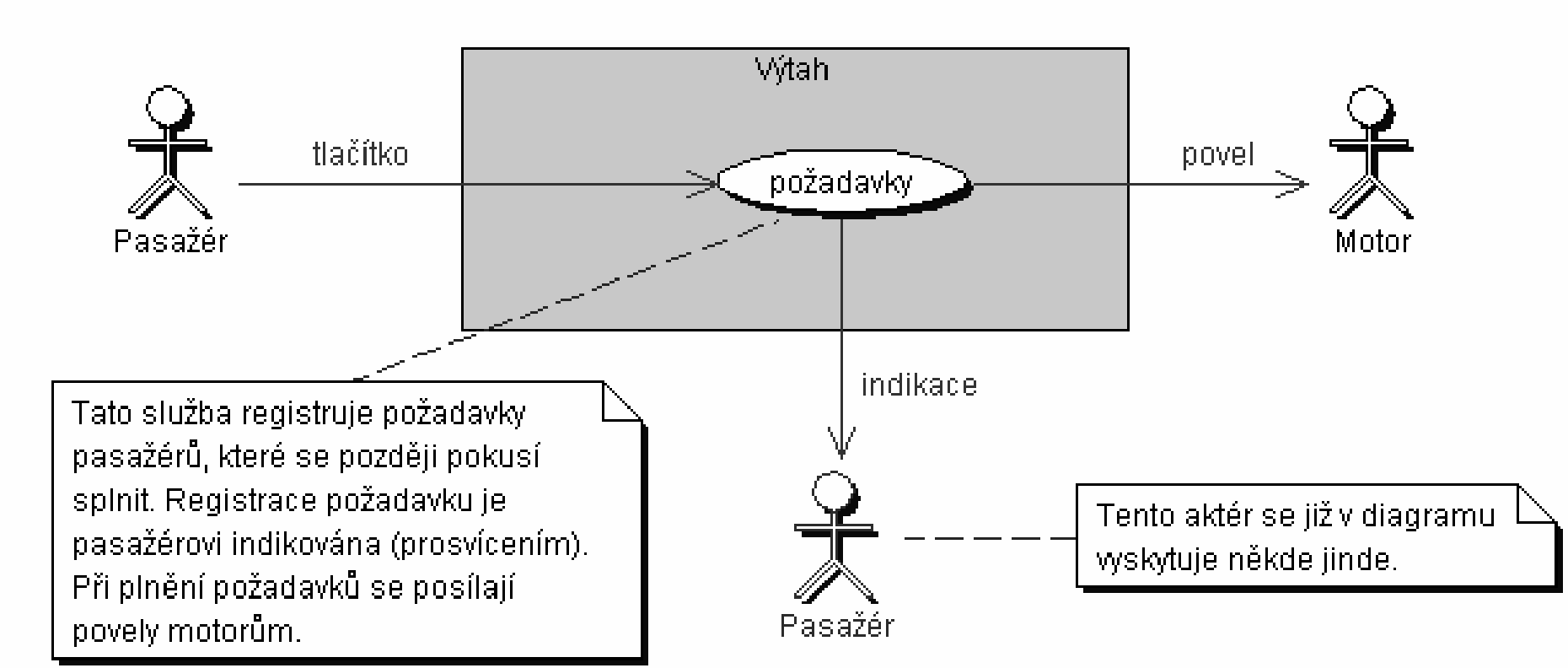
Sekundární aktéři



Doplňky k modelu jednání

- **orientovaná komunikace** - případ, kdy chceme vyznačit směr komunikace

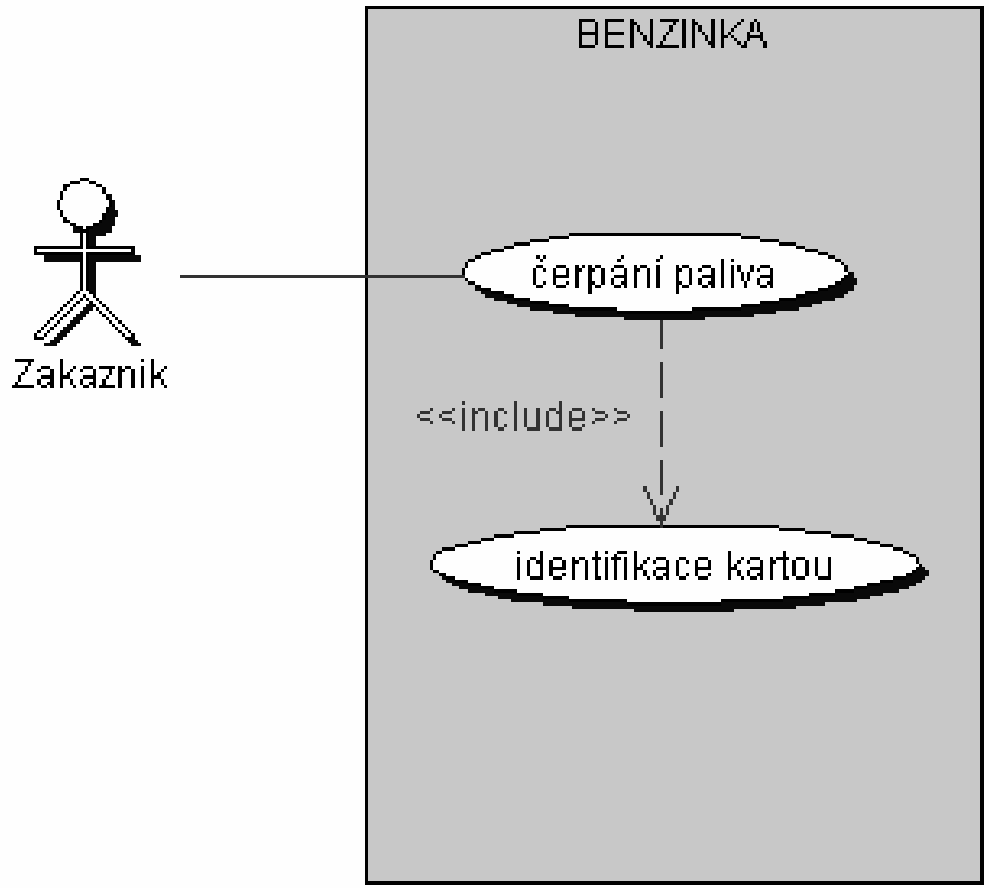
Orientovaná komunikace



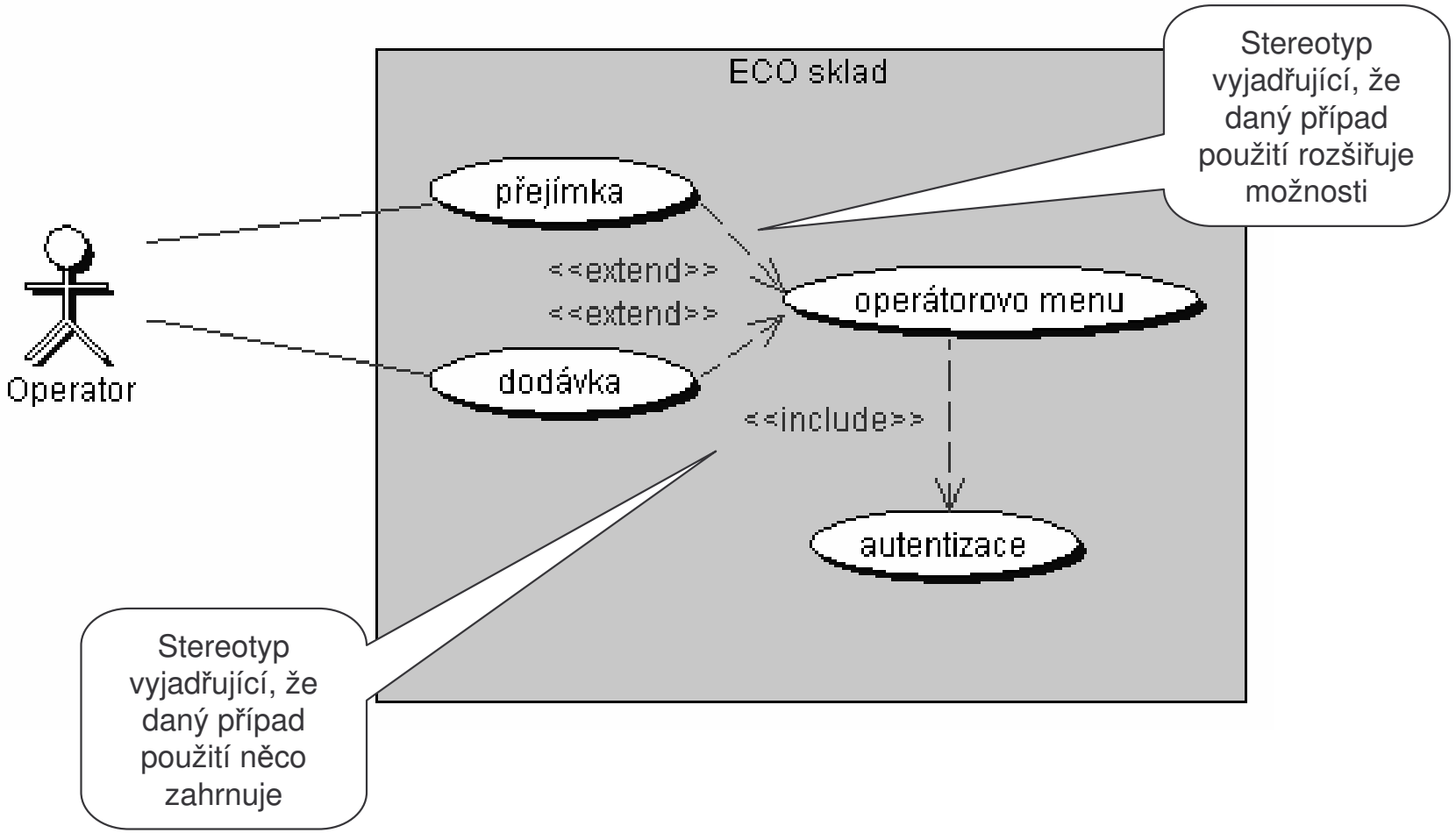
Doplňky k modelu jednání

- **vztahy mezi případy použití** - pokud chceme explicitně vyjádřit fakt, že takový vztah existuje
  - **<<include>>** - pokud jeden případ zahrnuje případ jiný (např. autentizace)
  - **<<extend>>** - pokud nějaký případ rozšiřuje chování (je zde možnost volby)
  - **generalizace/specializace**

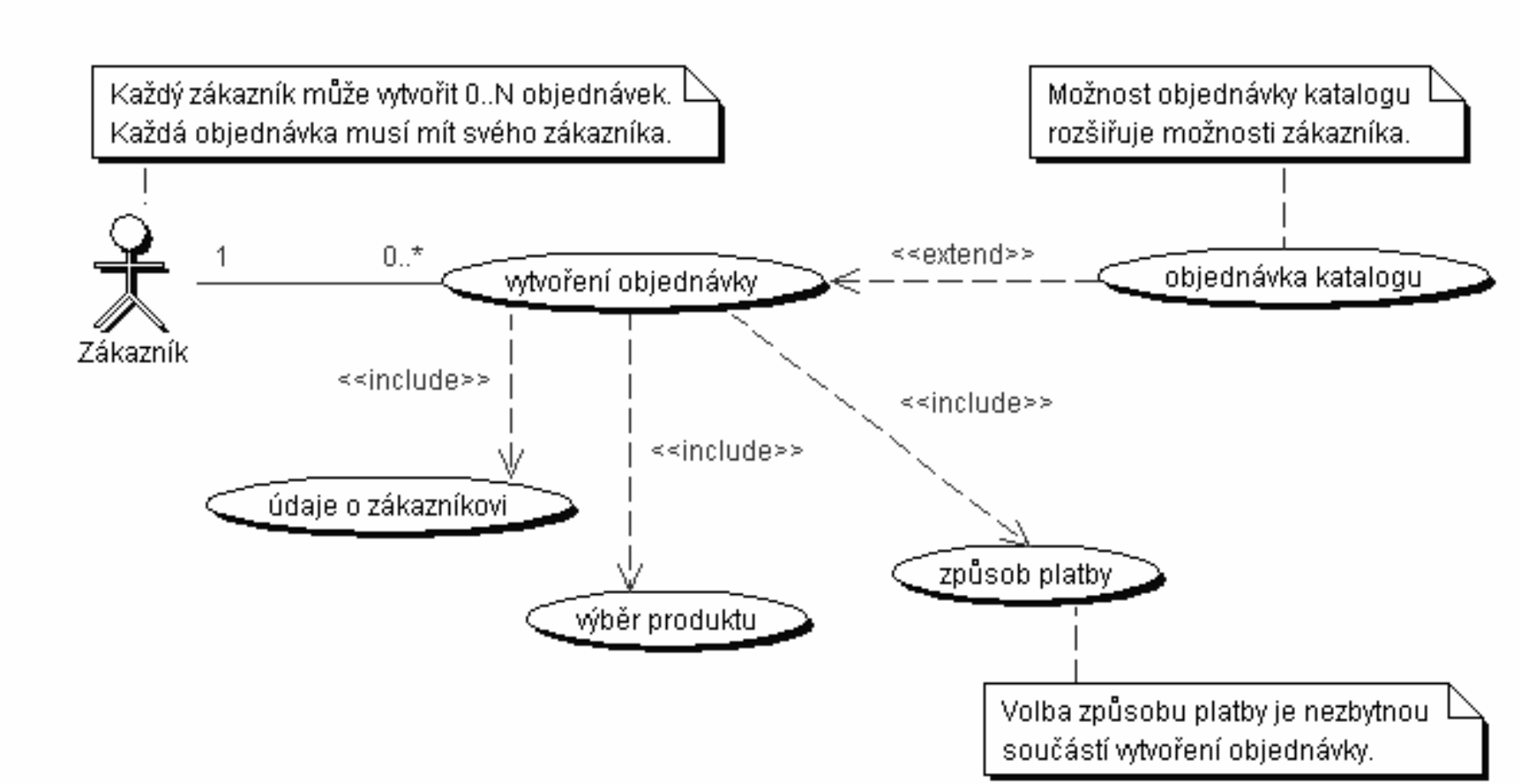
Vztahy mezi službami



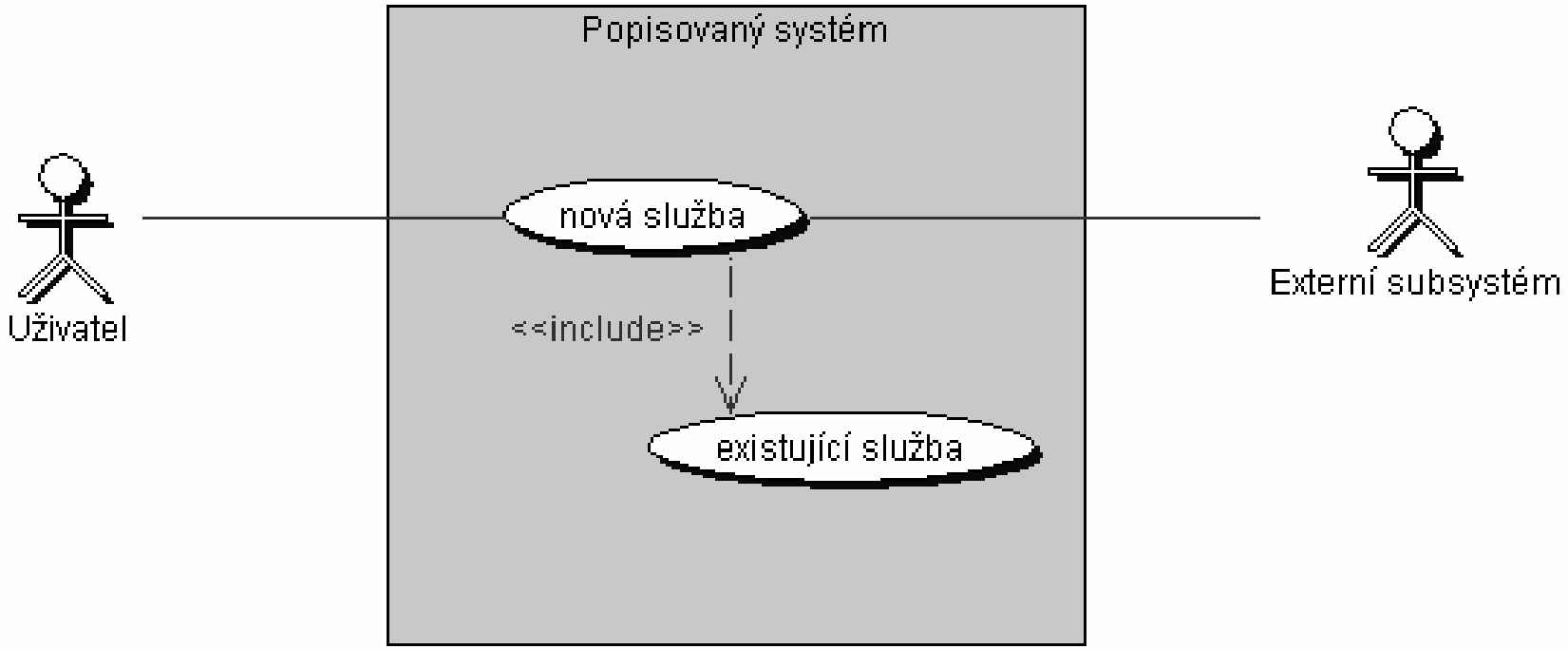
Vztahy mezi službami



Vztahy mezi službami

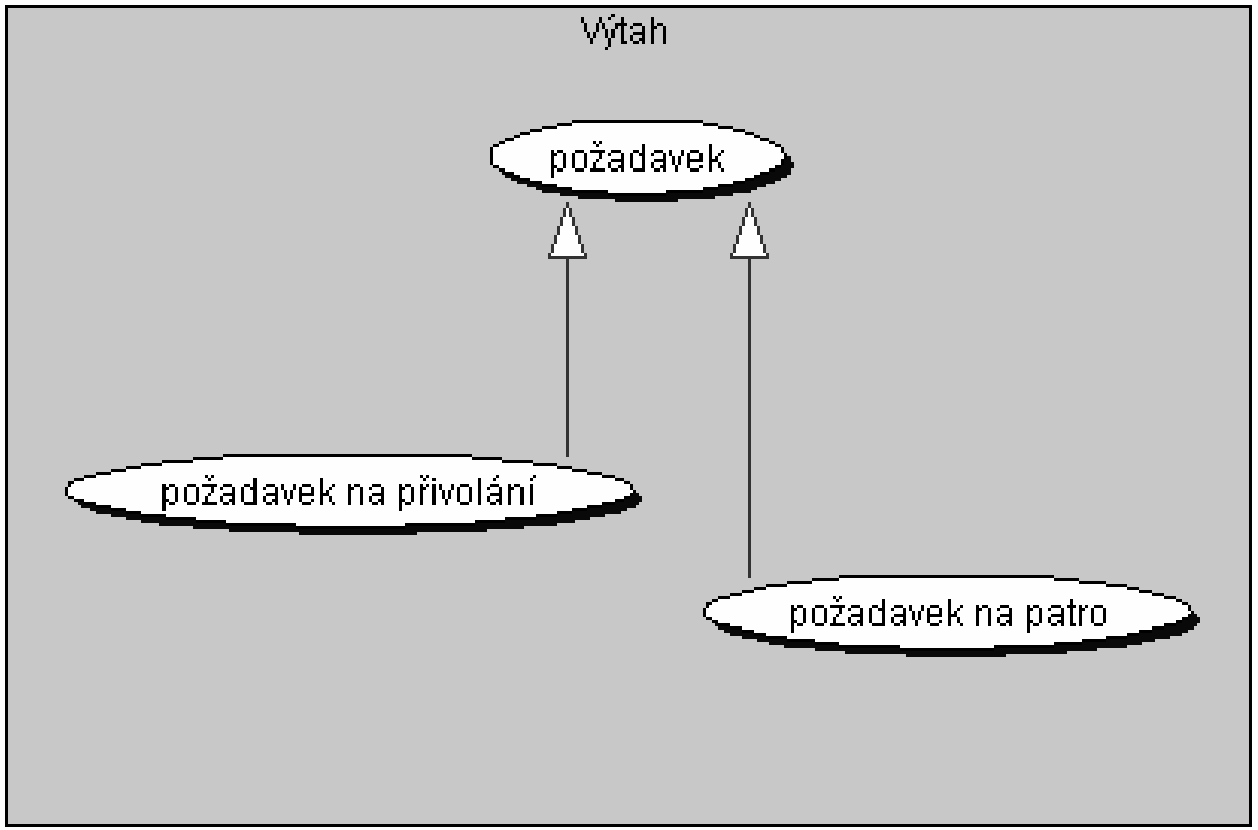


Kombinace různých prvků



Kombinace různých prvků v modelu jednání:
- nově vyvíjená část (nová služba)
- použité existující části (existující služba)
- část od jiného dodavatele (externí subsystém)

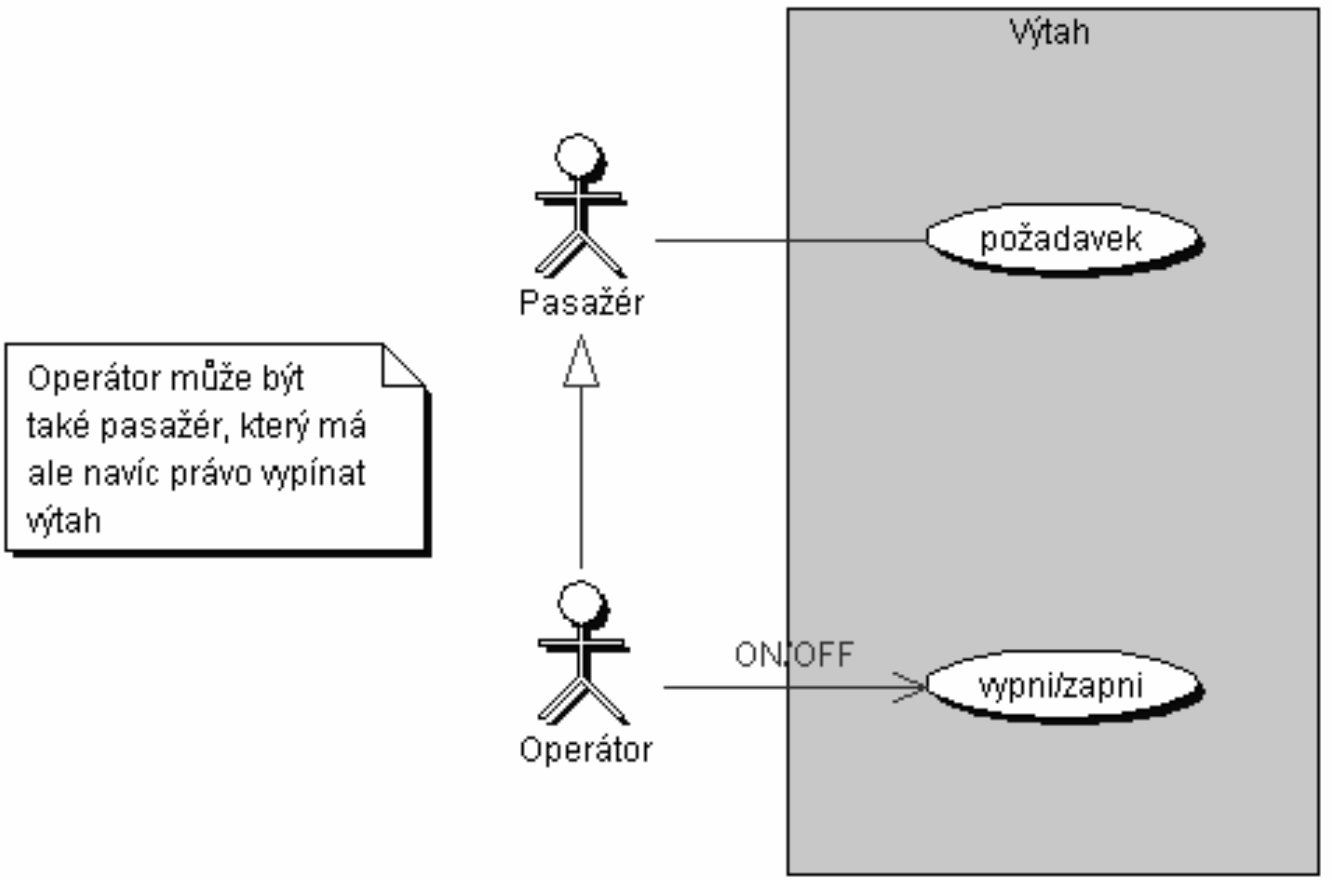
Generalizace služeb



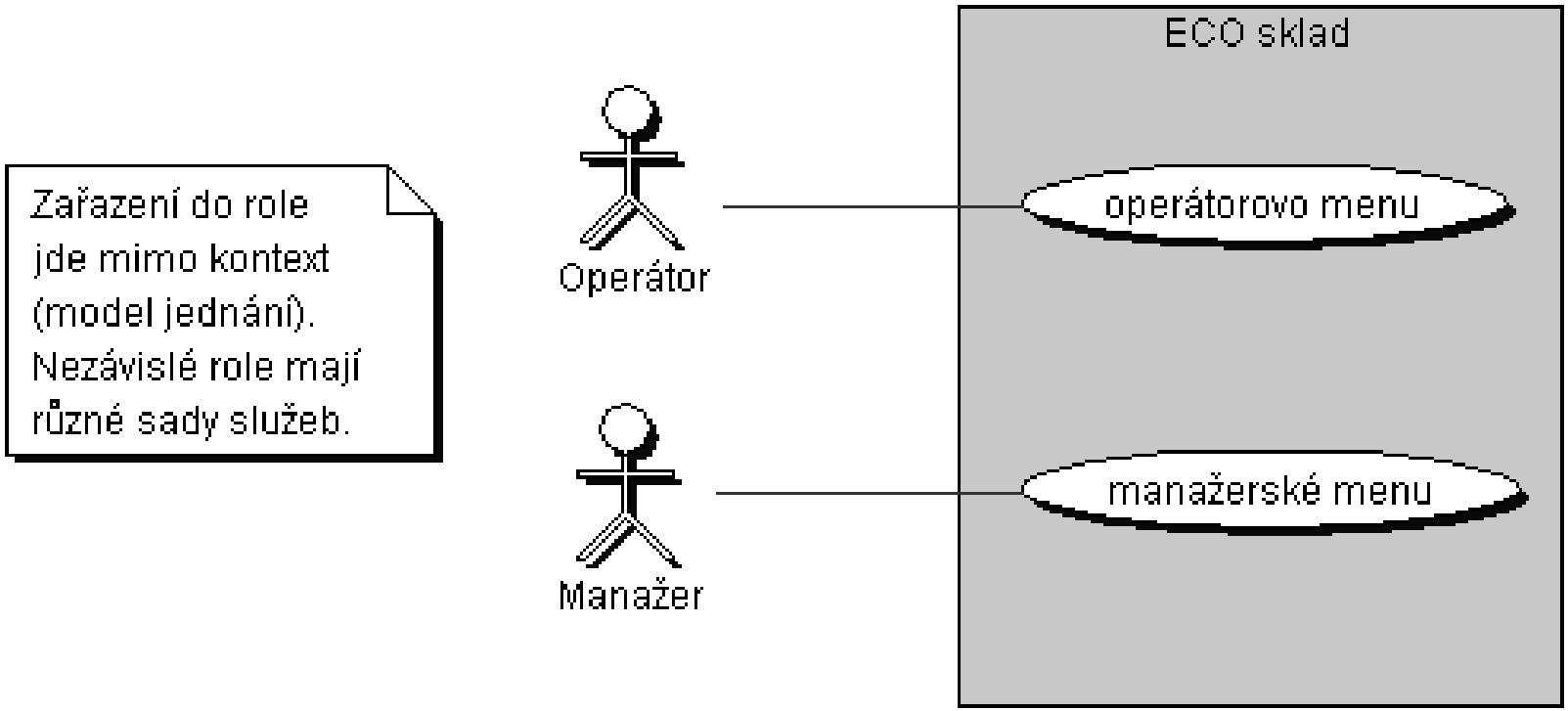
Doplňky k modelu jednání

- **vztahy mezi aktéry** - pokud chceme explicitně vyjádřit fakt, že takový vztah existuje
  - generalizace/specializace

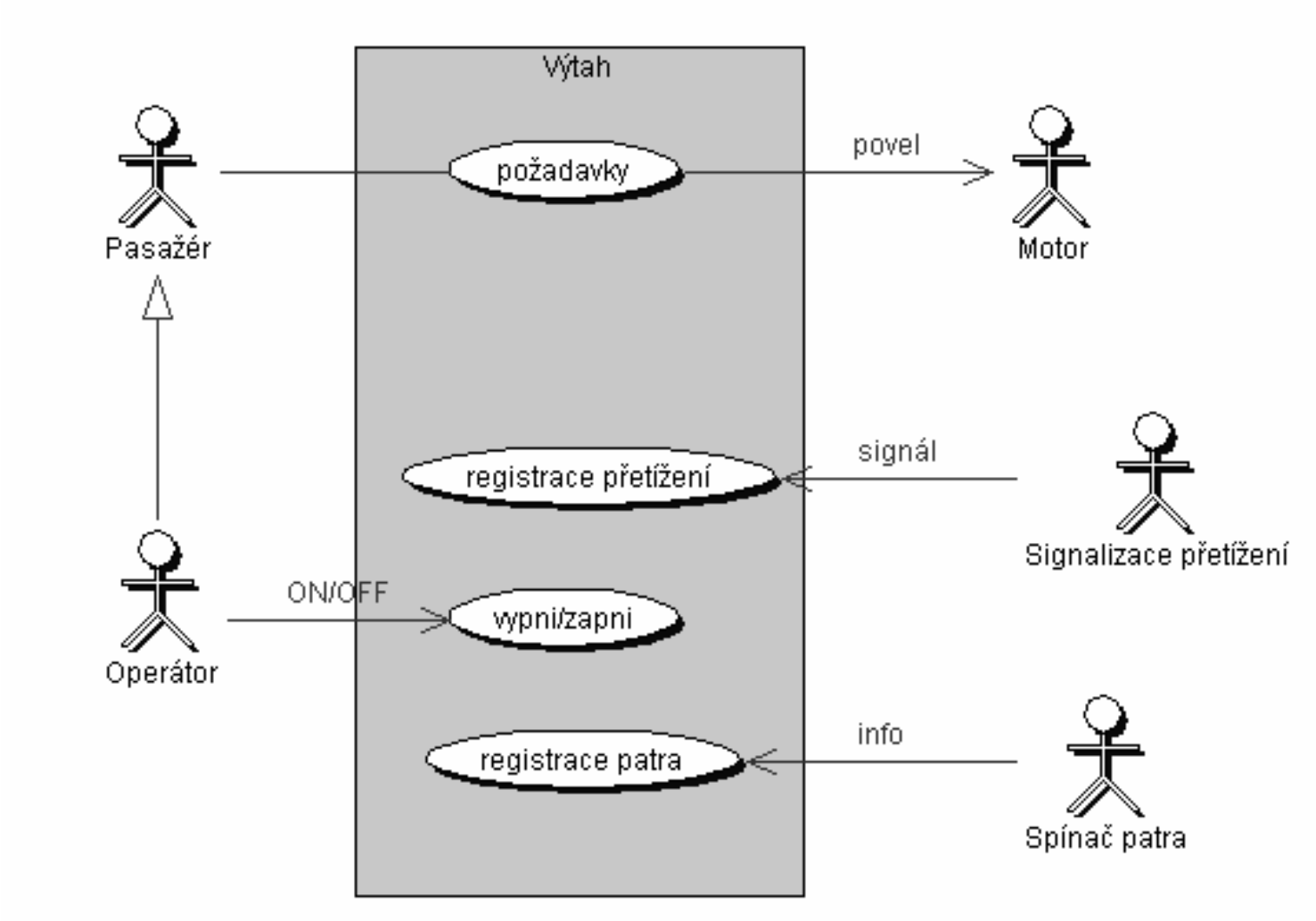
Generalizace aktérů



Autentizace do role

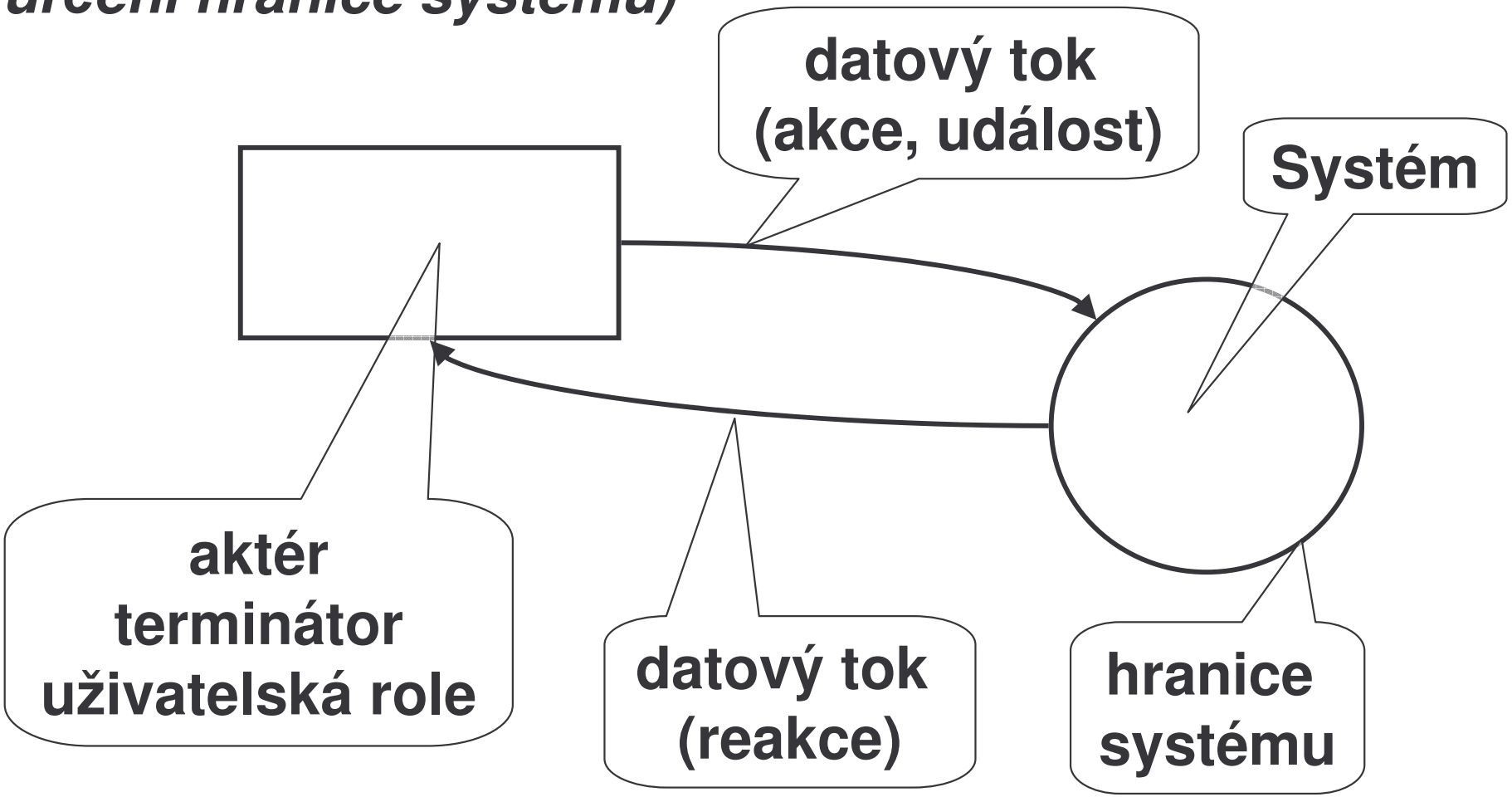


Model jednání pro Výtah



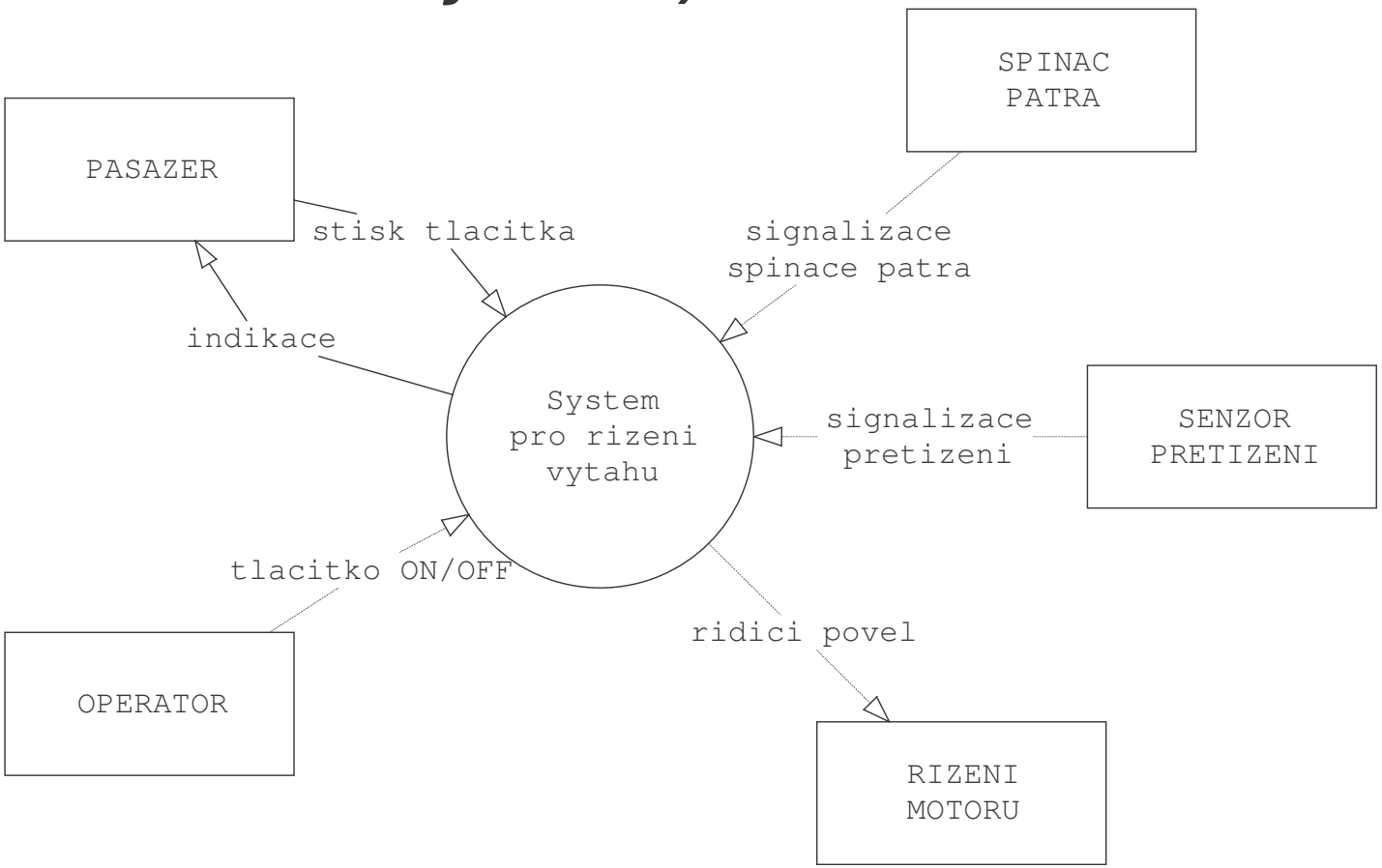
Kontextový diagram

(určení hranice systému)



Kontextový diagram pro “Výtah”

(určení hranice systému)



Chyby v definici kontextu

- Aktéři spolu komunikují mimo systém
- Není zdůrazněn dvojí výskyt aktéra
- Chybí datový tok pro některou událost
- Chybí datový tok pro některou reakci systému
- Datový tok není popsán v datovém slovníku
- Datový tok je popsán nevhodně (příliš obecně)
- Dva různí aktéři mají stejnou sadu událostí (pak to zřejmě nejsou různí aktéři)
- Za událost se považuje přihlášení do systému (zařazení do role jde mimo kontext)

Analýza

CO má systém umět

Návaznosti

- Dosud:
  - Úvodní studie, modelování požadavků
- Dnes:
  - Analýza
- Příště:
  - Architektura, modelem řízený vývoj

Analýza

Měla by odpovědět na otázku ***CO***?

- Musí proto definovat ***konceptuální model*** řešeného systému (***PIM – Platform Independent Model***)
- Musí definovat představu, s jakými ***daty*** bude systém pracovat, jaké ***služby*** bude systém poskytovat a jak se bude chování systému měnit - jaká bude ***dynamika*** systému
- Musí stanovit podmínky, za jakých je analytická dokumentace ***akceptovatelná***

Analytický (konceptuální, PIM) model

- (konceptuální) funkční model
  - Systém má poskytovat nějaké služby. V úvodní studii jsme se orientačně dohodli, jaké to služby budou, teď je třeba to říci přesně.
- (konceptuální) datový model
  - Aby bylo možno služby poskytovat, je potřeba pracovat s daty. V úvodní studii jsme se orientačně dohodli, jaká data to budou, teď je třeba to říci přesně.
- (konceptuální) dynamický model
  - Systém, nebo jeho prvky často vykazují dynamiku – jejich chování se mění na základě různých okolností. Teď je třeba říci přesně jak.

Datový model

- notace
- konceptuální model tříd nebo ER-model (diagramy + textový popis)
- další integritní omezení, která nejsou zachycena v diagramech
- datový slovník

Funkční model

- notace
- model jednání (seznam událostí, příp. scénáře)
- pokud scénář obsahuje složitější aktivity, pak dekompozice těchto aktivit na popisy jednodušší – mohou to být podrobnější scénáře, diagramy aktivit, hierarchická sada diagramů datových toků (DFD - kontextový diagram + diagramy úrovně 0,1,… + popis)
- minispecifikace elementárních operací
- datový slovník

Dynamický model

- notace
- scénáře životních cyklů
- stavové diagramy
- datový slovník

Postup při analýze I.

- Vstup:
  - úvodní studie, katalog požadavků - CIM
- Výstup:
  - analytická dokumentace - PIM
- Postup:
  - paralelně zpracuj koncept a projekt

Postup při analýze II.

Zpracování konceptu:

- Vstup: CIM
  - deklarace záměru, odborný článek, katalog požadavků, seznam aktérů, seznam událostí, model jednání, kontext, 1.verze datového slovníku (z úvodní studie)
- Výstup: PIM
  - konceptuální analytický model (datový, funkční a dynamický model, 2.verze datového slovníku)

Postup při analýze III.

Zpracování projektu:

- Vstup:
  - seznam úloh, harmonogram (z úvodní studie)
- Výstup:
  - projektová dokumentace (projektový deník, seznam zdrojů, matice zodpovědností, harmonogram, plán testů, akceptační test)

Úvod do notace UML

Unified Modeling Language

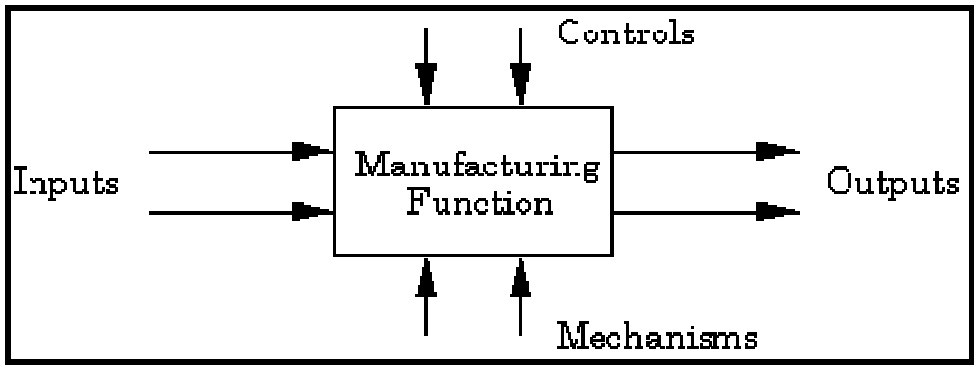
- jak se to zapisuje

Proč UML?

- UML je unifikovaný vyjadřovací prostředek pro dokumentaci obsahové části informatických projektů (dokumentace projektu obsahuje ještě tzv. projektovou dokumentaci, která popisuje projekt jako takový).
- UML umožňuje vyjádření dokumentace analýzy, návrhu i implementace.
- UML byl vyvinut na základě zkušeností s mnoha různými metodikami.
- UML umožňuje i vyjádření strukturovaného přístupu, ale byl určen zejména pro objektově-orientovaný přístup.

Alternativní notace

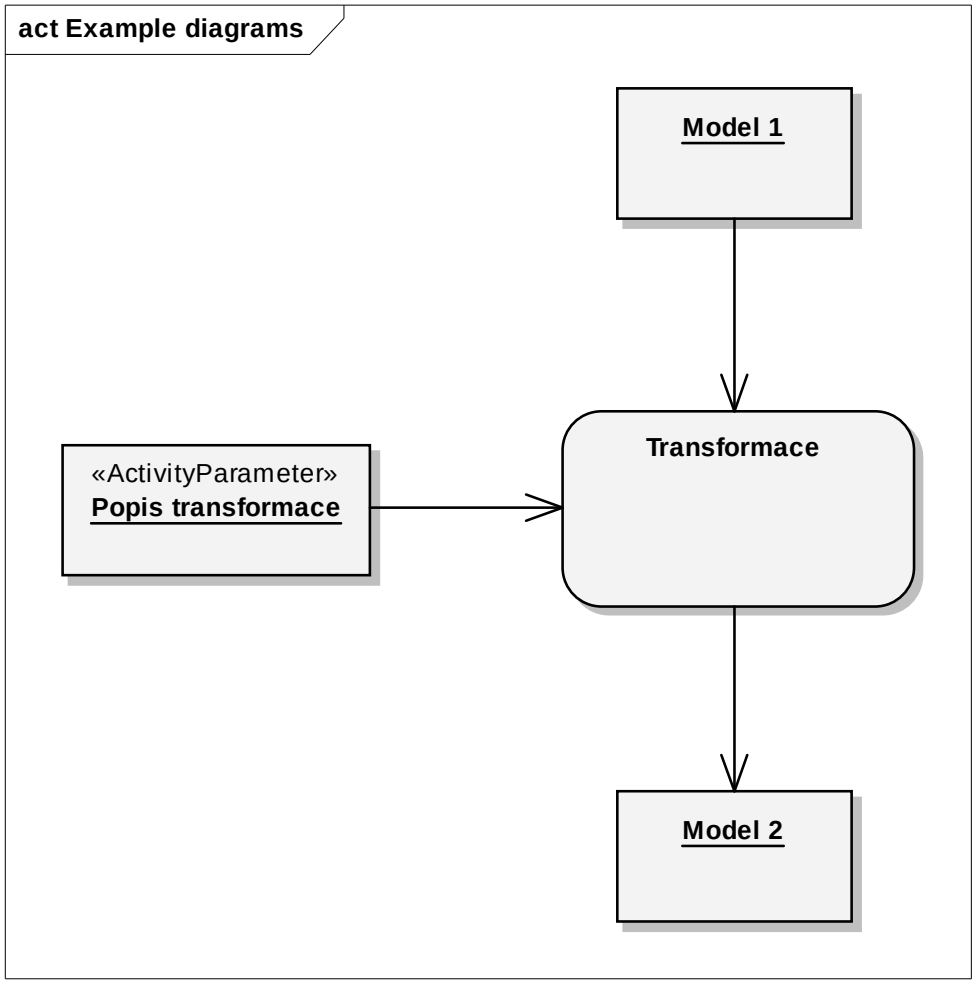
- Integrated Definition – IDEF (U.S. Air Force - http://www.idef.com)



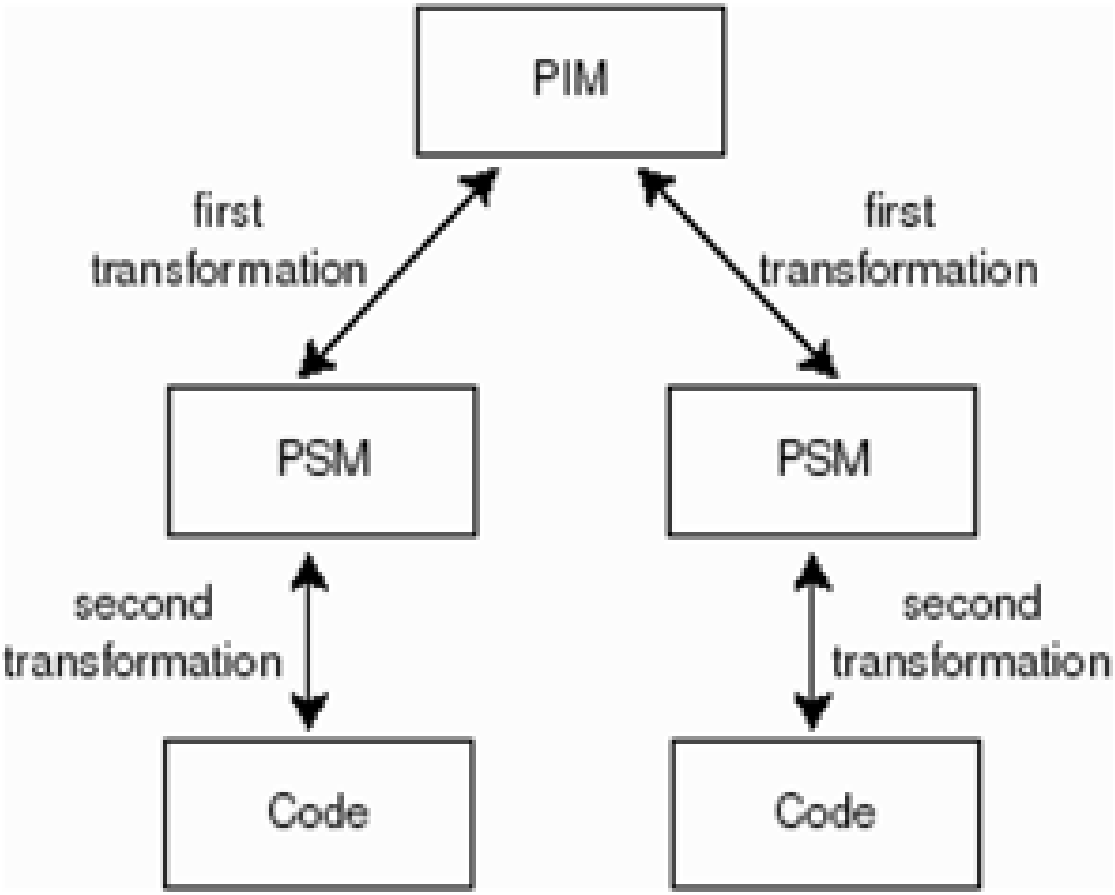
Alternativní notace

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

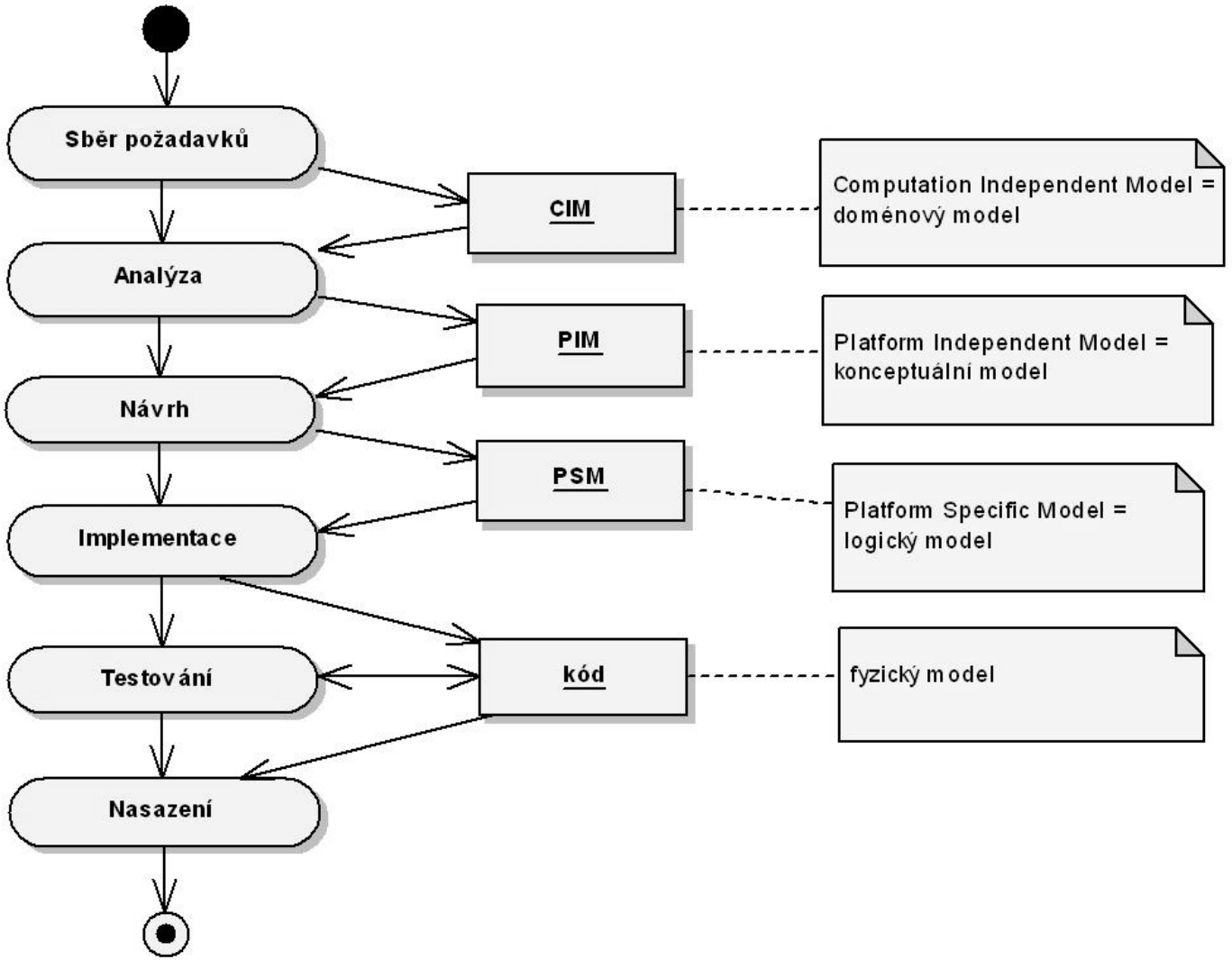
Vývoj software jako posloupnost transformací



Modelem řízený vývoj (MDD)



Modelem řízený vývoj v UML



6 dobrých zásad tvorby SW

- Vyvíjej iterativně a přírůstkově
- Řádně rozpoznej a zpracuj požadavky
- **Modeluj visuálně (=> UML)**
- Používej komponentové technologie
- Kontinuálně ověřuj kvalitu, testuj
- Snaž se být připraven na změny

Best practices
IBM Rational

Co to je UML?

- **UML (Unified Modeling Language)** je: *„Standard konsorcia OMG (Object Management Group) pro záznam, vizualizaci a dokumentaci artefaktů systémů s převážně softwarovou charakteristikou“.*

Co není UML?

- UML není žádná metodika – slouží pouze pro vyjádření artefaktů určitého typu, které metodiky požadují. Jsou metodiky, které UML intensivně využívají (např. UP, RUP), jiné nikoliv.
- UML není všespasitelné (U znamená unifikovaný, nikoli univerzální).

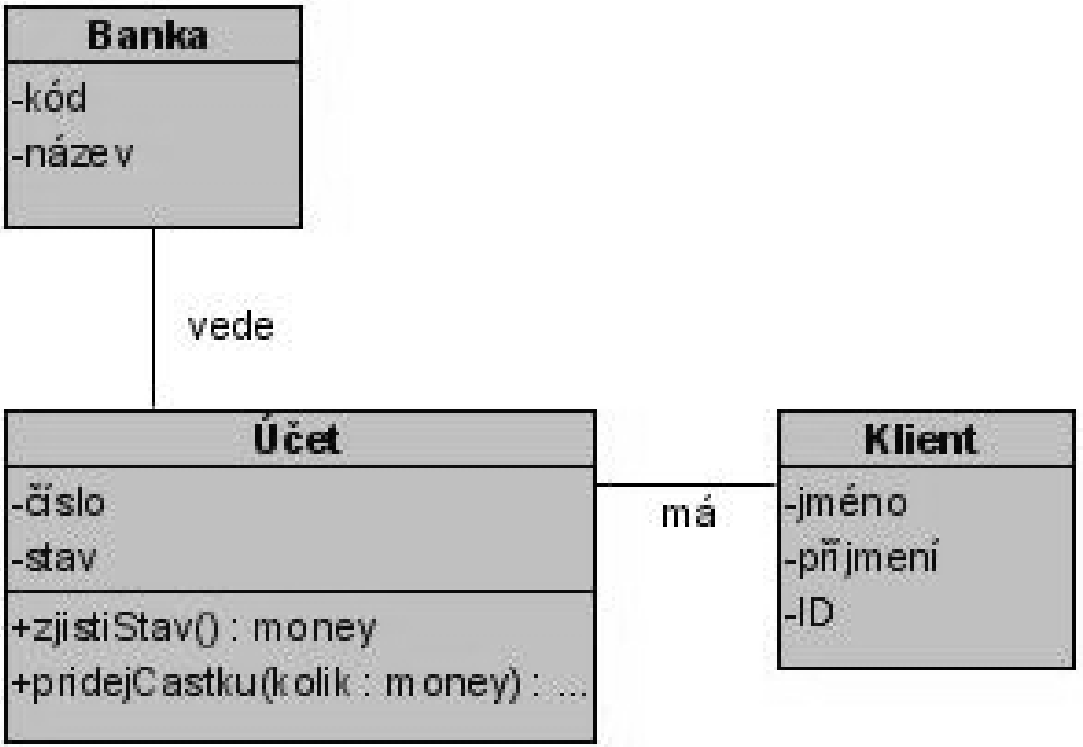
Tři úrovně využívání UML

- Jako **notaci** pro zachycení artefaktů, na kterých se je potřeba domluvit. Lze jej využít přímo (nakreslíme obrázek, abychom si upřesnili imlementaci), ale i zpětně (obrázek nám poslouží k pochopení existujícího kódu).
- Jako **dokumentační prostředek**, ve kterém dokumentujeme systém, či jeho části.
- Jako **programovací jazyk**, ze kterého se skutečná implementace (přesněji asi jen její kostra) generuje.

Základní diagramy UML (1.5)

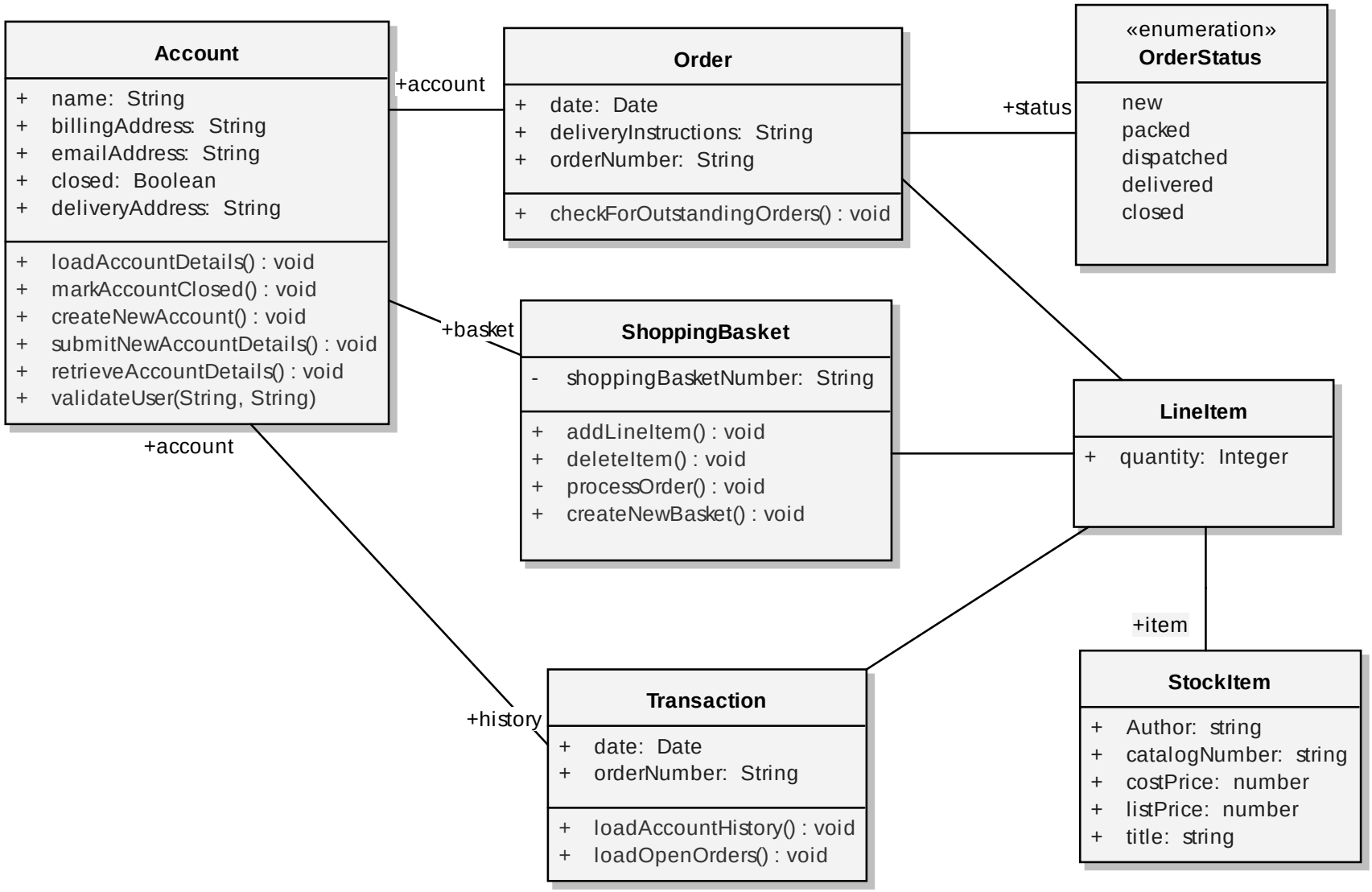
- diagramy tříd
- diagramy případů užití (model jednání)
- stavové diagramy
- scénáře (diagramy sekvencí)
- diagramy komunikace (spolupráce)
- diagramy aktivit (činností)
- diagramy komponent
- diagramy nasazení

Diagramy tříd

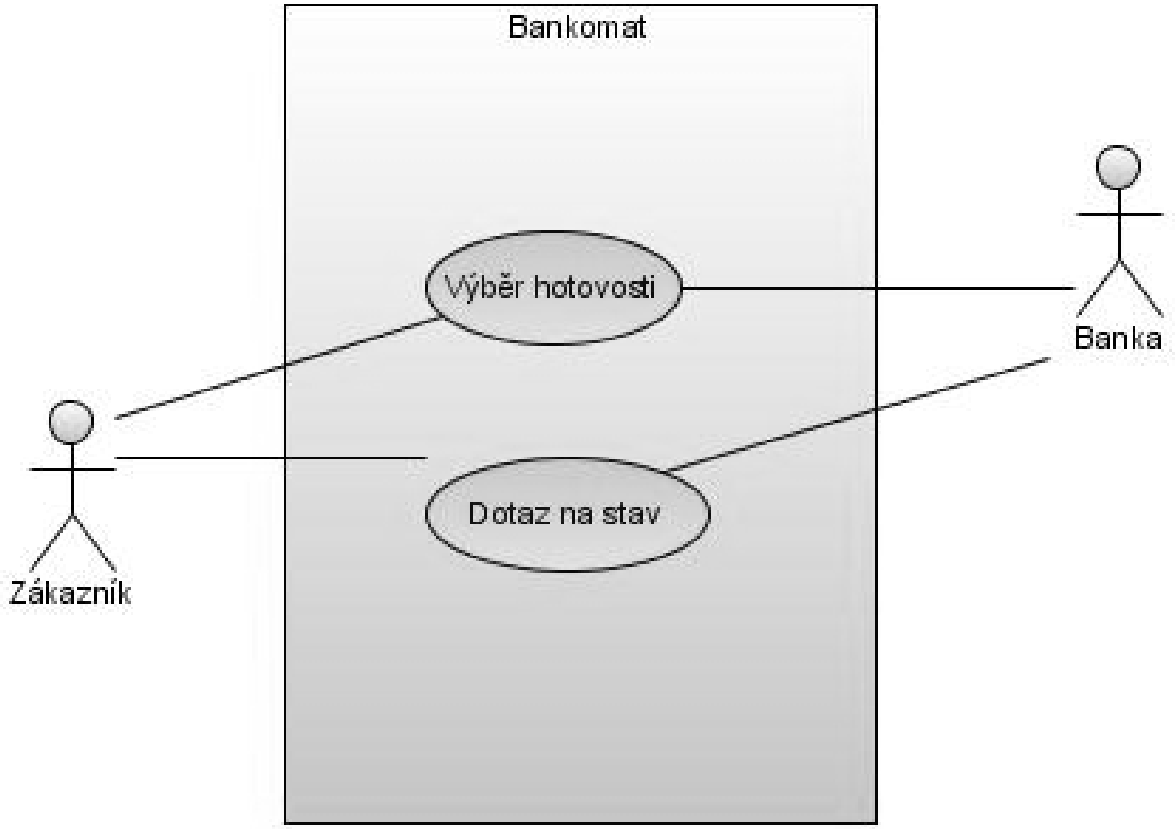


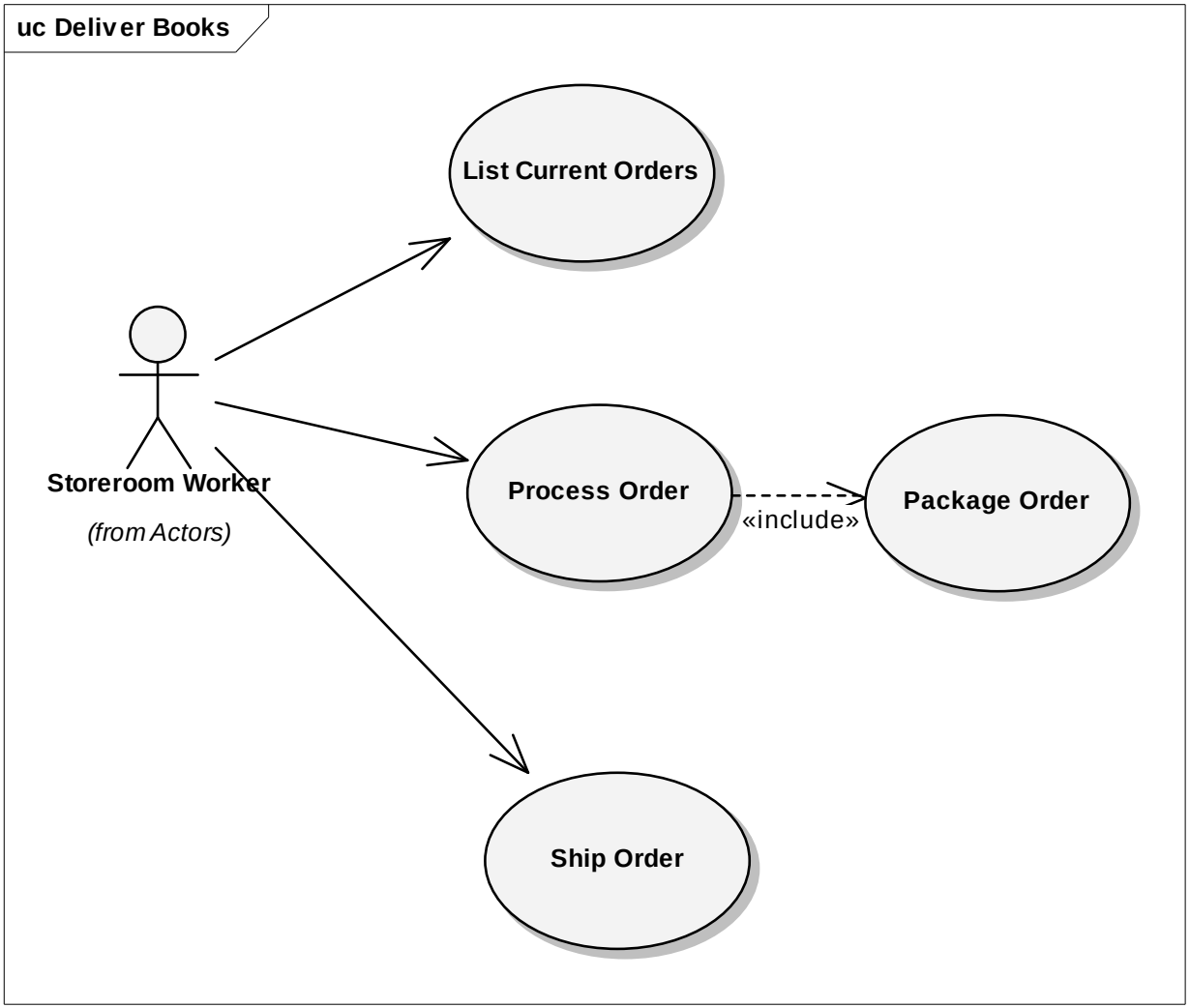
object Class Model

Konceptuální model pro elektronický obchod (PIM)

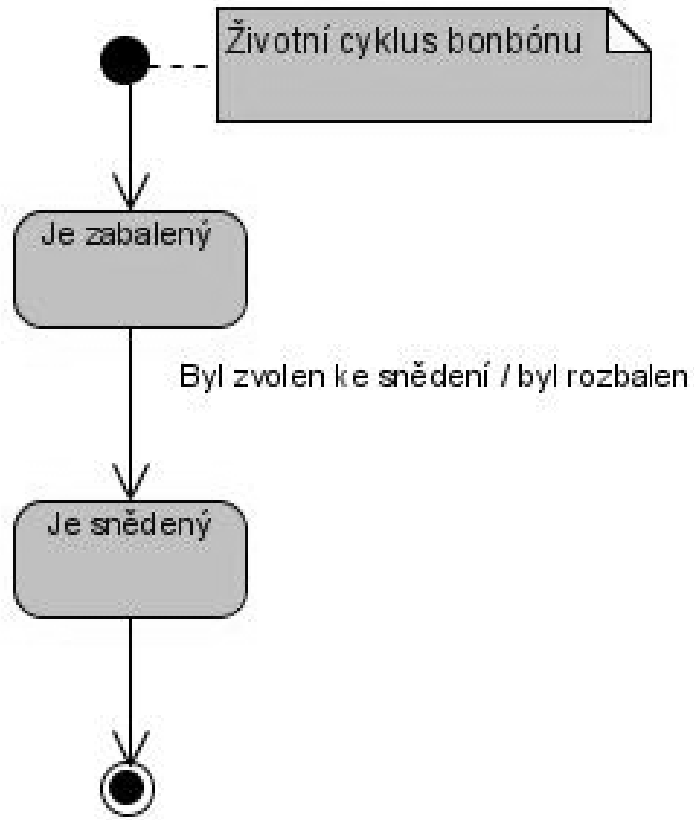


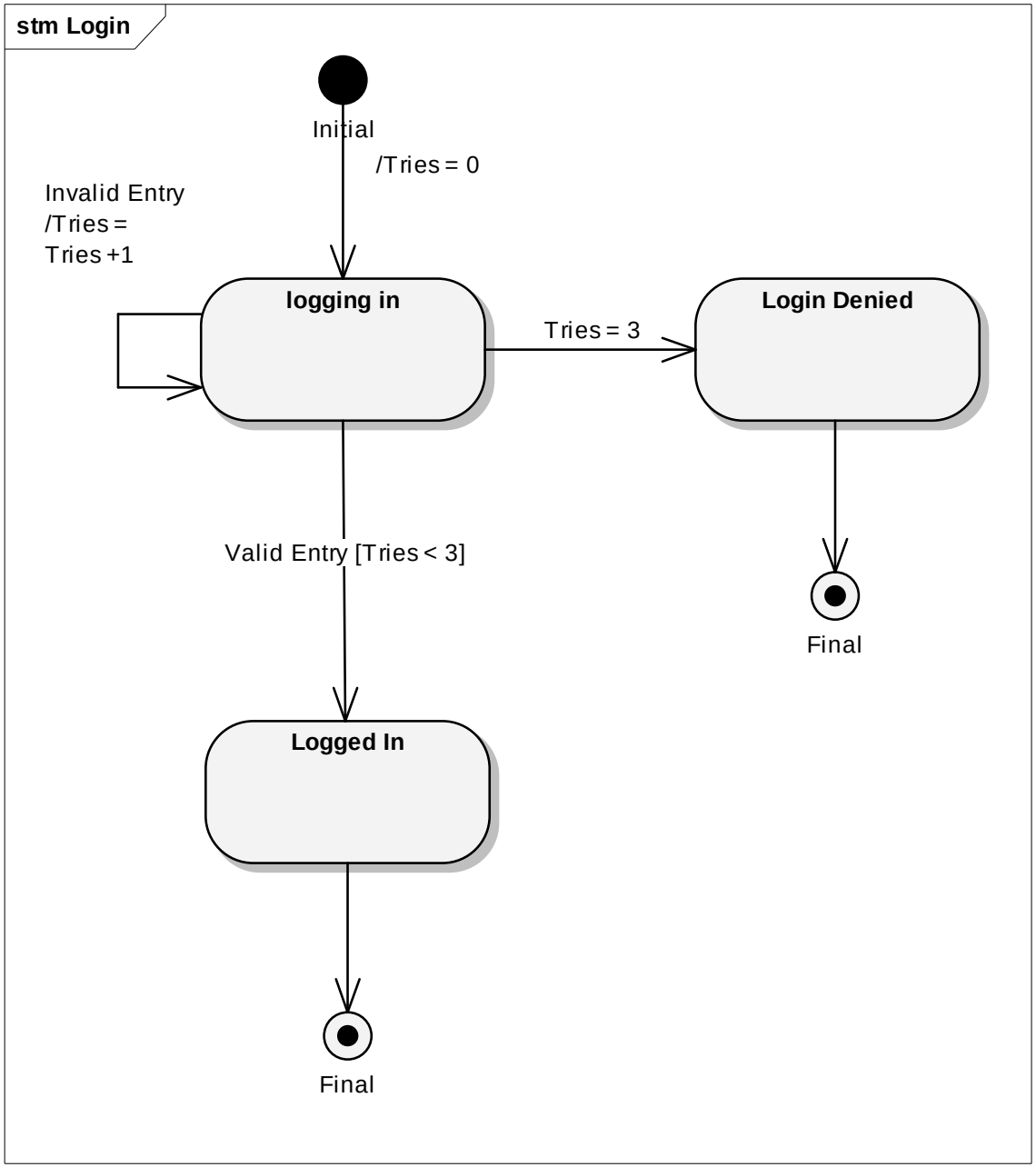
Diagramy případů užití (základ modelu jednání)



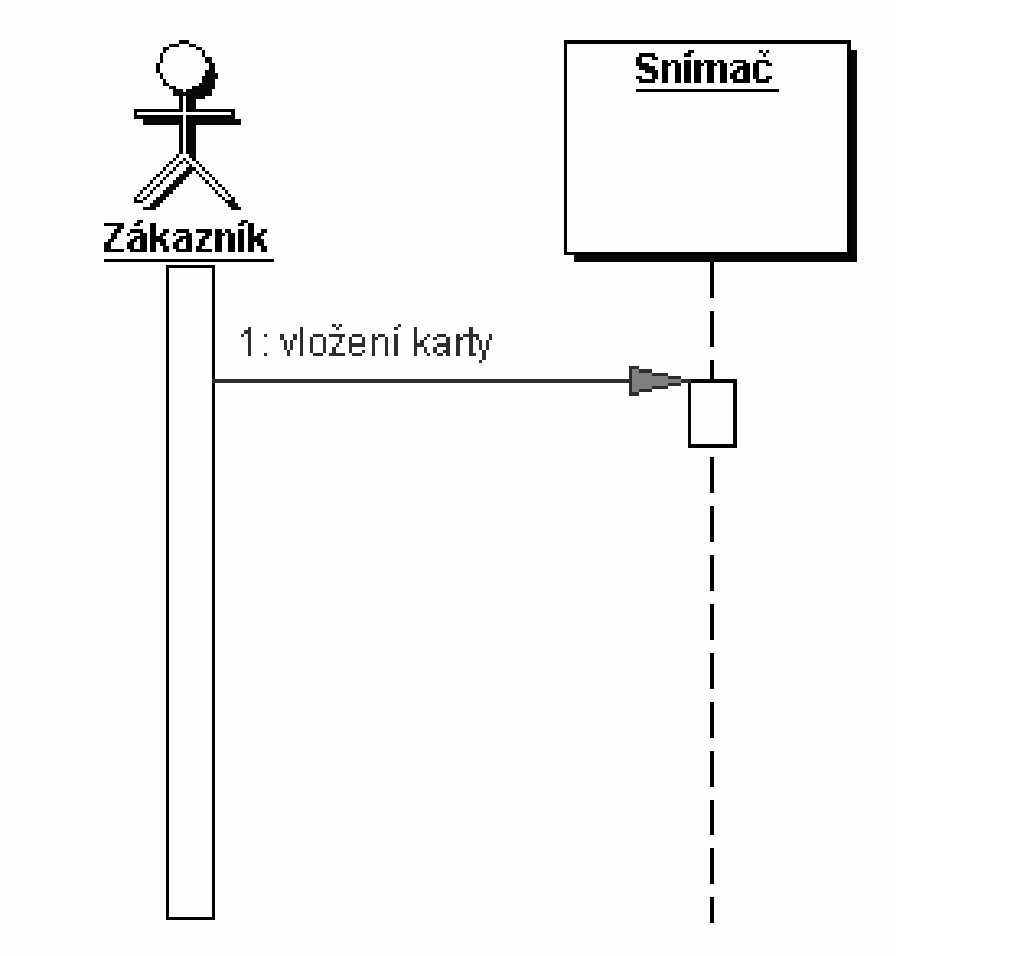

uc Deliver Books
List Current Orders
Storeroom Worker
(from Actors)
Process Order
«include»
Package Order
Ship Order

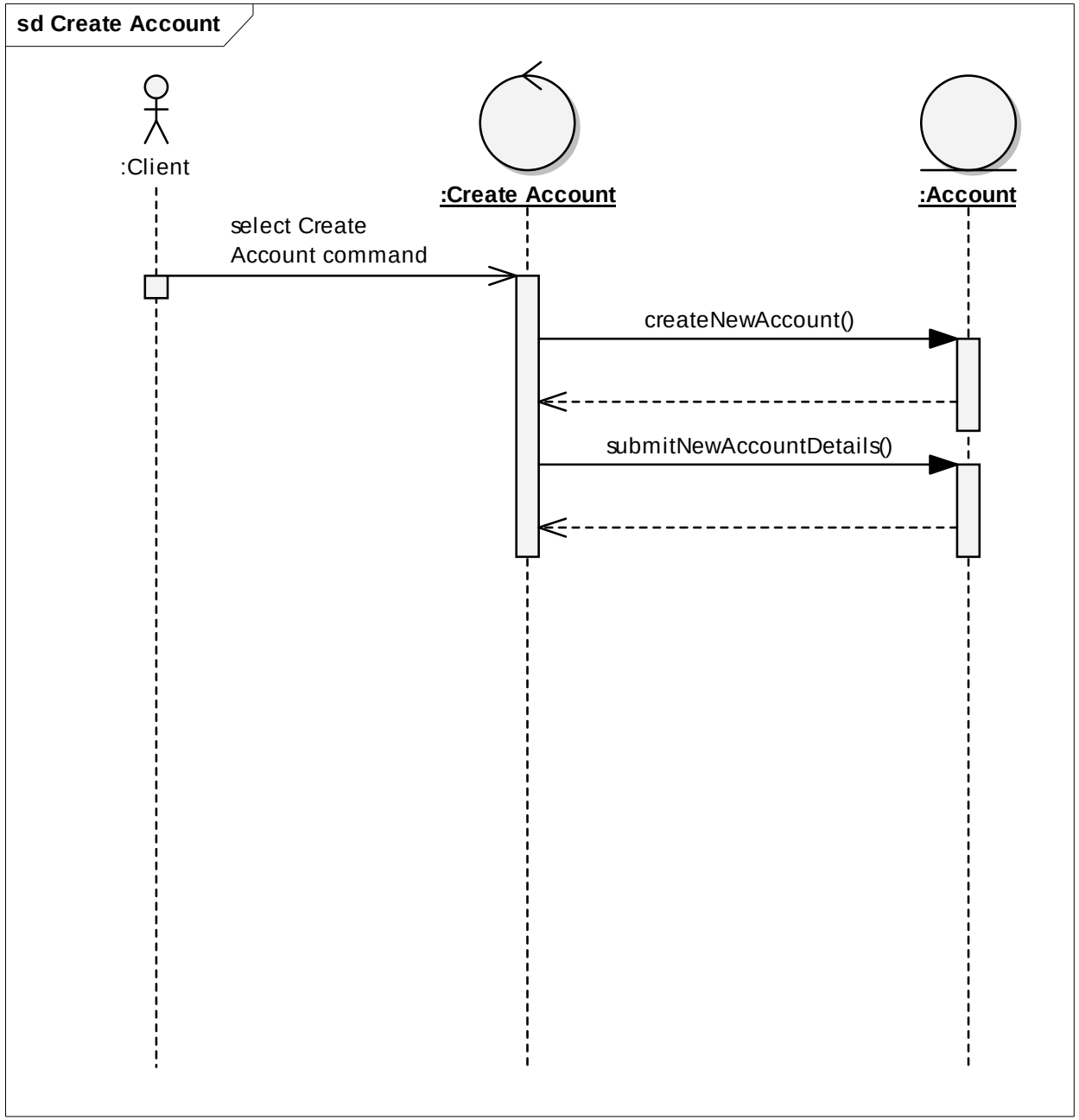
Stavové diagramy



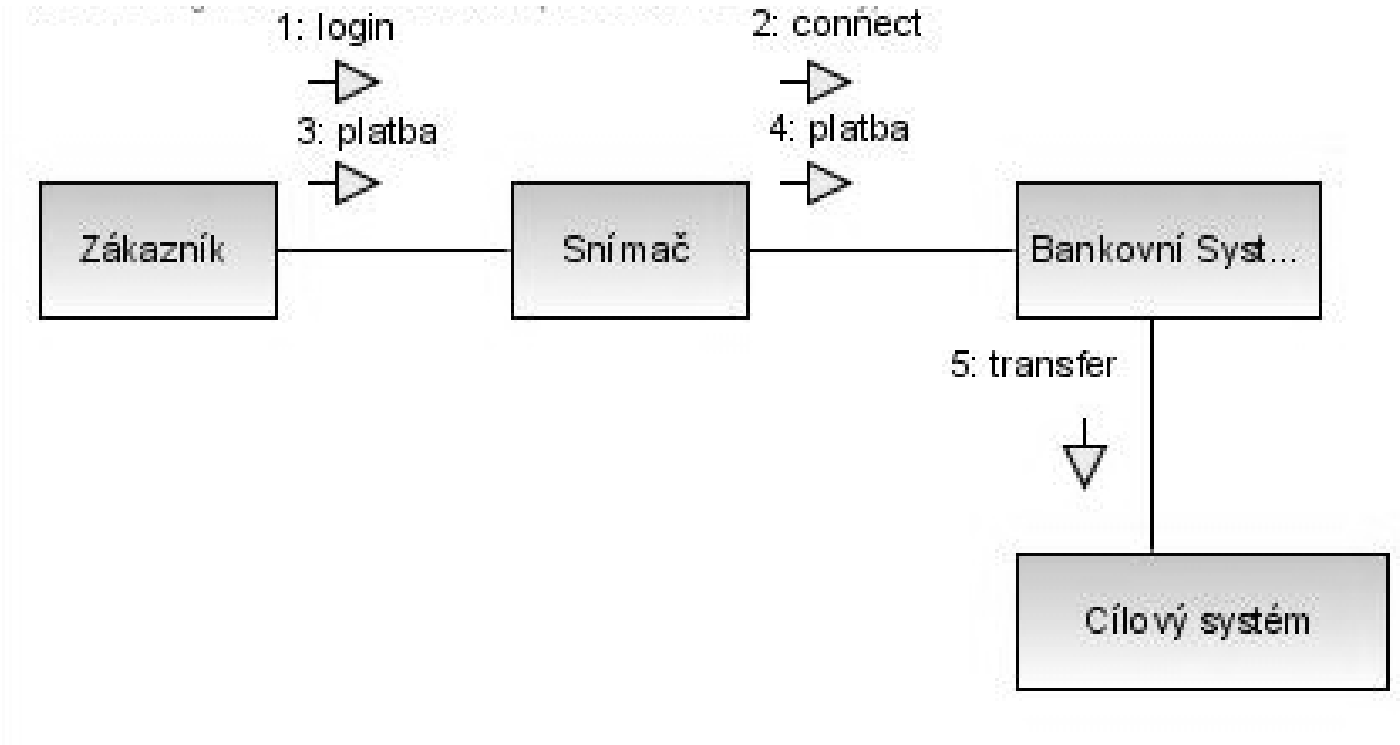

stm Login
Initial
/Tries = 0
Invalid Entry
/Tries =
Tries +1
logging in
Tries = 3
Login Denied
Valid Entry [Tries < 3]
Final
Logged In
Final

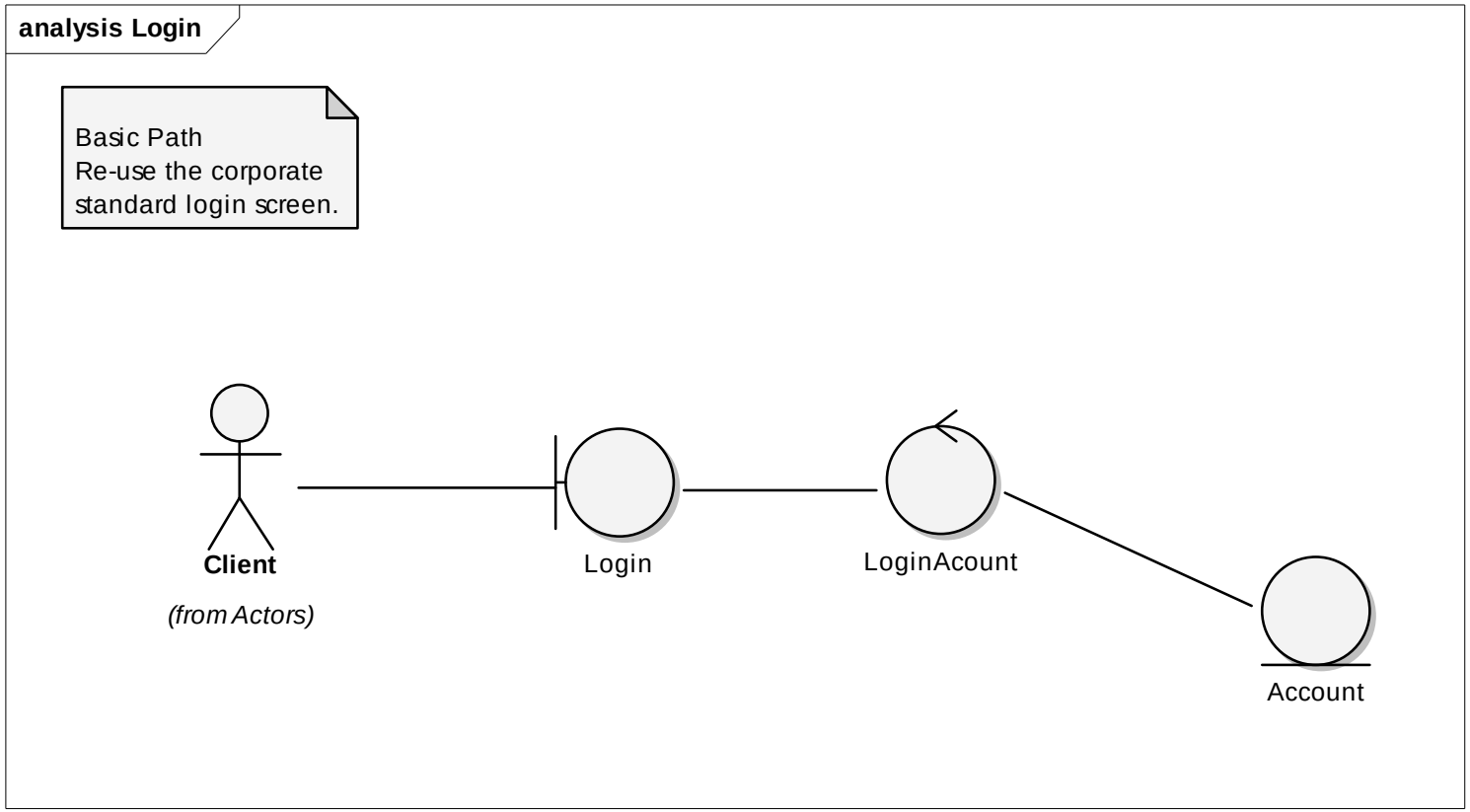
Scénáře (diagramy sekvencí)



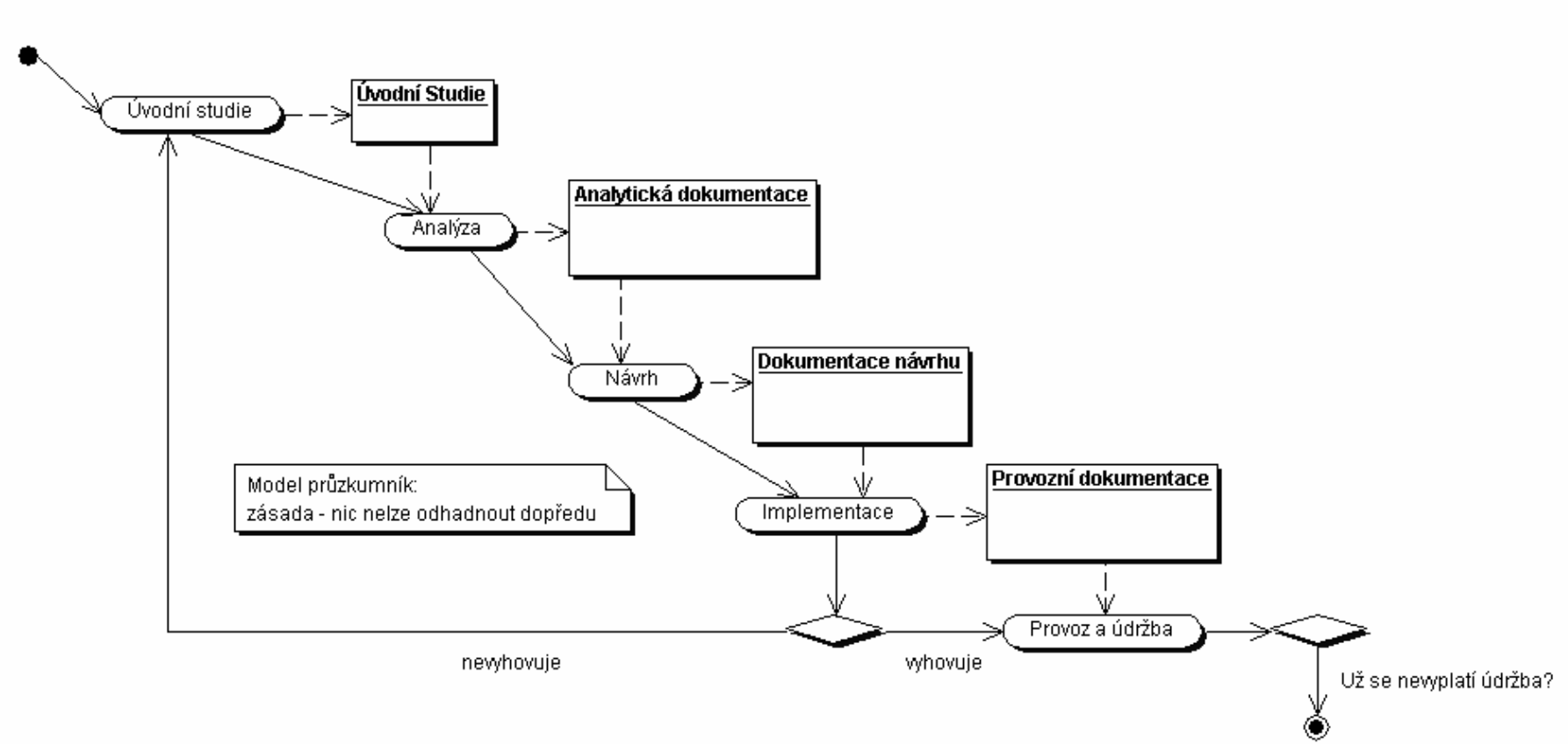

sd Create Account
:Client
:Create Account
:Account
select Create
Account command
createNewAccount()
submitNewAccountDetails()

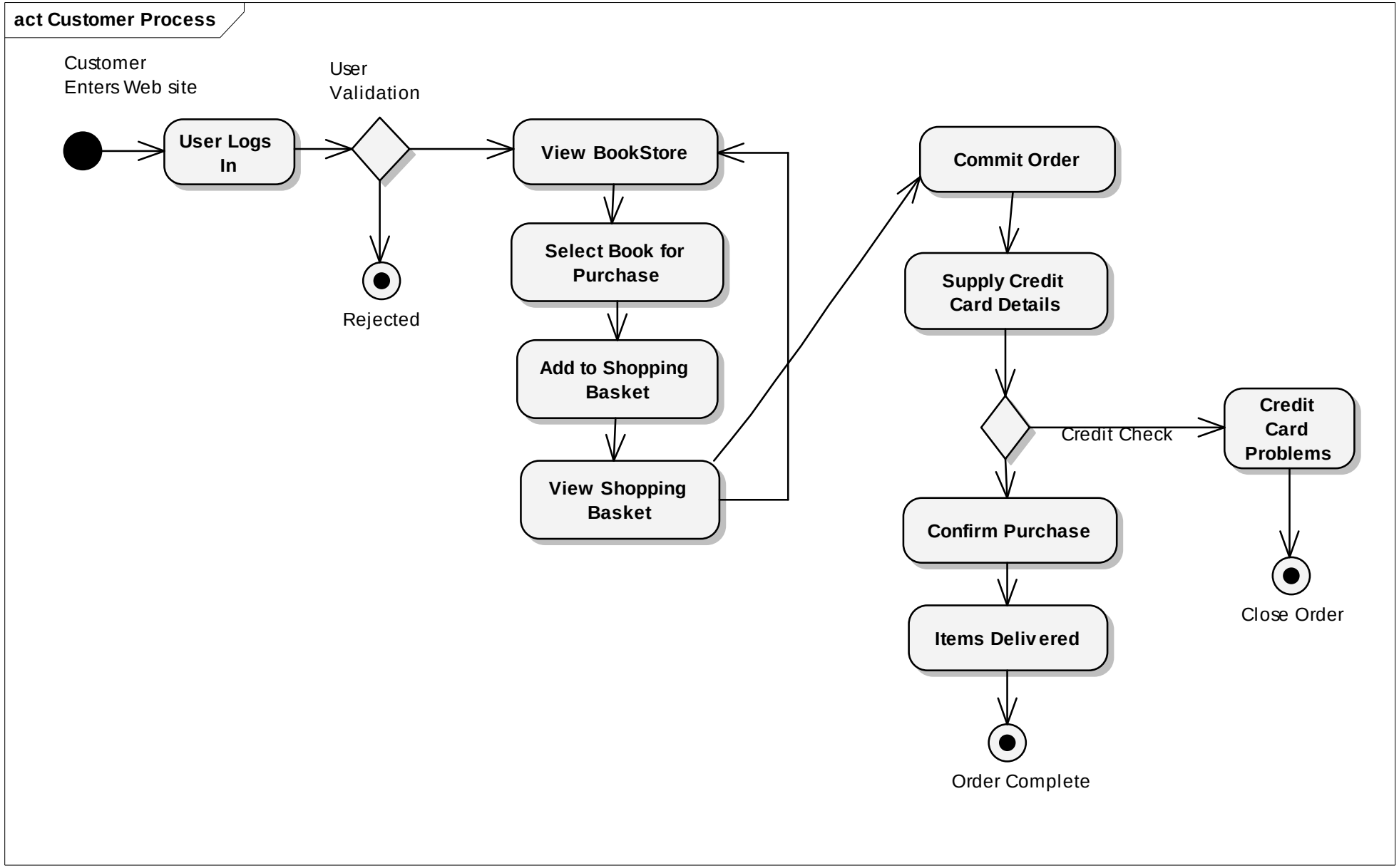
Diagramy komunikace (spolupráce)



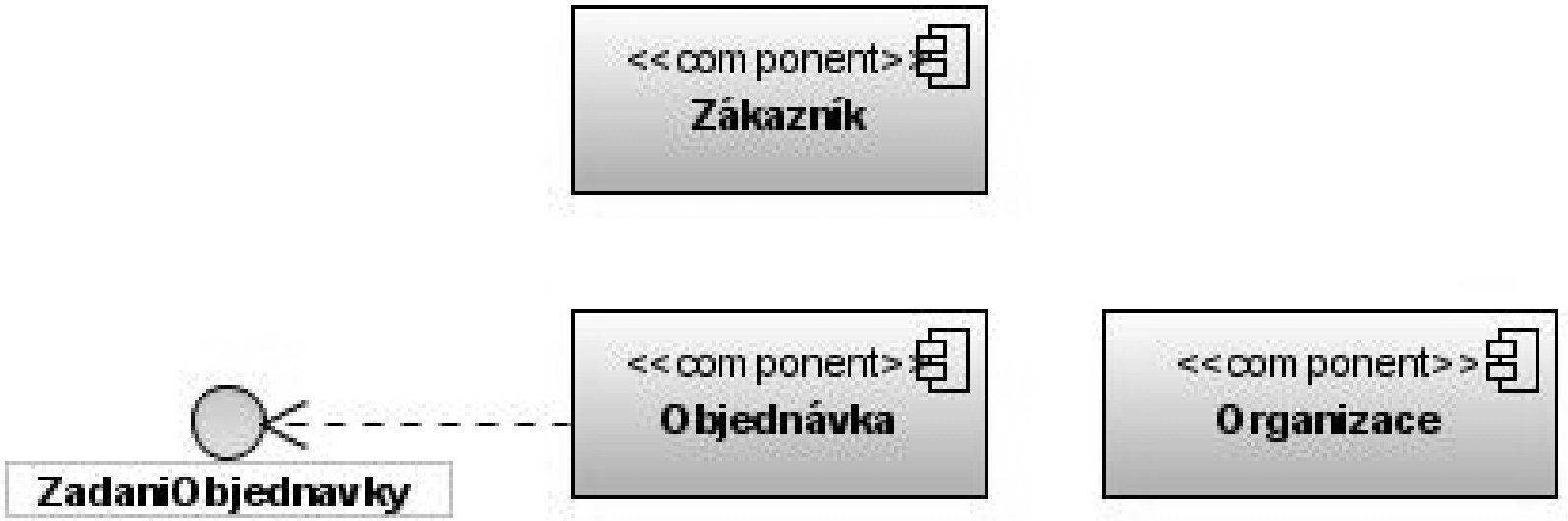

analysis Login
Basic Path
Re-use the corporate standard login screen.
Client
(from Actors)
Login
LoginAcount
Account

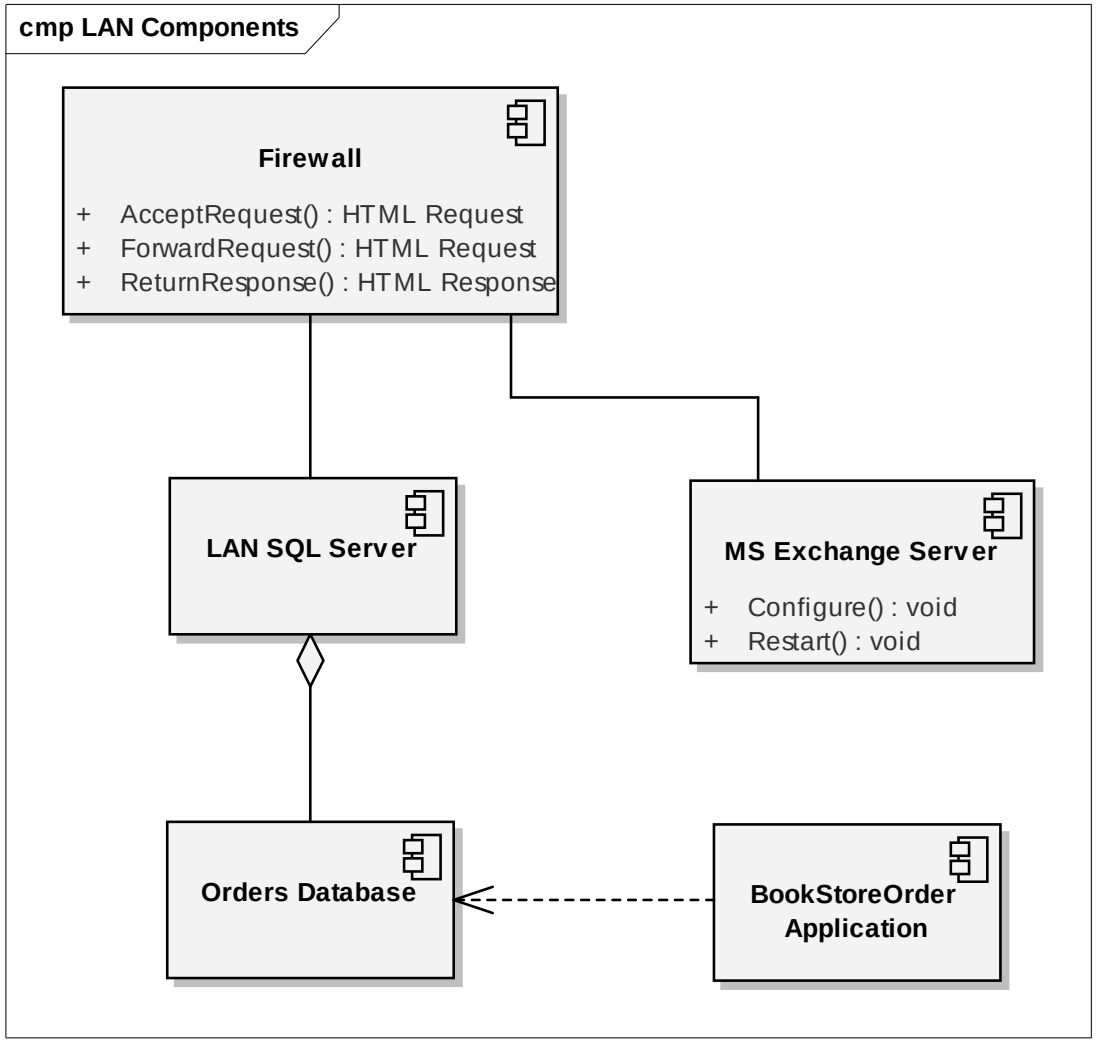
Diagramy aktivit (činností)



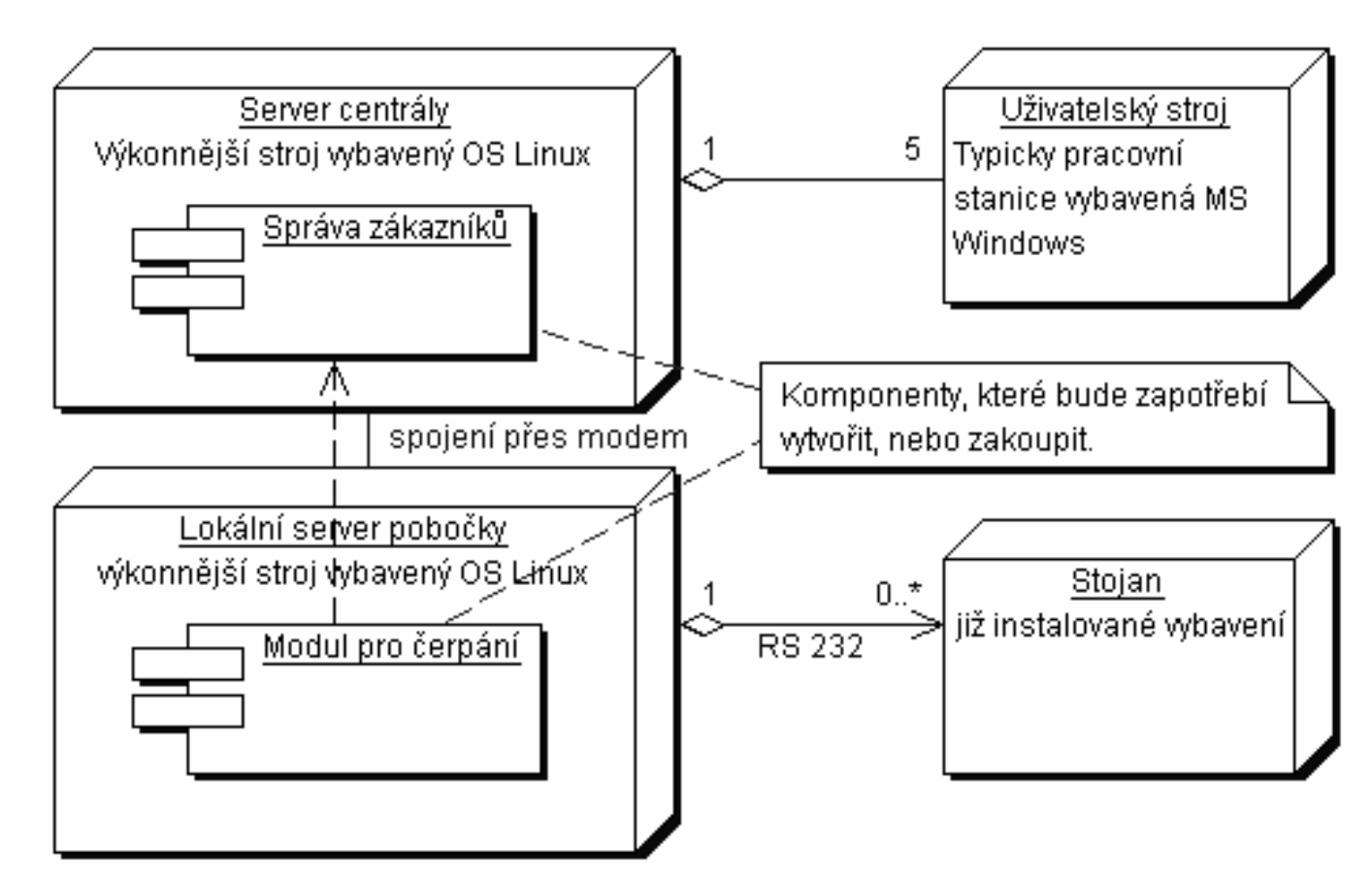

act Customer Process
Customer
Enters Web site
User
Validation
User Logs In
Rejected
View BookStore
Select Book for Purchase
Add to Shopping Basket
View Shopping Basket
Commit Order
Supply Credit Card Details
Credit Check
Credit Card Problems
Close Order
Confirm Purchase
Items Delivered
Order Complete

Diagramy komponent




cmp LAN Components
Firewall
+ AcceptRequest() : HTML Request
+ ForwardRequest() : HTML Request
+ ReturnResponse() : HTML Response
LAN SQL Server
MS Exchange Server
+ Configure() : void
+ Restart() : void
Orders Database
BookStoreOrder Application

Diagramy nasazení (deployment)



Historie UML

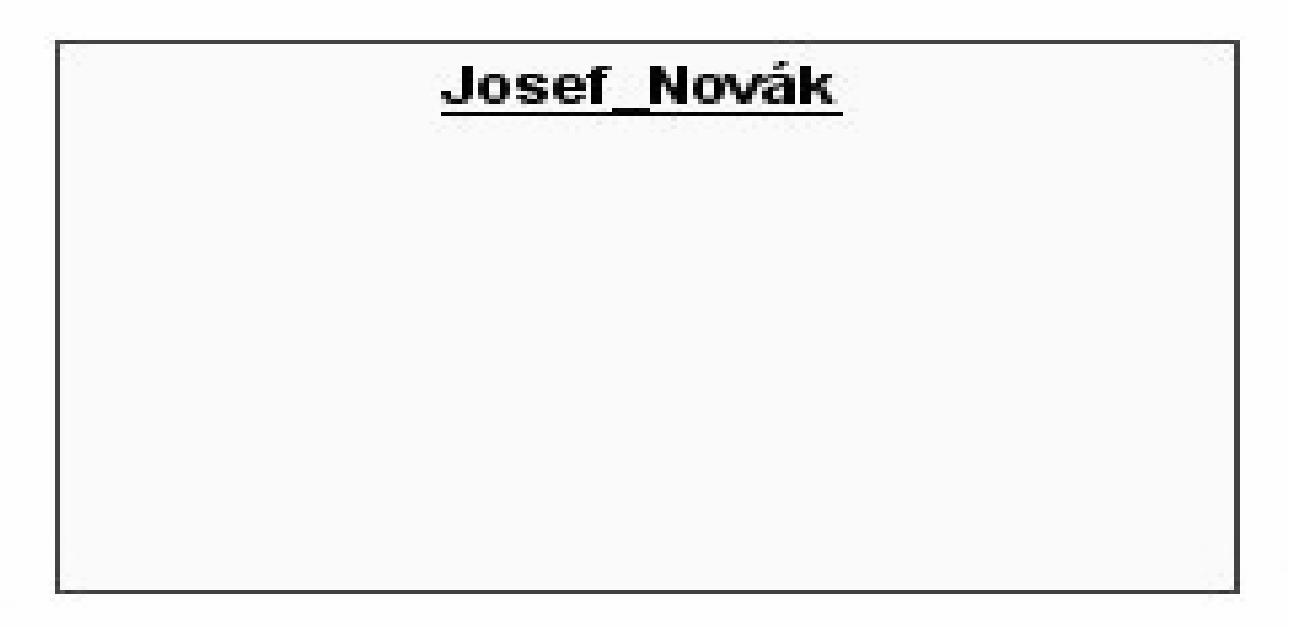
- **1994** – zahájení vývoje UML (Rumbaugh, Booch)
- **1995** – notace pro Unified Method 0.8, připojuje se Jacobson a model jednání, Rational Unified Process
- **1996** – UML 1.0 (zabudování připomínek, kooperace)
- **1997 – UML 1.1 (preciznější sémantika, přijat jako standard OMG)**
- **1999** – UML 1.3 (upřesněná verze 1.1)
- **2001** – UML 1.4 (komponenty)
- **2003 – UML 1.5 (diagramy aktivit, sémantika akcí)**
- **2005 – UML 2.0 (nové diagramy, změny)** - původně plánované ukončení květen 2002, ale ☺ (příliš mnoho hlav,obtížné)
- 2006 – UML 2.1 (další snaha o upřesnění sémantiky)

Složky UML

- Elementární prvky (infrastruktura)
- Diagramy UML (superstruktura)
- OCL (Object Constraint Language – jazyk pro popis omezení)
- Výměnný formát (buď jen model – XMI, nebo včetně diagramů konvertovaných do SVG)

Některé základní ikony

Ikona pro objekt

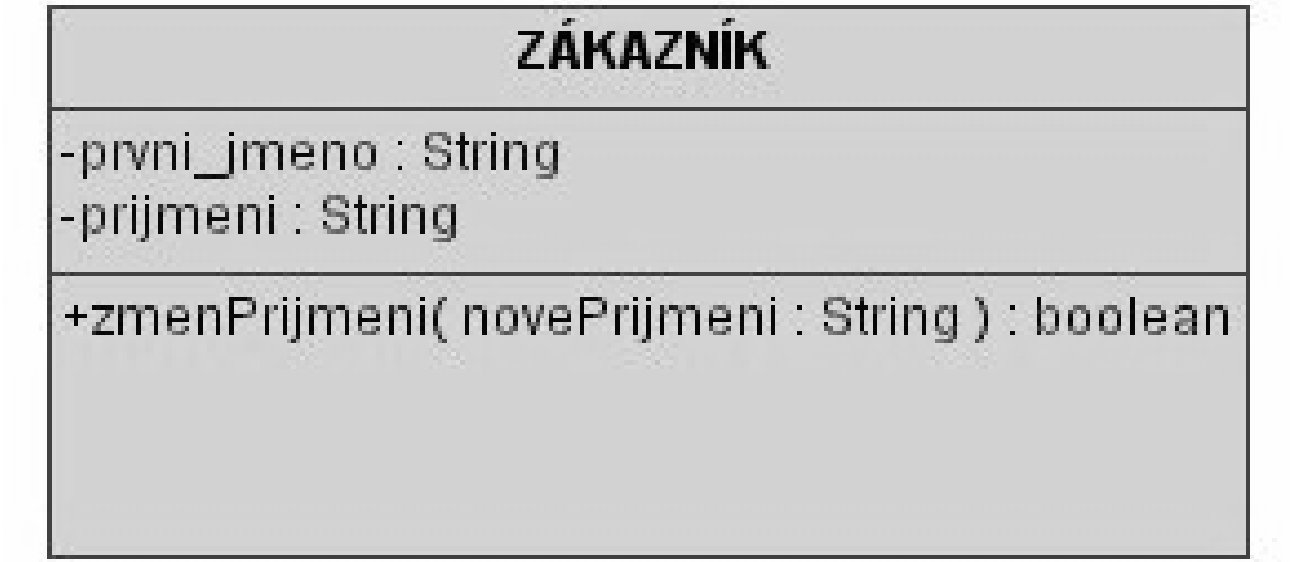


Jedinečný objekt identifikovaný svým jménem

Ikona pro třídu

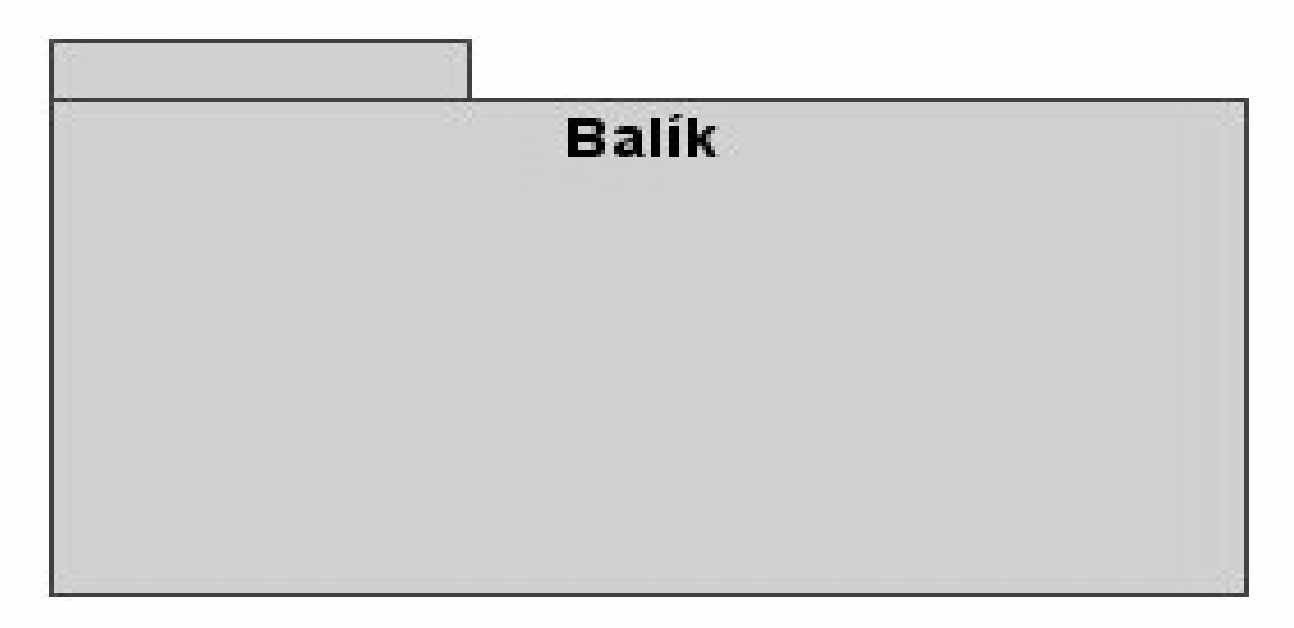
Třída zahrnuje objekty stejného typu – se stejnými atributy a operacemi

2D-symbol pro třídu



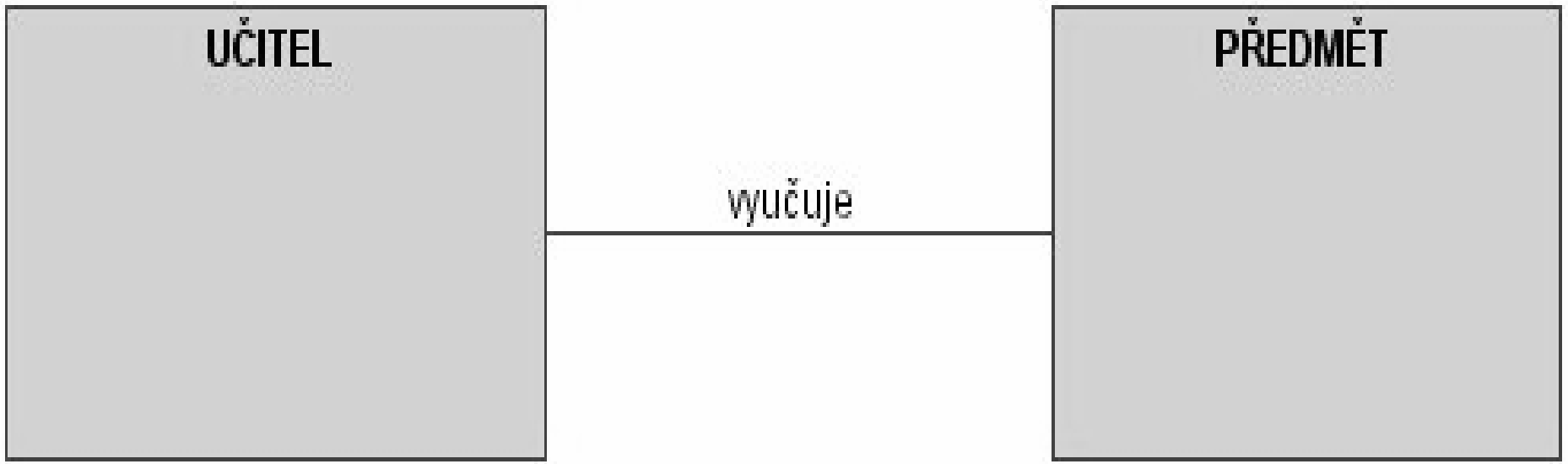
Třída s atributy a operacemi

Ikona pro balík (package)



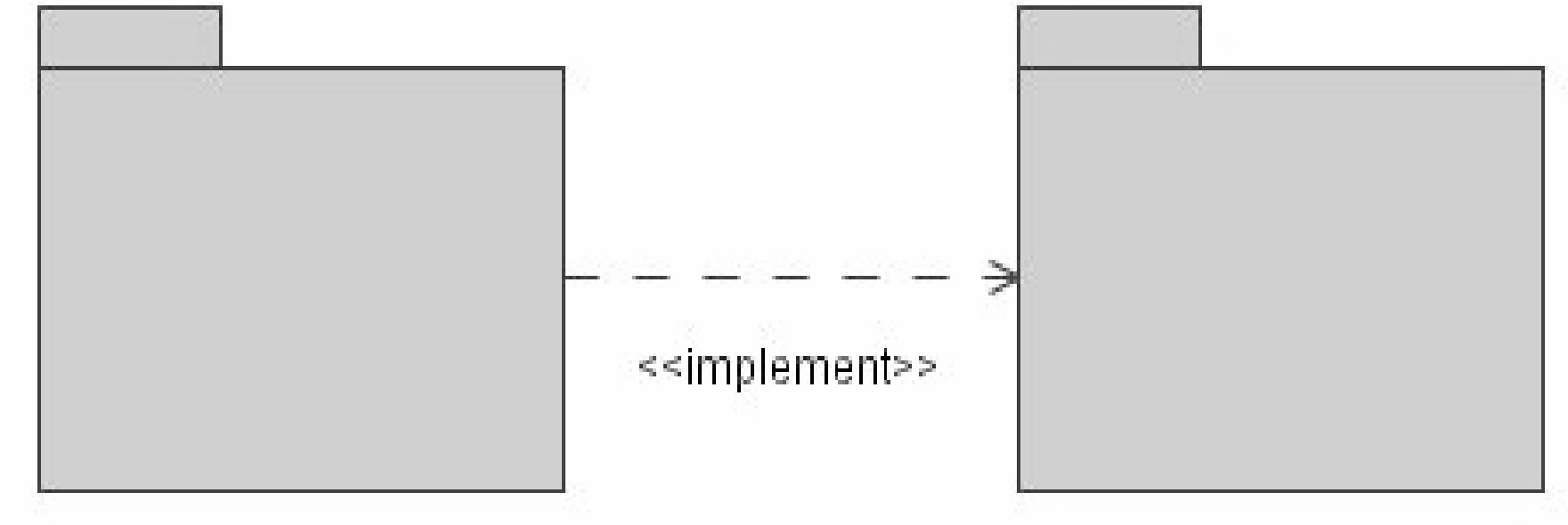
Logická skupina elementů

Spojnice



Vyjadřuje vztah (v tomto případě symetrický vztah mezi třídami)

Spojnice



Vyjadřuje závislost (v tomto případě, že něco implementuje něco jiného)

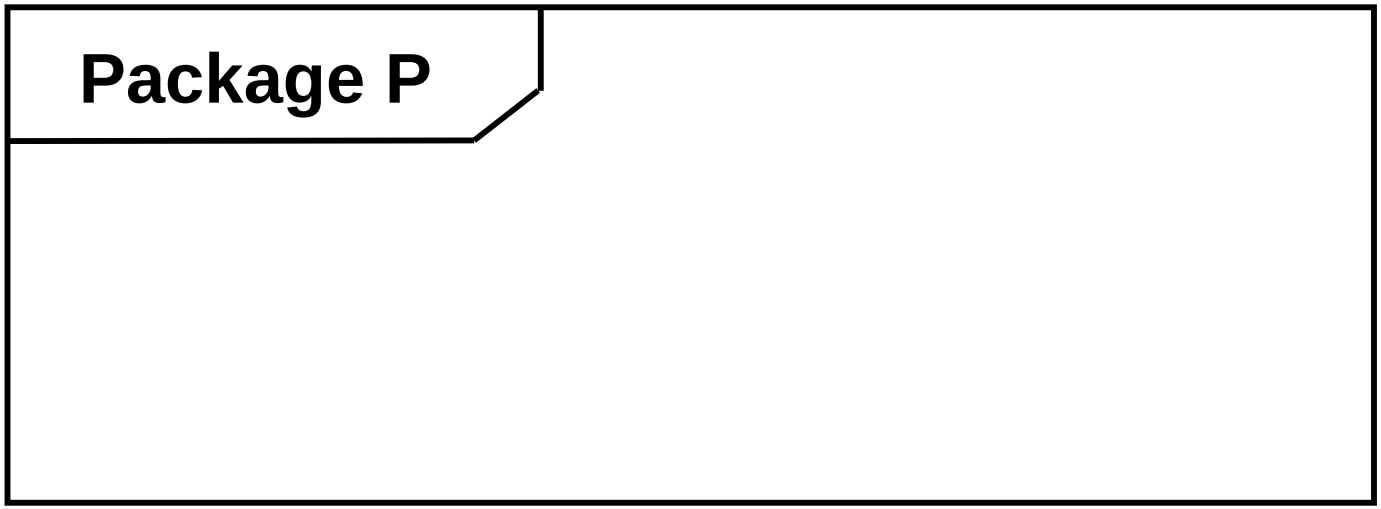
Ikona pro případ užití (use case)

Záznam o případu použití – vně systému může nastat událost, na kterou musí systém reagovat

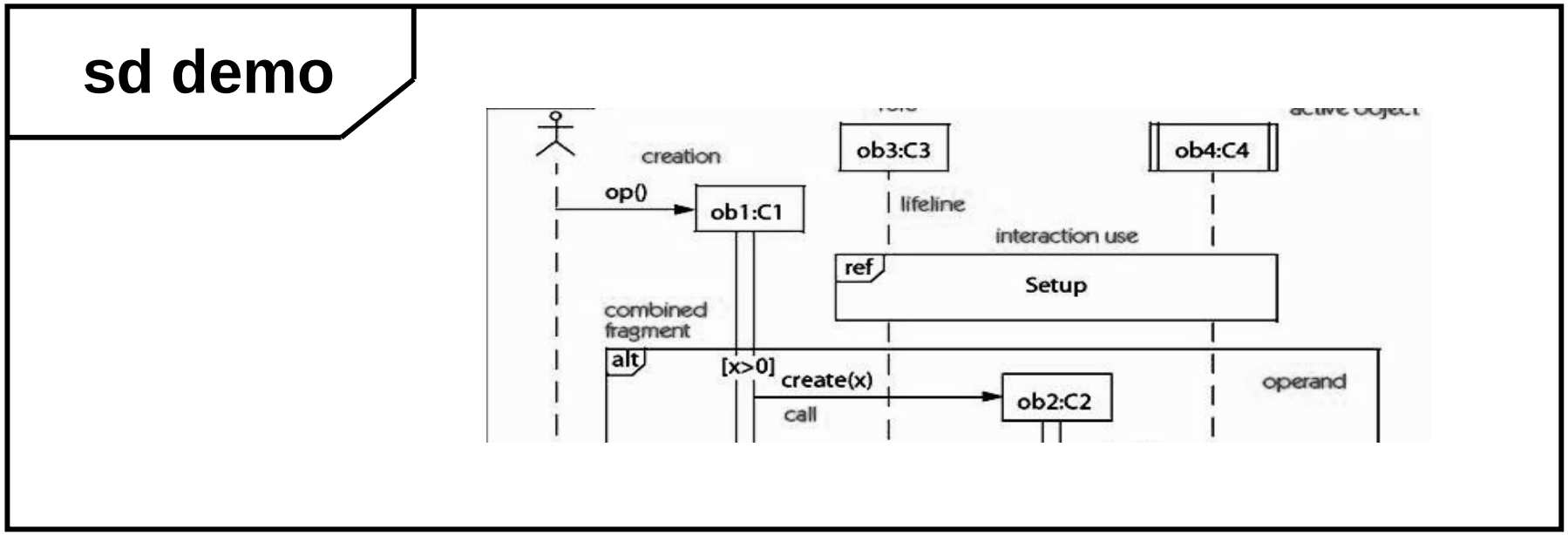
Ikona pro aktéra

Uživatelská role nebo spolupracující systém

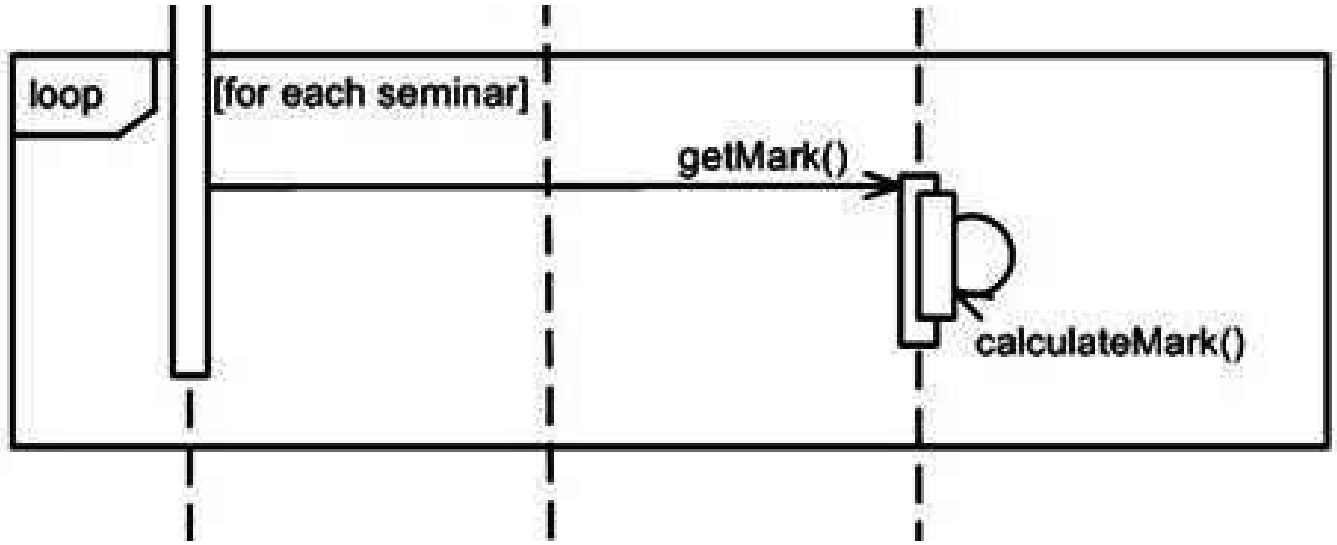
Nová ikona – rámec (frame)



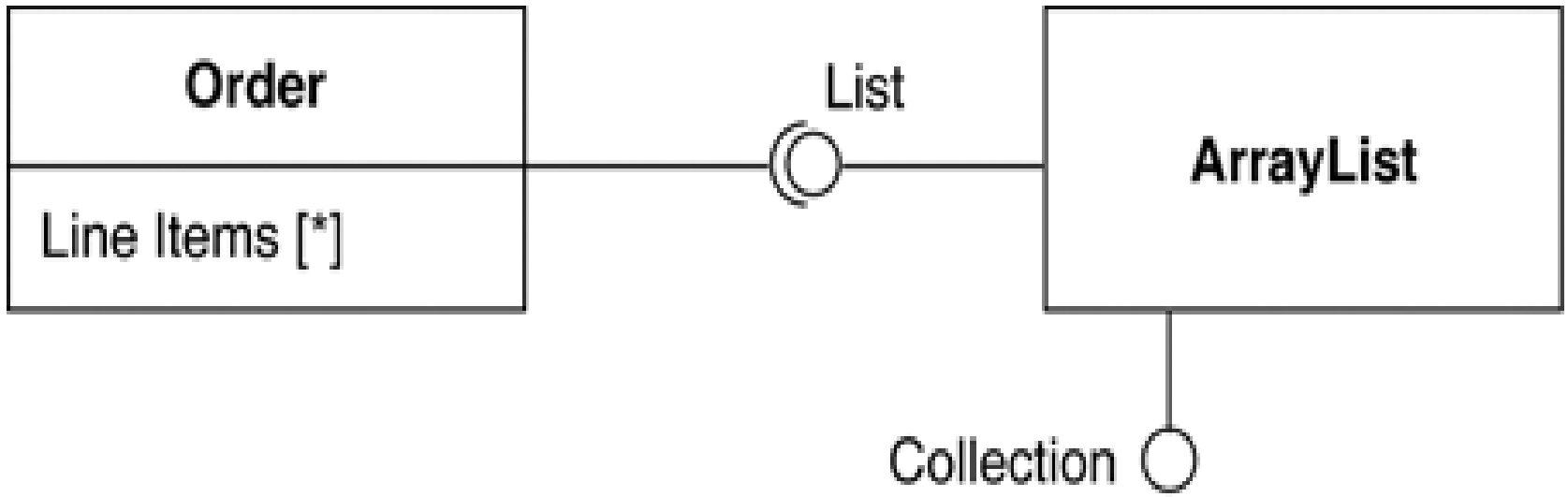
Příklad použití rámce pro diagram



Příklad použití rámce ve scénáři



Nová ikona pro „interface“



Artefakt

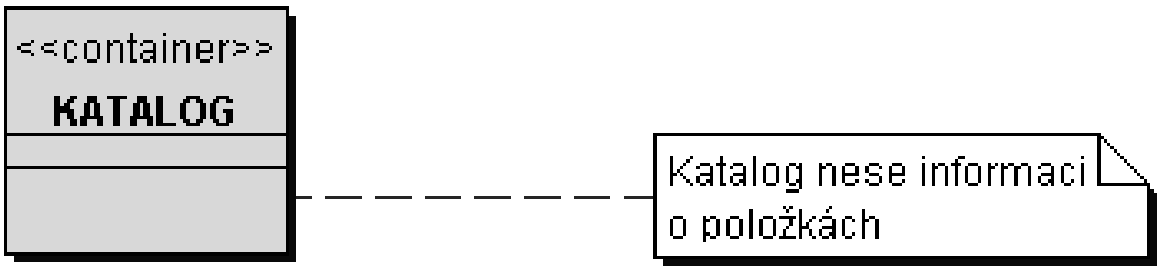
- Artefakt je specifikace fyzického kousku informace, která je použita nebo produkována při vývoji softwaru nebo při jeho nasazení a použití.
- Příkladem artefaktu může být modelový soubor, zdrojový soubor, skript, vykonatelný program, tabulka v databázi, libovolná součást dodávky, textový dokument, či zpráva.
- Instance artefaktu:

Použití artefaktu v diagramech

- Komponenta „Objednávka“ je „manifestována“ artefaktem „Objednávka.jar“

Poznámky

- Poznámka obsahuje libovolný text
- Zobrazuje se jako obdélník s „přehnutým rohem“
- Může být přidružena k elementu
- Může obsahovat stereotyp (např. **<<constraint>>** - pak se jedná o specifikaci omezení)



Struktura UML 2.0

Specifikace standardu UML 2.0 je rozdělena do 4 částí:

- **definice infrastruktury** – notace UML (syntax), definuje základní elementy, jádro UML společné pro UML a související standardy, např. MOF či CWM
- **definice superstruktury** - sémantika definovaná pomocí metamodelu, popis prvků metamodelu, konstrukty používané uživateli UML – elementy diagramů a diagramy
- **definice výměnné struktury** (diagram interchange) – pro export a import modelů a diagramů, formát pro výměnu dokumentů (diagramů) mezi různými nástroji, specifikace převodu do výměnných formátů (CORBA IDL, XML DTD)
- **Object Constraint Language** (OCL) – jazyk pro formálně přesný popis různých omezení platných v modelu

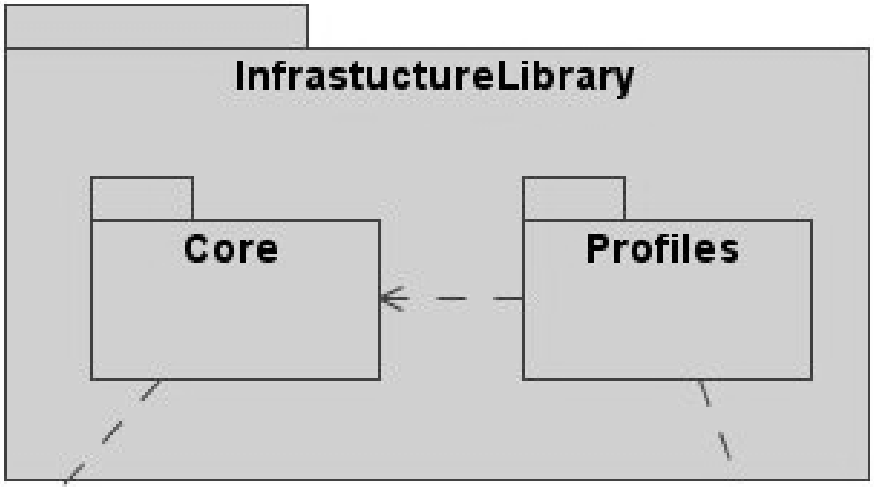
Aktuální verze UML 2.0

- **Superstructure Specification 2.0**: Revised Final Adopted Specification (formal/05-07-04) August 2005
- **Infrastructure Specification 2.0**: Revised Final Adopted Specification (formal/05-07-05) March 2006
- **Diagram Interchange Specification 2.0**: Convenience Document (ptc/05-06-04) June 2005
- **OCL 2.0**: Available Specification (formal/06-05-01) May 2006

`http://www.uml.org`

Infrastruktura UML (2.0)

Infrastruktura UML 2.0 je reprezentována balíkem InfrastructureLibrary.

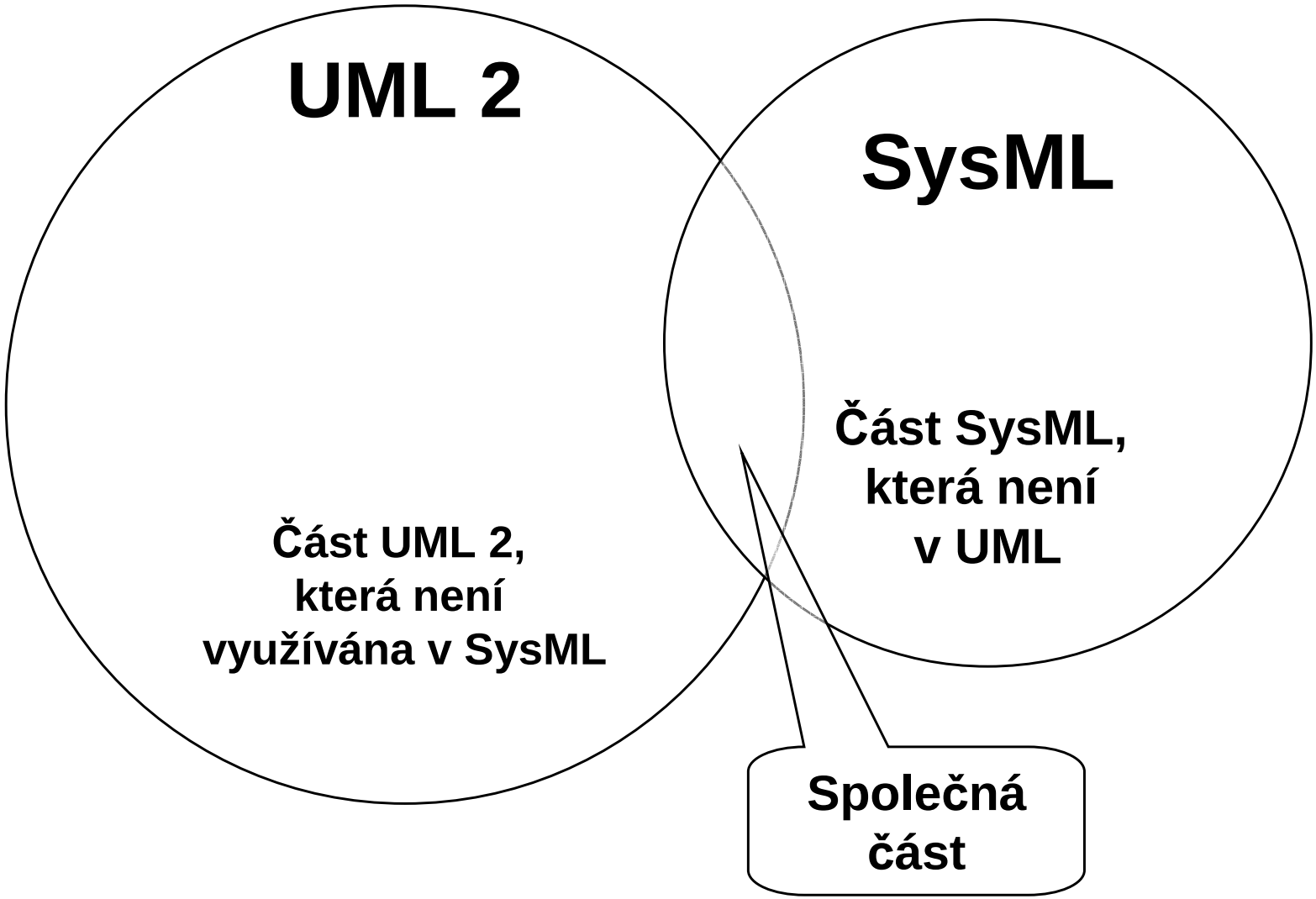


Balík Core představuje jádro definice infrastruktury. Jedná se o společný metamodel pro UML, MOF, CWM a Profily.

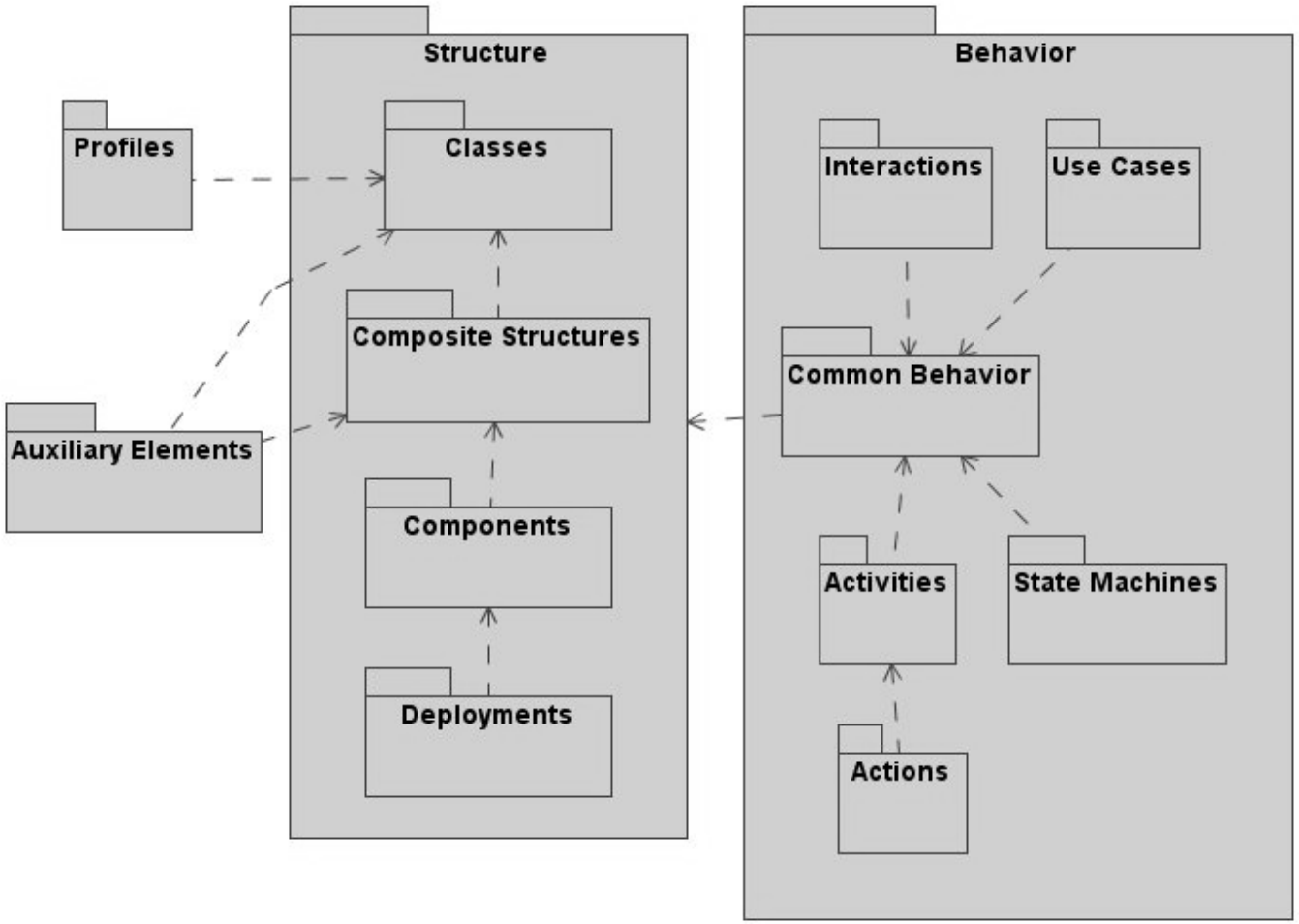
Profily slouží pro přzpůsobení jádra pro konkrétní domény (C++, EJB, softwarové modely, reálný čas, ...). Balík Profiles obsahuje mechanismy, které umožňují uzpůsobit prvky metamodelu pro konkrétní modely (z úrovně M3 na M2).

Profily v UML

Příklad: SysML je rozšíření UML, které využívá jen některé diagramy UML a přidává nové diagramy, které v UML nejsou.

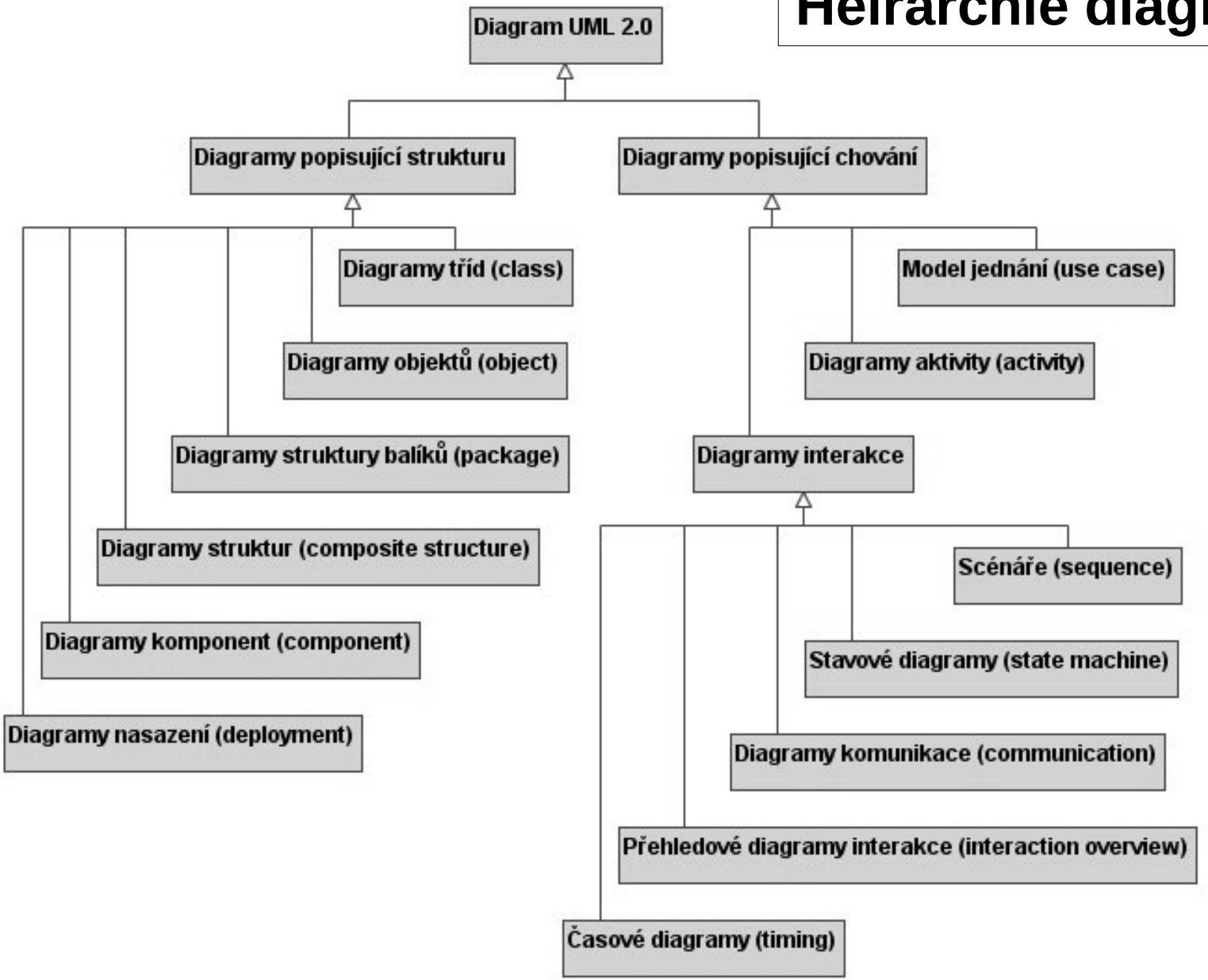

UML 2
SysML
Část SysML,
která není
v UML
Část UML 2,
která není
využívána v SysML
Společná
část

Superstruktura UML (2.0)



Diagramy verze 1.5 a 2.0

Heirarchie diagramů

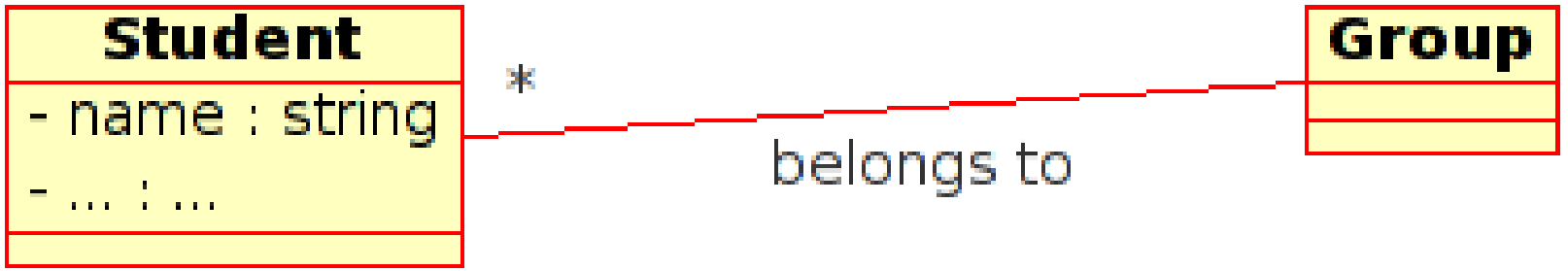


The End

Logický a fyzický model, MDA

Michal Píše
pisem1@fel.cvut.cz

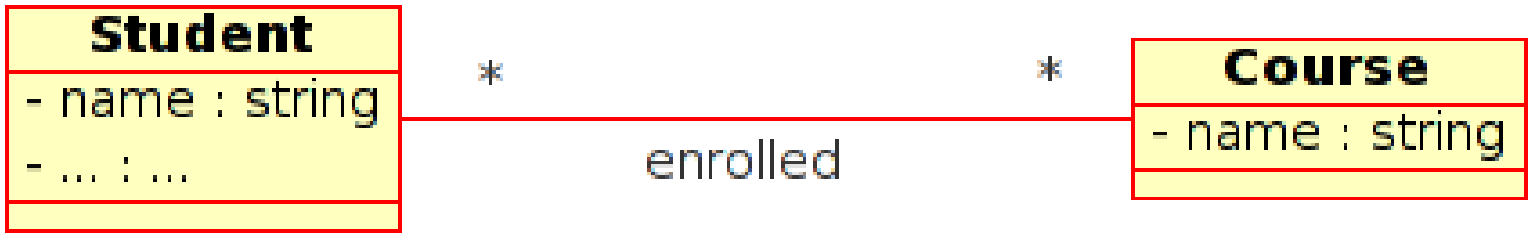
Jednoduchý model



Jednoduchý model - DB

- Tabulky student a group
- Každá tabulka má sloupce odpovídající jednotlivým atributům
- Tabulka student má navíc cizí klíč do tabulky group

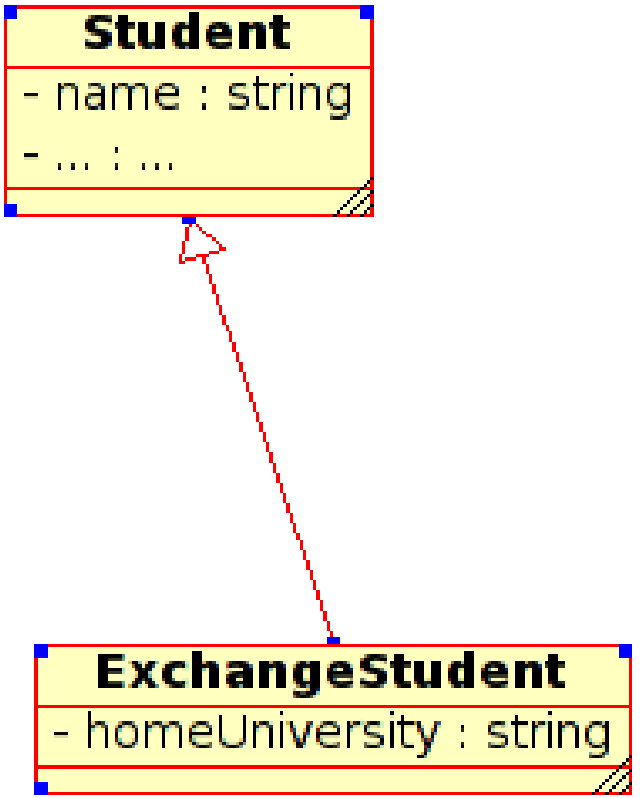
Lehce složitější model



Lehce složitější model - DB

- **Tři** tabulky: student, course a student_enrolled_course
- Tabulka student_enrolled_course na modelu není explicitně vidět, ale z multiplicity asociace enrolled se její existence dá odvodit
- Model stále obsahuje plnou informaci o jeho fyzické realizaci

Složitější model



Složitější model - DB

?

První možnost

- Dvě tabulky: student a foreign_student
- Každá tabulka má sloupce odpovídající atributům dané třídy
  - student má sloupec name
  - foreign_student má sloupce name a home_university
- Při hledání studenta (nebo studentů) se musíme podívat do obou tabulek

Druhá možnost

- Dvě tabulky: student a foreign_student
- Každá tabulka má sloupce odpovídající atributům přidaným v dané třídě
  - student má sloupec name
  - foreign_student má sloupec home_university
- Jedna instance třídy ForeignStudent je rozprostřená přes víc tabulek, takže při přístupu na přidané atributy musíme používat kartézský součin (join)

Další možnosti

- Různé kombinace předchozích řešení
- Zcela odlišné koncepce
- Zabývají se tím tvůrci objektově-orientovaných databází

Důsledek

- Původní model nenese celou informaci o implementaci
- Pokud do původního modelu informaci o implementaci zaneseme, model se tím znepřehlední
- V ideálním případě by model měl být přehledný a zároveň obsahovat plnou informaci
- Co s tím?

Řešení

- Dva modely, logický a fyzický
- Logický model je přehledný
- Fyzický model je odvozením (zpřesněním) fyzického a obsahuje plnou informaci
- Oba modely musí být synchronizovány!

Terminologie MDA

- Logický model se jmenuje PIM (Platform Independent Model)
- Fyzický model se jmenuje PSM (Platform Specific Model)
- Zpřesňování implementace může probíhat po krocích, takže fyzických modelů může být víc

Příklad zpřesňování

- Na úrovni PIM mám jenom třídy a asociace
- První zpřesnění: budu to psát v Javě
  - Takže vím, že mám třídy a interfacy
- Druhé zpřesnění: budu používat EJB
  - Entity Beans, Session Beans,...
- Třetí zpřesnění: budu používat Oracle a WebSphere
- ...

Výhody

- Různé úrovně abstrakce modelu
  - Analytik chce jinou úroveň abstrakce než programátor
- Zachování podstatné části investic
  - Při přechodu na jinou technologii se zahodí jen ty modely, které na dané technologii závisí
  - **Analýza se nemusí dělat znova!**

Výhody II

- Korespondence modelů
  - Specifičtější model může být poloautomaticky odvozen z toho abstraktnějšího
- Flexibilita
  - Do procesu odvození se dá kdykoli zasáhnout

Výhody III

- Generování kódu
  - Nejspecifičtější model nese tolik informace, že podstatná část kódu aplikace se z něj dá vygenerovat
- Snadná změna architektury aplikace
  - Změnou formy generovaného kódu změním architekturu celé aplikace

Pozor!

- Vývoj musí být “model-driven” tzn. mění se nejprve model, teprve potom se změny promítají do kódu
- Vývoj nemusí nutně probíhat vodopádově – i modely se dají vyrábět iterativně
- Diagram tříd popisuje jenom statickou strukturu aplikace, vývoj řízený modelem se týká i ostatních diagramů

Problémy

- Aplikace se neskládá jenom z “databázových” tříd, potřebovali bychom i třídy “paměťové” - jak je v modelu rozlišit?
  - Poznámkou?
- Java nemá jenom třídy, má i interfacy – jak je v modelu rozlišit?
  - Zase poznámkou?

Problémy II

- Java “umí” jednoduchou dědičnost tříd a vícenásobnou dědičnost interfaců – jak to v modelu ohlídat?
  - Dávat si na to pozor?
- Java “neumí” násobné atributy
  - Stačí je v UML nepoužívat?
- V databázi nemá smysl mluvit o modifikátorech přístupu – sloupce nejsou private, protected nebo public – co s tím?
  - Ignorovat?

Problémy III

- U “databázových” tříd bych potřeboval evidovat dvě jména – jméno v kódu a jméno v databázi – jak to do modelu dopsat?
  - Definovat mechanismus odvození jednoho jména z druhého?
- U tříd UI bych potřeboval definovat např. události, které v dané třídě mohou vznikat a kam se pošlou, jak?
  - V poznámce?

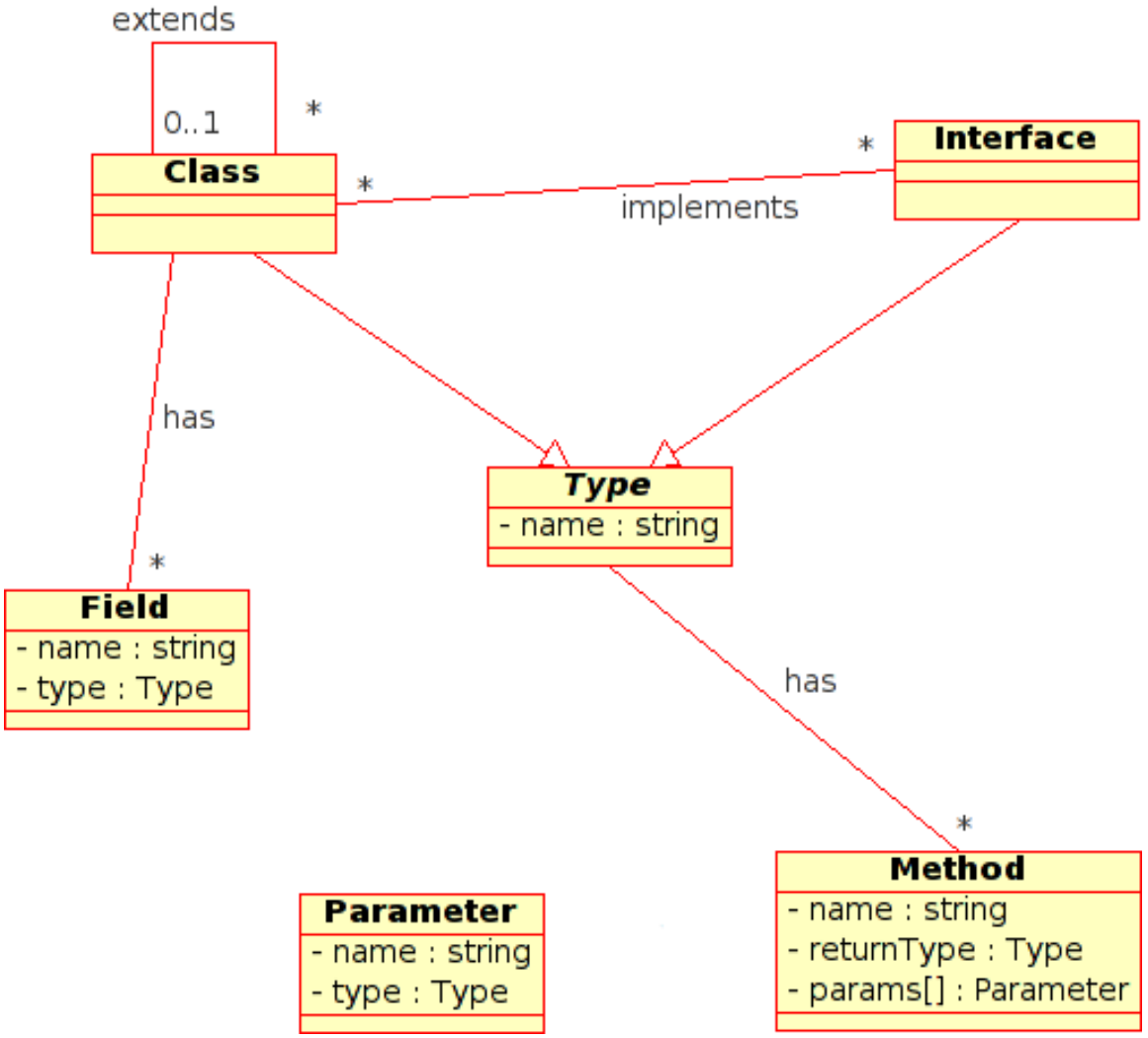
Problémy obecně

- UML někdy umí víc, než potřebujeme
- UML někdy umí méně, než potřebujeme
- UML by někdy mělo kontrolovat nějaká omezení

Řešení

- Popíšeme strukturu pojmů, se kterými pracujeme stejně, jako popisujeme strukturu objektů v aplikaci
  - Popíšeme, co je to v Javě třída, interface, atribut,...
  - Popíšeme, co je to tabulka, sloupec,...

Java


extends
0..1
*
Class
*
*
Interface
implements
has
*
Field
- name : string
- type : Type
Type
- name : string
has
*
Method
- name : string
- returnType : Type
- params[] : Parameter
Parameter
- name : string
- type : Type

Databáze

Zkuste si doma

Vrstvy návrhu

- Data – v návrhu se neobjevují
- Metadata = model – to, co se vytváří během analýzy a návrhu
- Metametadata = metamodel – definice pojmů, které se používají v modelu
- Metametamodel = definice pojmů, které se používají při definici pojmů, které se používají v modelu
- …

Jak to vyřeší naše problémy?

- Každý model bude navrhován v nějakém metamodelu
  - PIM v obecném
  - PSM např. v metamodelu Javy a J2EE
- UML nástroje se před tvorbou modelu zeptají na metamodel a nabídnou nám jenom to, co metamodel definuje
- MDA nástroje nám umožní popsat způsob transformace modelů

Terminologie MDA

- MOF – Metaobject Facility
  - Standard, který definuje metametamodel, tzn. pomocí kterého se vytvářejí metamodely
- UML Profile for Java
  - Zjednodušeně řečeno metamodel Javy

Terminologie MDA II

- XMI – XML Metadata Interchange
  - Formát založený na XML, ve kterém se ukládají modely a metamodely
- QVT – Queries/Views/Transformations
  - Jazyk umožňující popsat transformace modelů

Dotazy

?

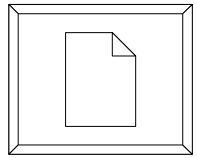
Analýza (pokračování)

Návaznosti

- Minule:
  - Architektura, modelem řízený vývoj
- Dnes:
  - Analýza (pokračování)
- Příště:
  - Návrh

Aktuální stav

- Všechny týmy by měly mít vypracován model jednání (příp. i kontext), detailní harmonogram práce s požadovanými zdroji a rozpočet – pozor harmonogramy jsou 2 – pro Vás a pro Vaše následníky.
- To vše by se mělo zobrazit na stránkách projektů a platí:
  - **Co není na service.felk.cvut.cz neexistuje!**
  - **Co není včas, neboduje se!**



Pár poznámek:

- Datový slovník popisuje pouze pojmy ze specifikované oblasti, nikoli např. co to je „CASE“.
- Datový slovník popisuje především strukturu pojmů, nikoli jejich význam – něco lze k termínům připojit formou poznámky.
- Př.

  ZÁKAZNÍK = @ID+příjmení+jméno+…

  evidovaný zákazník

Model jednání a kontext

- Model jednání (use case model) slouží pro evidenci aktérů a služeb systému.
- Kontextový diagram slouží pro evidenci aktérů a datových toků.
- Oba modely se tedy doplňují, ale představují pouze prvý krok popisu, který musí být doplněn podrobnějším popisem služeb a dat.
- Každý případ použití zastupuje sadu **aktivit**, které aktér se systémem provádí – sadu **scénářů**, jak komunikace se systémem probíhá.

Analýza

Měla by odpovědět na otázku ***CO***?

- Musí proto definovat ***konceptuální model*** řešeného systému (***PIM – Platform Independent Model***)
- Musí definovat představu, s jakými ***daty*** bude systém pracovat, jaké ***služby*** bude systém poskytovat a jak se bude chování systému měnit - jaká bude ***dynamika*** systému
- Musí stanovit podmínky, za jakých je analytická dokumentace ***akceptovatelná***

Funkční model

Funkčně orientovaná analýza

- Začínáme seznamem funkcí - modelem jednání
- Pro každý případ užití navrhneme scénář jednání (původce, událost, akce, participanti, výstupy - reakce), diagramy aktivit, příp. diagramy datových toků (návaznosti funkcí)
- Popis akcí (minispecifikace základních akcí)
- Identifikace objektů
- Identifikace vztahů mezi objekty
- Modelování životních cyklů objektů

Model jednání ECO-skladu

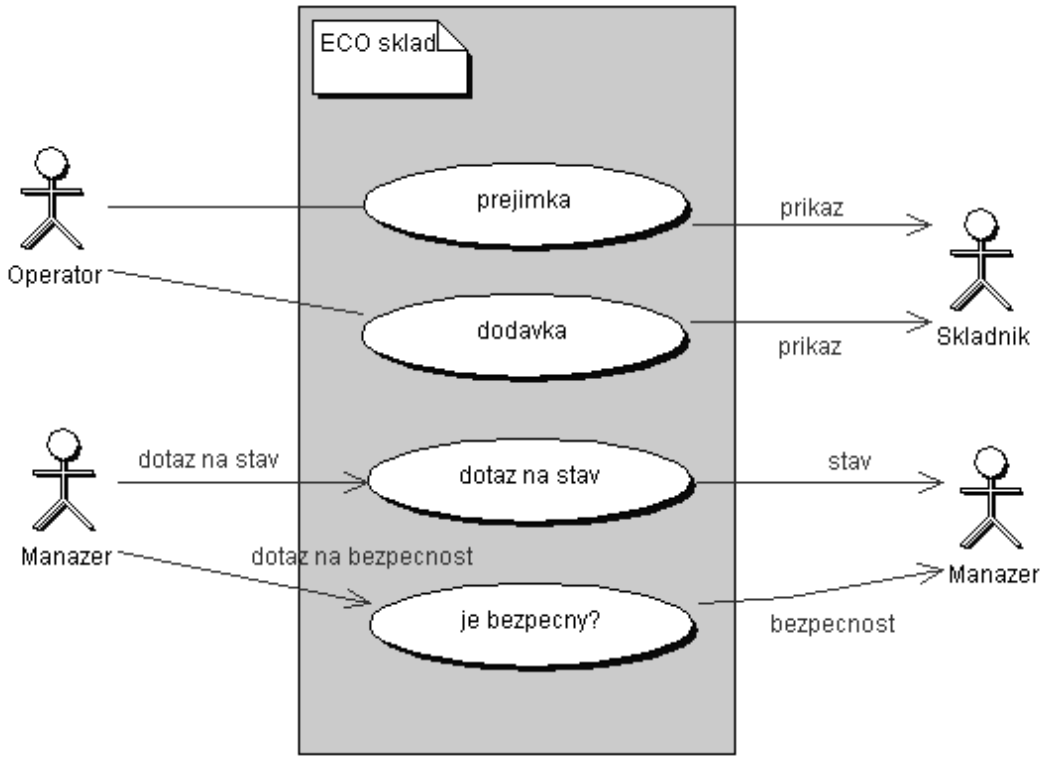
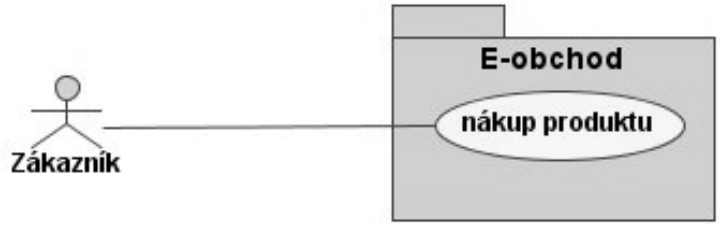


Diagram případů užití je pouhá evidence služeb, ty musí být popsány přesněji. Striktně řečeno - model jednání obsahuje diagram případů užití a jejich popis.

Jak lze služby evidované v modelu jednání popsat?

- Textovým popisem (to je podmínka nutná, nikoli postačující).
- Minispecifikací (strukturovaným popisem operace)
- Dekompozicí na služby jednodušší
  - pomocí scénáře
  - pomocí diagramu datových toků (DFD)
  - pomocí diagramu aktivity
  - pomocí diagramu komunikace
  - pomocí stavového diagramu



1. Zákazník prohlíží katalog a vybere si zboží k nákupu
2. Zákazník zvolí nákup
3. Zákazník vyplní dodací informace (adresa, expresní nebo standardní dodávka)
4. Systém zobrazí plnou cenu včetně ceny dodání
5. Zákazník vyplní platební informace (číslo kreditní karty)
6. Systém autorizuje platbu
7. Systém potvrdí prodej
8. Systém zašle potvrzovací e-mail zákazníkovi

Alternativy:

3a. Uživatel je pravidelným zákazníkem

- 3a1. Systém zobrazí naposled zapamatované dodací a platební informace
- 3a2. Uživatel může potvrdit, nebo změnit zobrazené informace a scénář pokračuje v kroku 6

6a. Systému se nepovedlo autorizovat platbu

- 6a1. Zákazník může opravit platební informace, nebo zrušit nákup

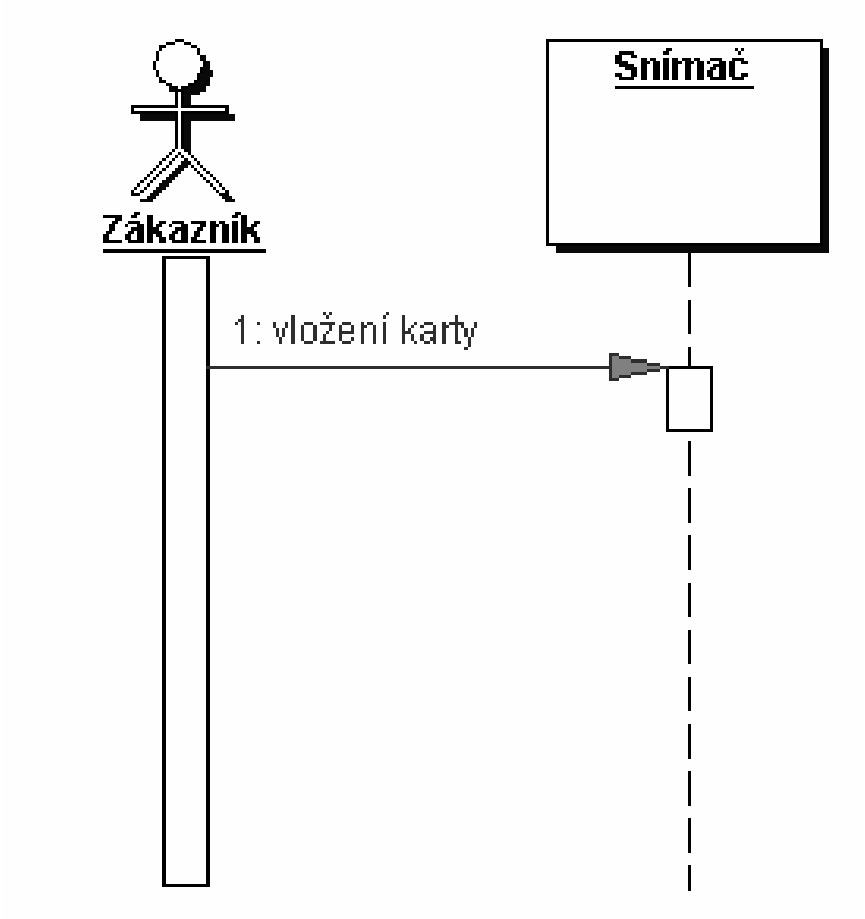
Scénáře (Sequence diagrams)

(zachycení sledu událostí)

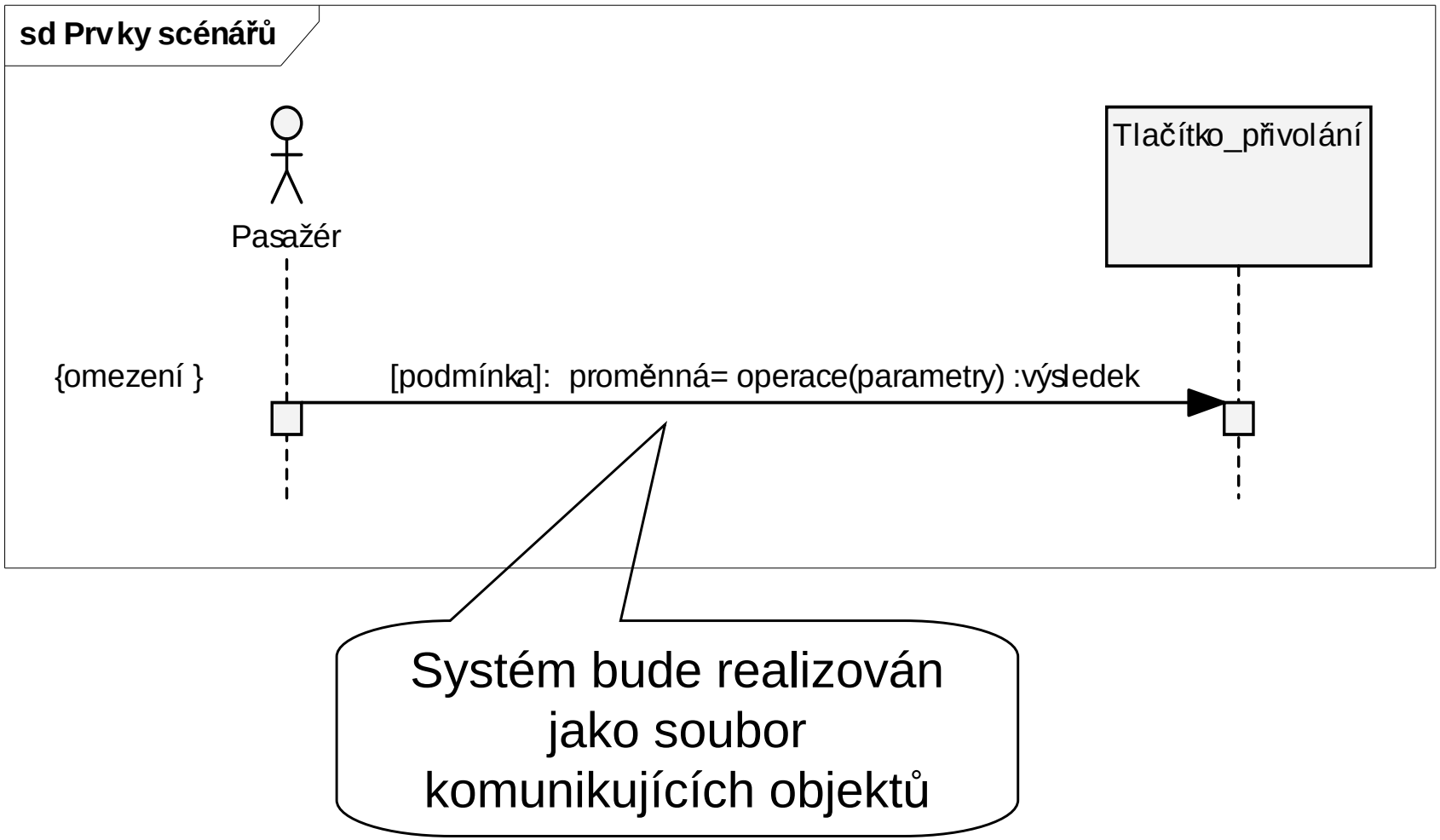
Prvky:

- **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

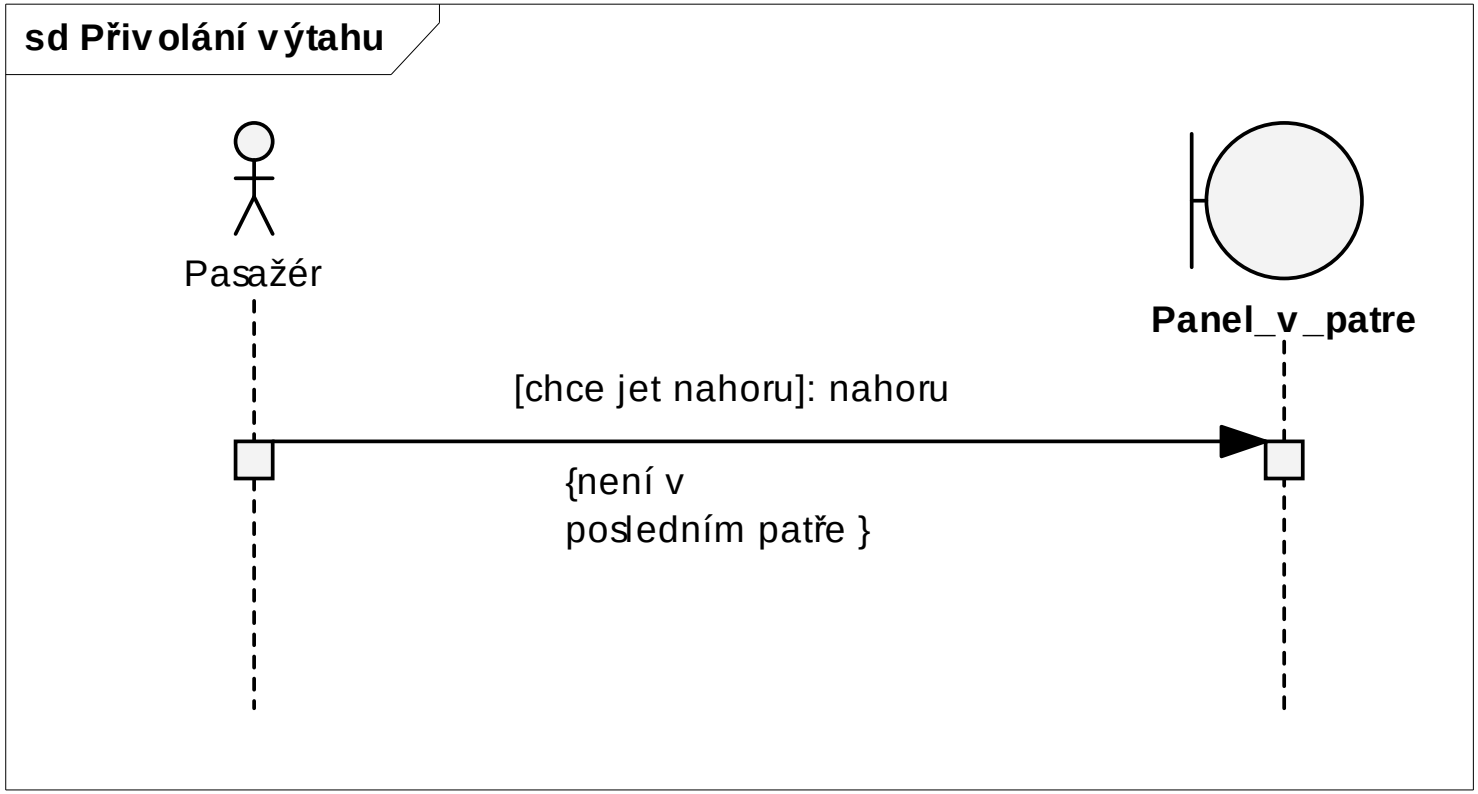
Zákazník se „autentizuje“



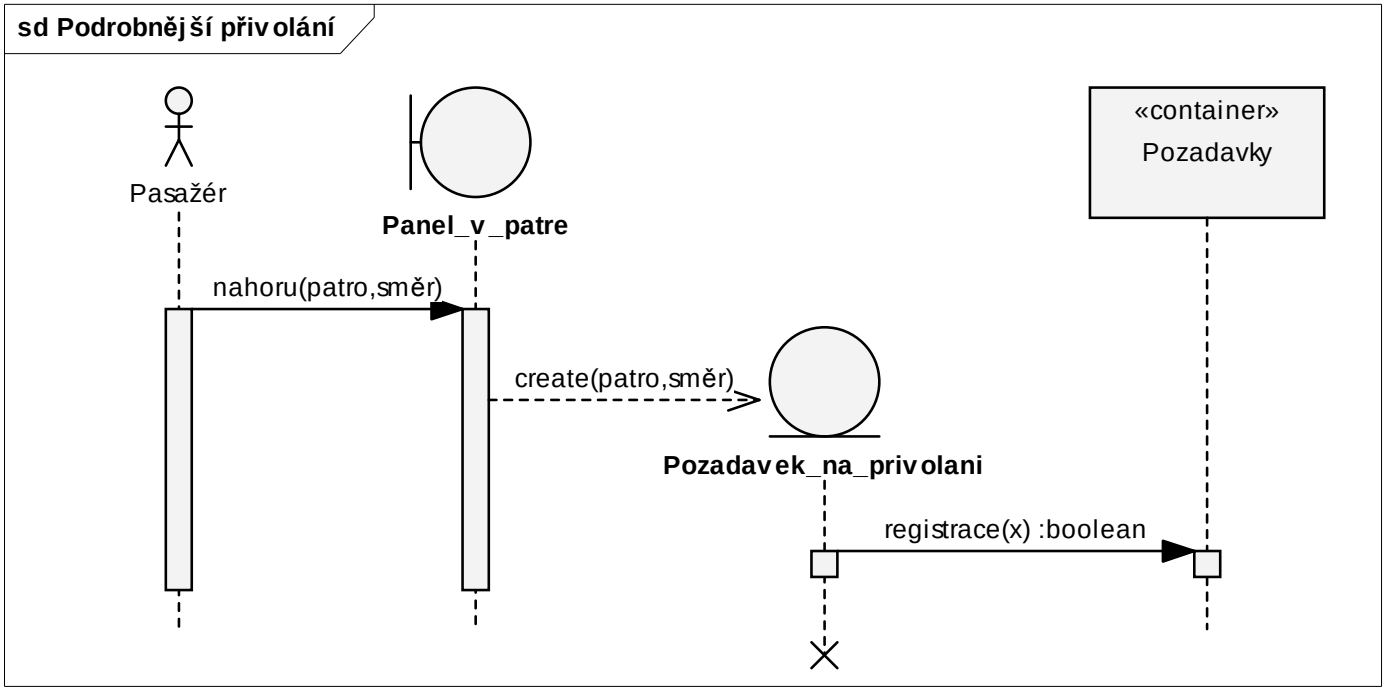
Základní princip scénáře



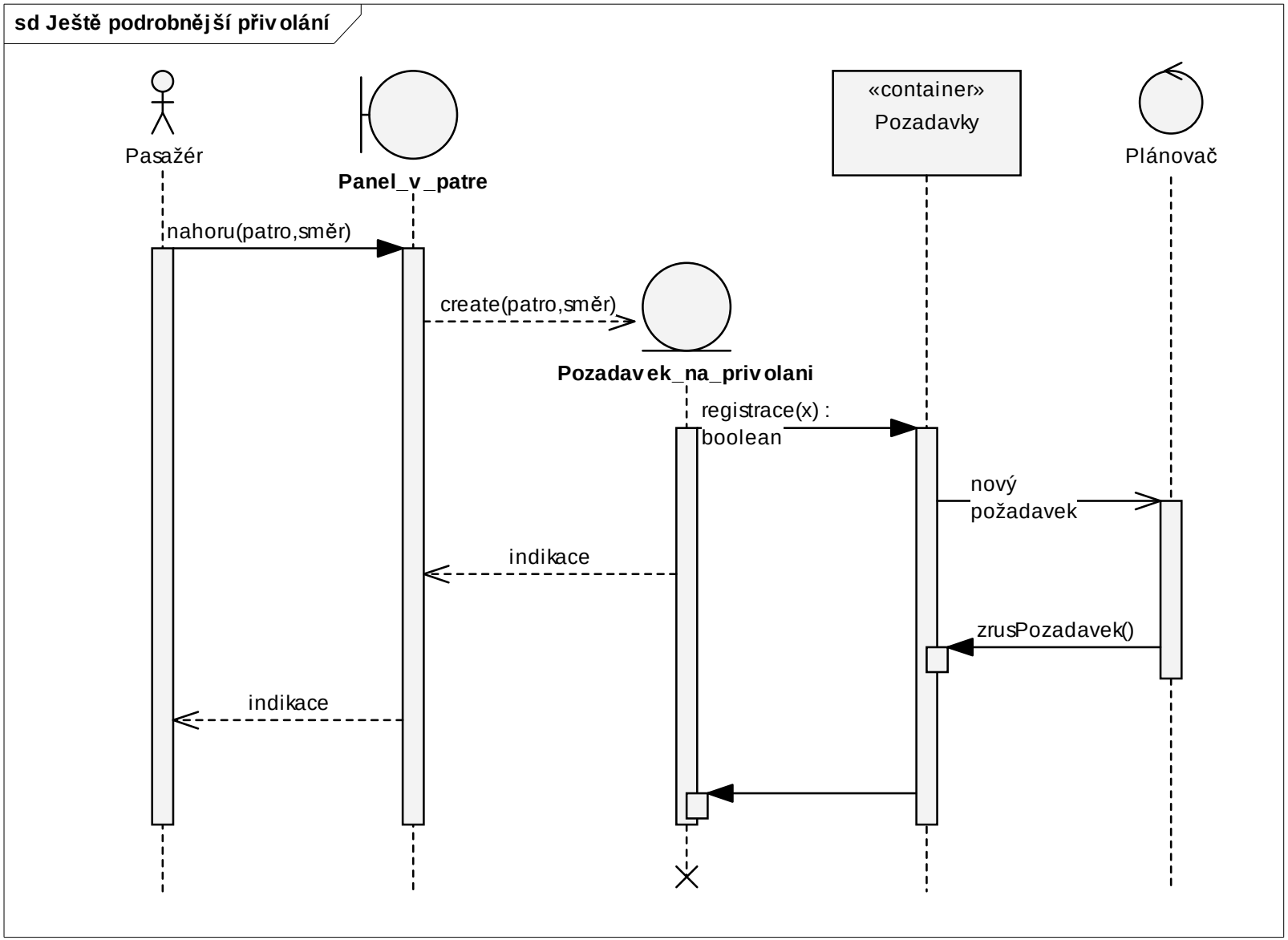
V konceptu nejprve pouze volíme metodu



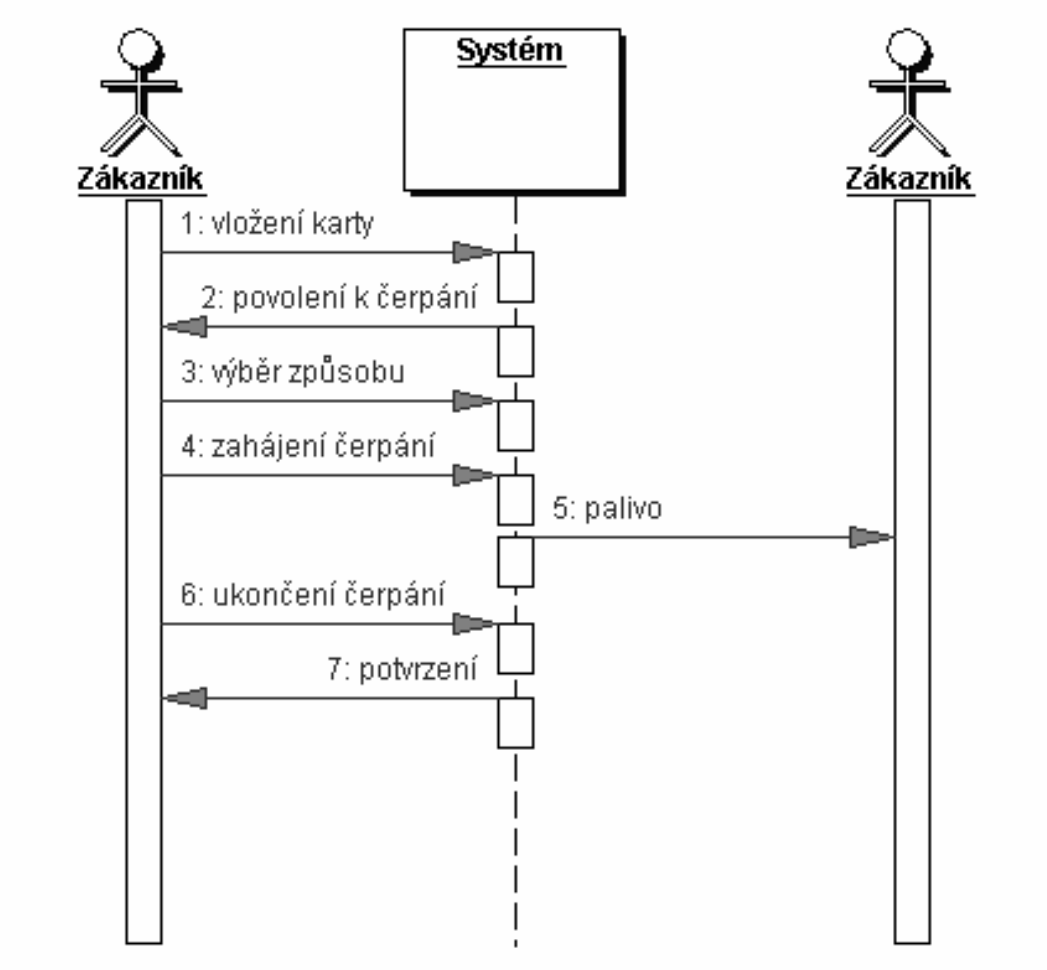
Později můžeme popsat činnost podrobněji, využít konstrukce a destrukce



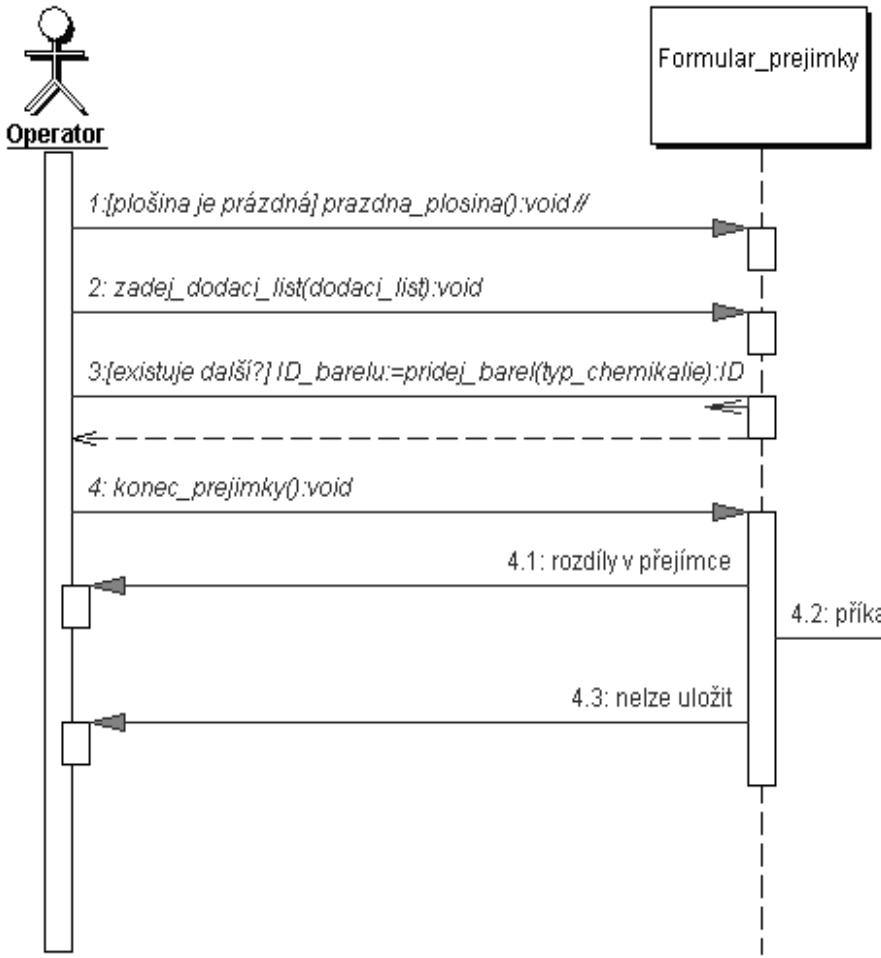
Reakce a návratové hodnoty



Hrubý scénář pro „čerpání“



Scénář pro “přejímku”



Scénář je pouze dekompozice, jeho složky musí být popsány přesněji – scénář, dekompozice, minispecifikace

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 “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: informuje systém, že již byly vyloženy všechny barely

Reads: plošina, zadaný_dodací_list

Changes: budovy ve skladu

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

Assumes:

- **sklad** je bezpečný

Popis pro “konec přejímky” (pokr.)

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

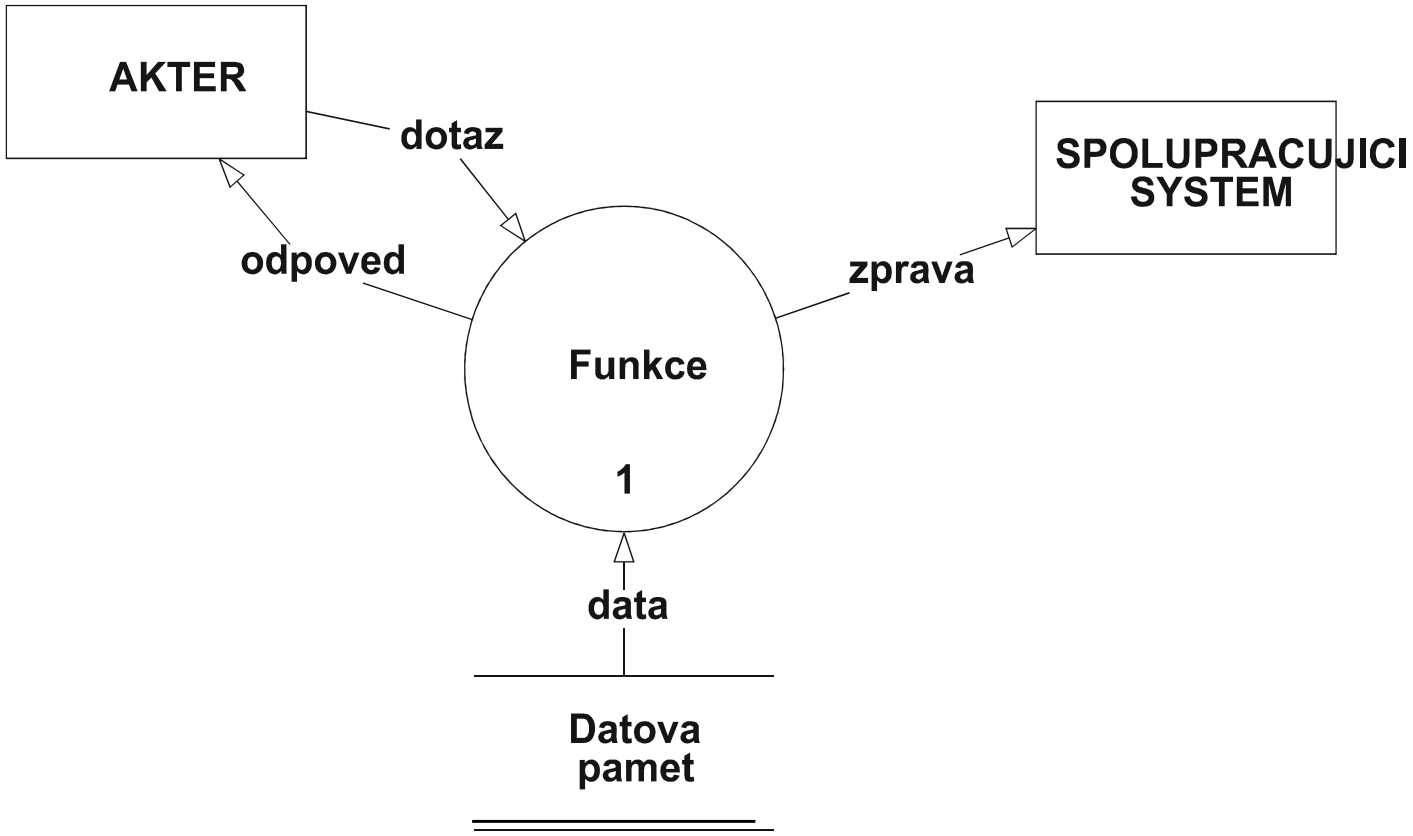
Diagramy datových toků (DFD – Data Flow Diagrams)

(zachycení vazeb funkcí a toků dat, dokumentace dekompozice)

Komponenty:

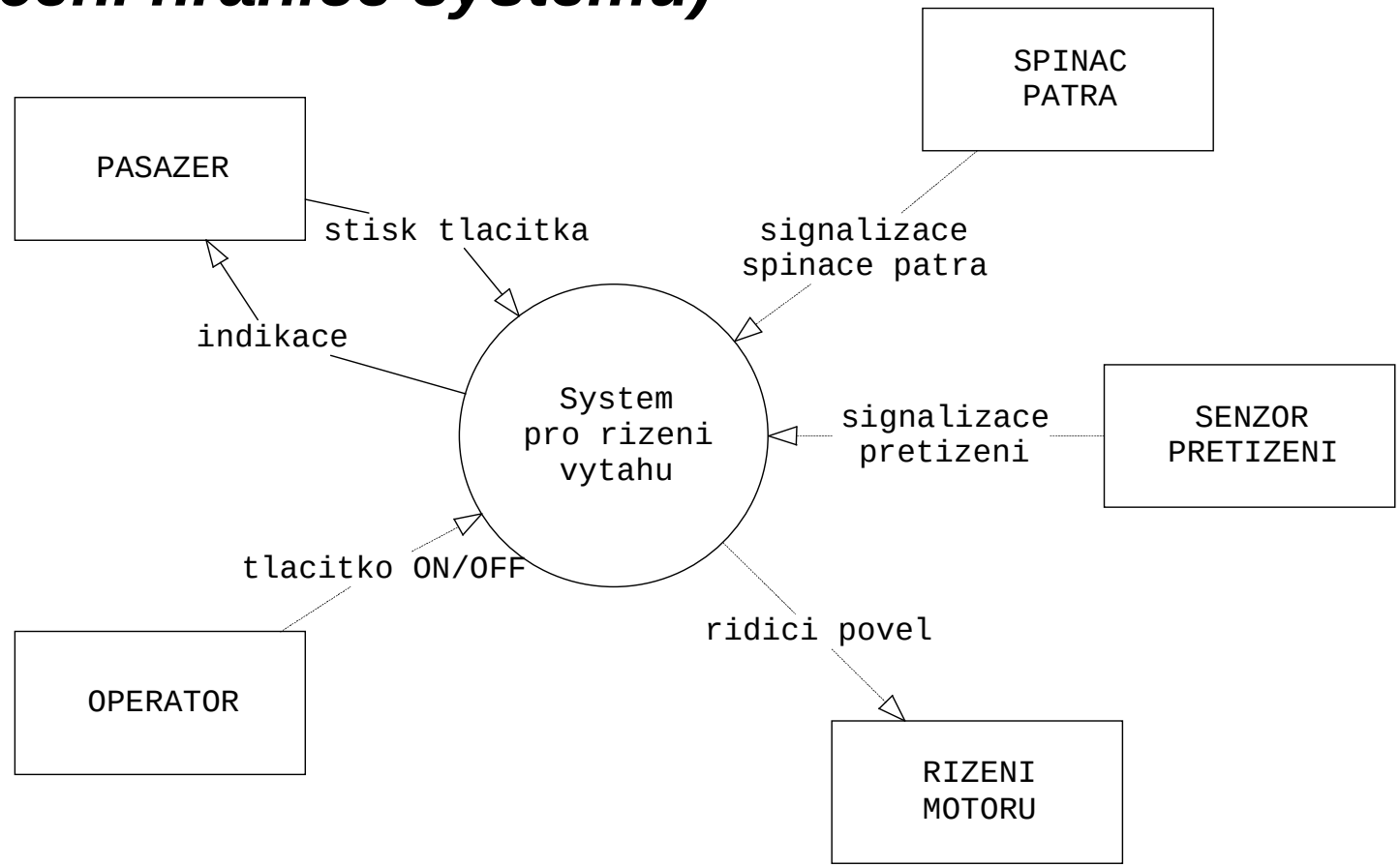
- funkce (procesy, akce)
- datové toky (data flows) - *orientované hrany vyznačující toky dat*
- datové paměti (data stores) - *místa, kde si potřebujeme něco pamatovat*
- aktéři (terminátory) - *uživatelské role nebo spolupracující systémy*

Notace DFD (Yourdon)

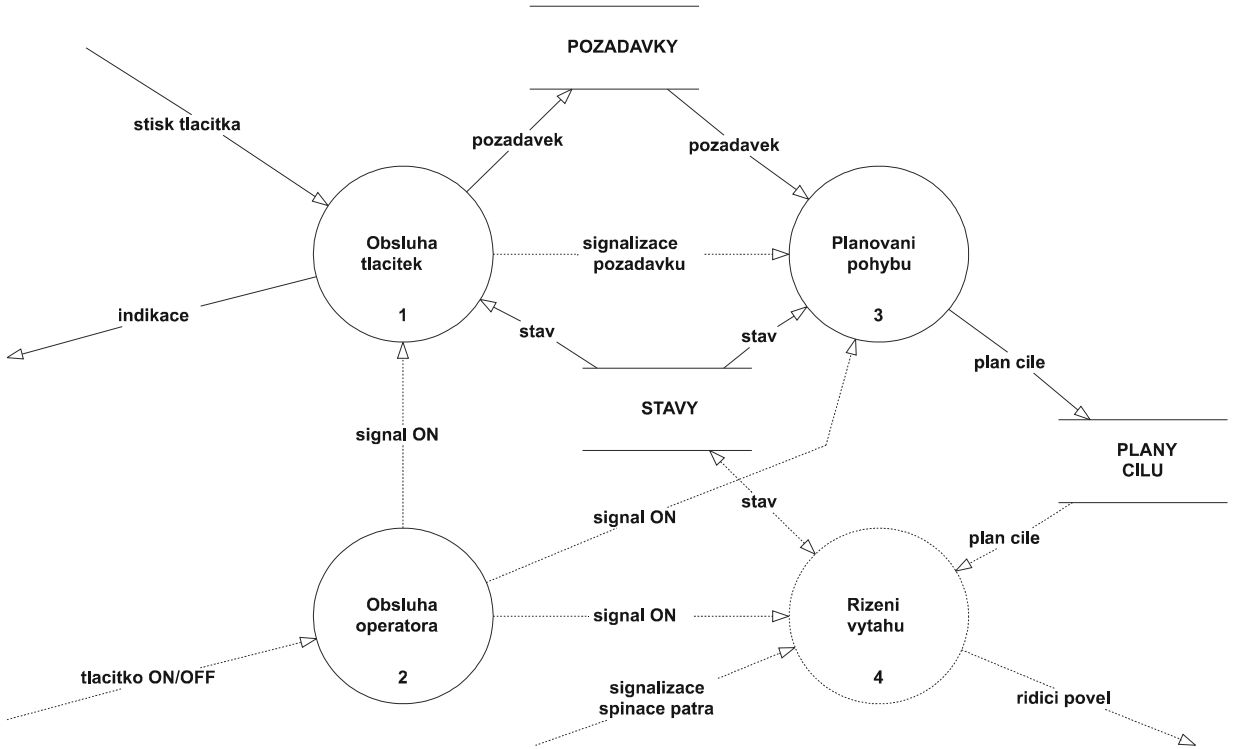


Kontextový diagram pro “Výtah”

(určení hranice systému)



DFD pro výtah (úroveň 0)



Diagramy aktivit

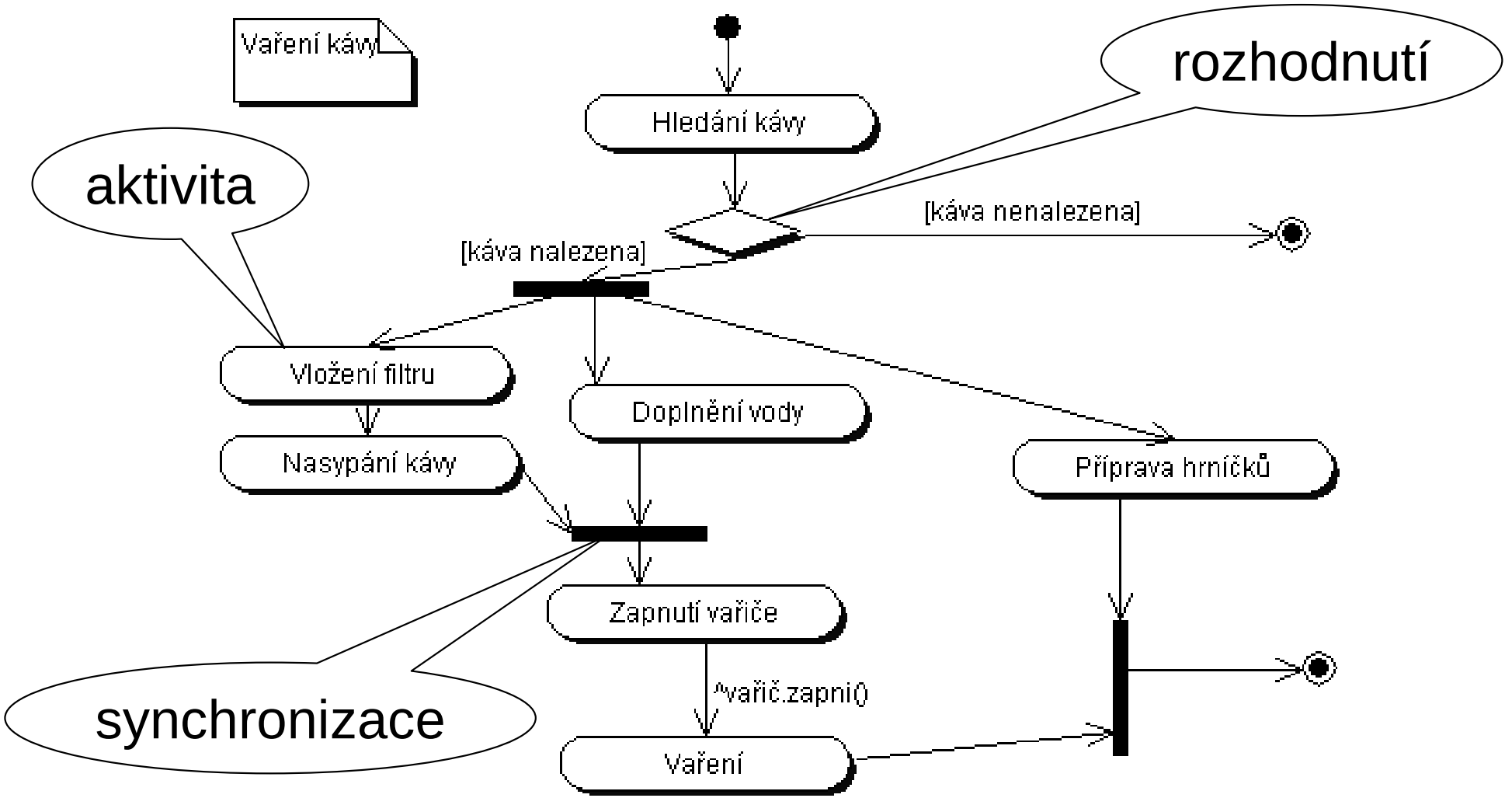
- Diagramy popisující aktivity (procesy) a jejich návaznosti.
- Přechody mezi aktivitami jsou vyvolány dokončením akce (jsou synchronní).
- Používají se pro dokumentaci tříd, metod, nebo případů použití (jako „workflow“).
- Obecně se mohou použít pro procesní modelování.
- Nahrazují v UML neexistující diagramy datových toků.

Diagramy aktivit (Activity diagrams)

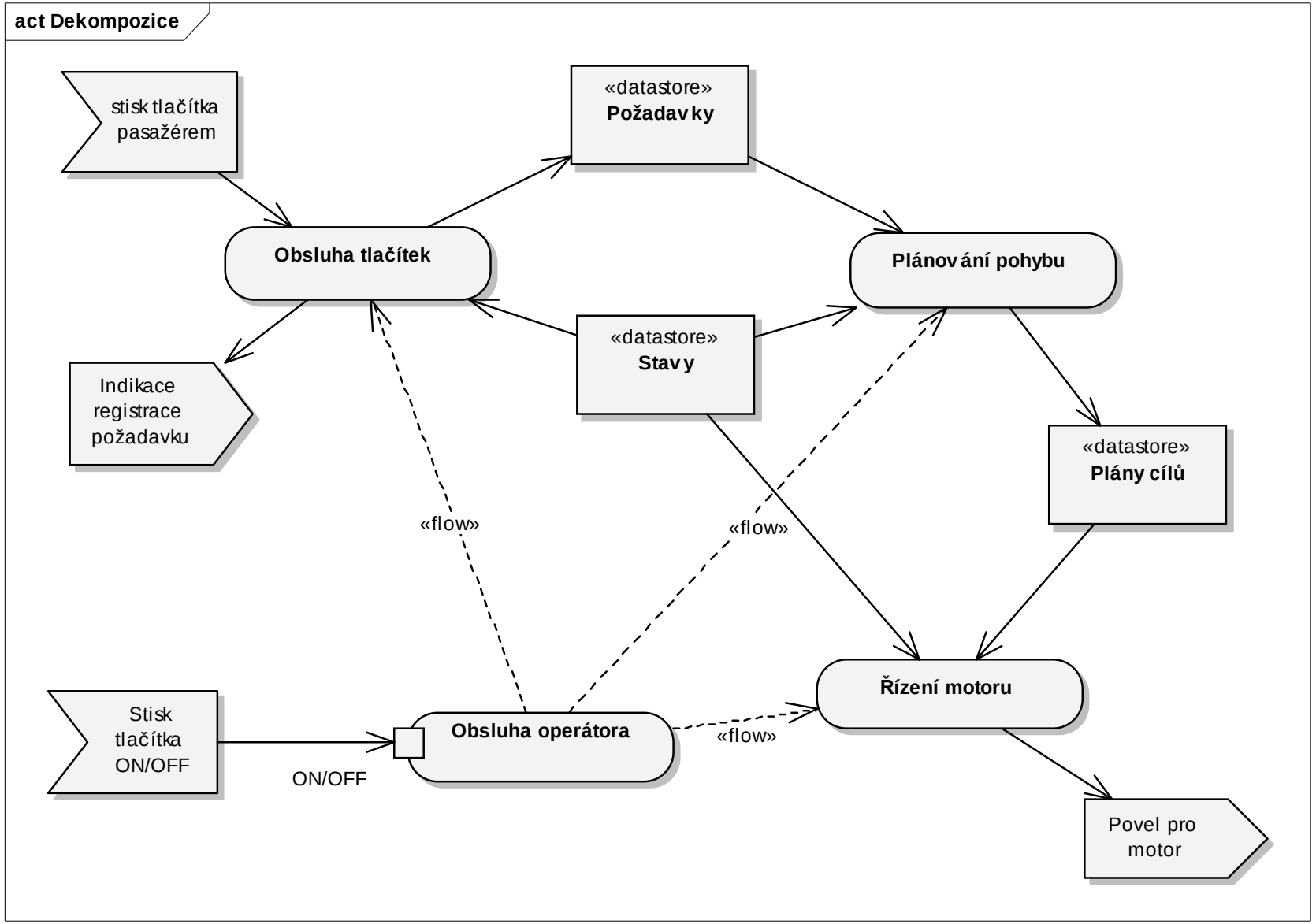
Prvky:

- **Aktivity** – činnosti, které modelujeme
- **Přechody** – po ukončení činnosti se přejde k činnosti jiné
- **Objekty** – s činností může souviset vytváření nebo konzumace objektů
- **Začátek**, **Konec**
- **Synchronizační značky** (rozvětvení a synchronizace)
- **Plavecké dráhy** – okruhy zodpovědností

Příklad diagramu aktivity



DFD v UML – diagram aktivity



Možný model vývoje, tj. rozklad produkce software na kroky a definice potřebných artefaktů

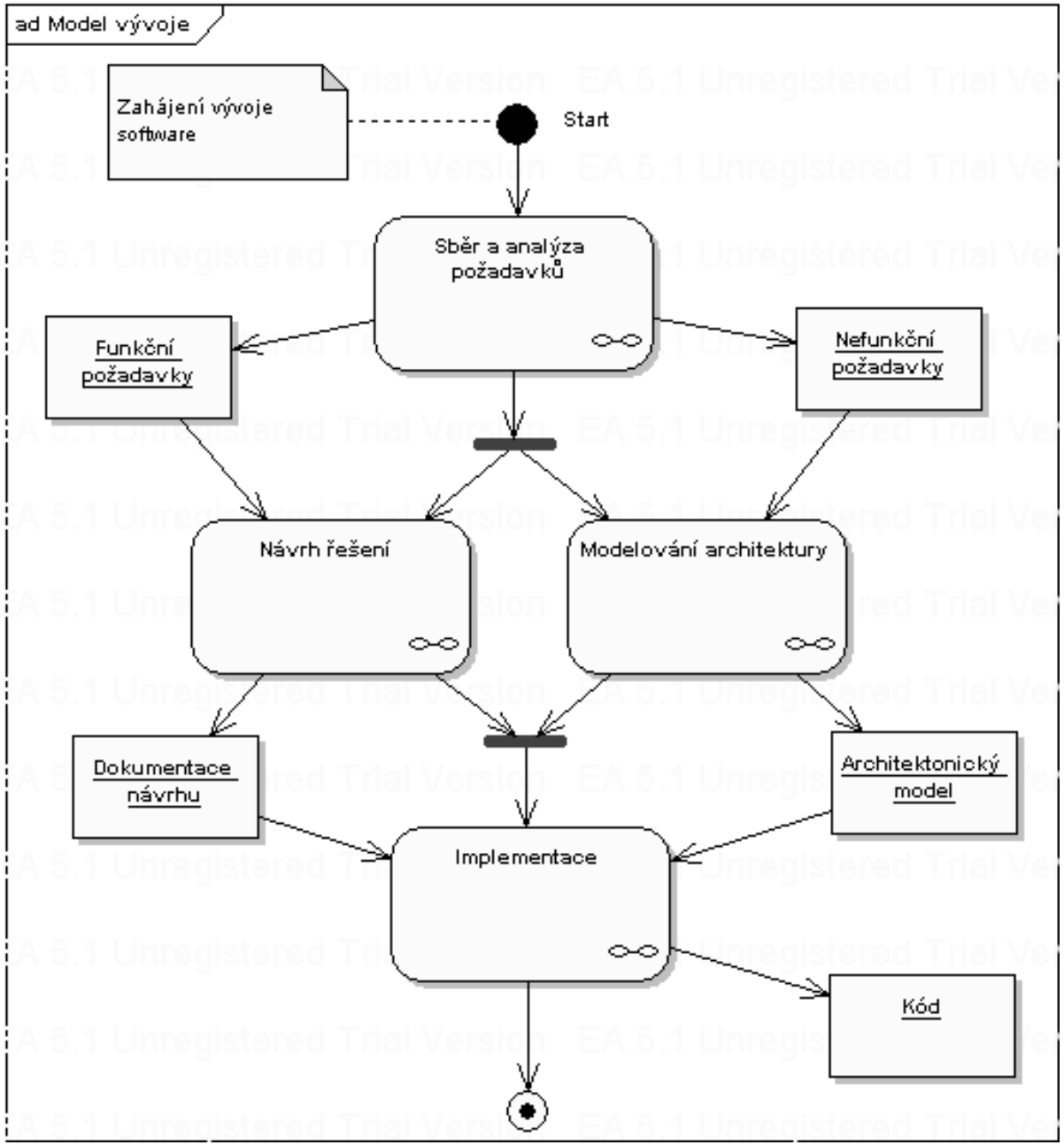


Diagram aktivity pro „přivolání výtahu“

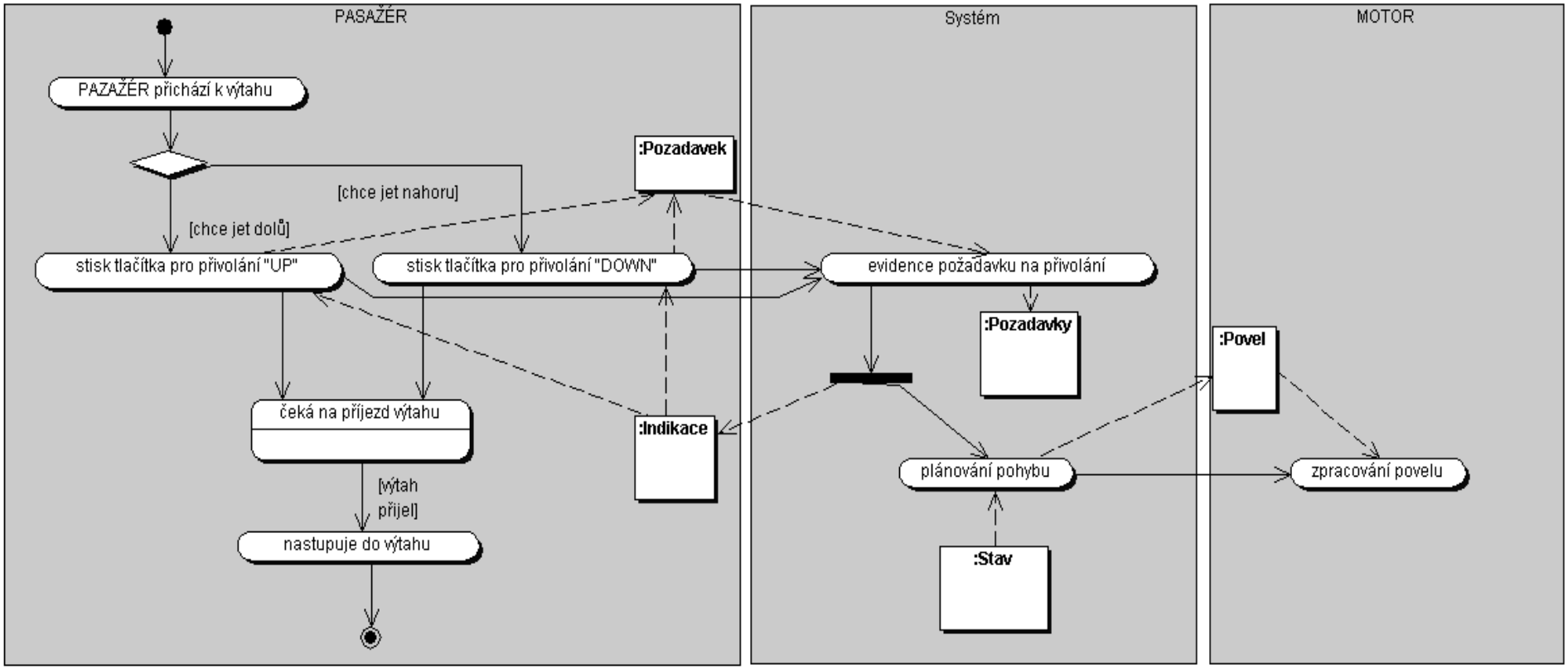
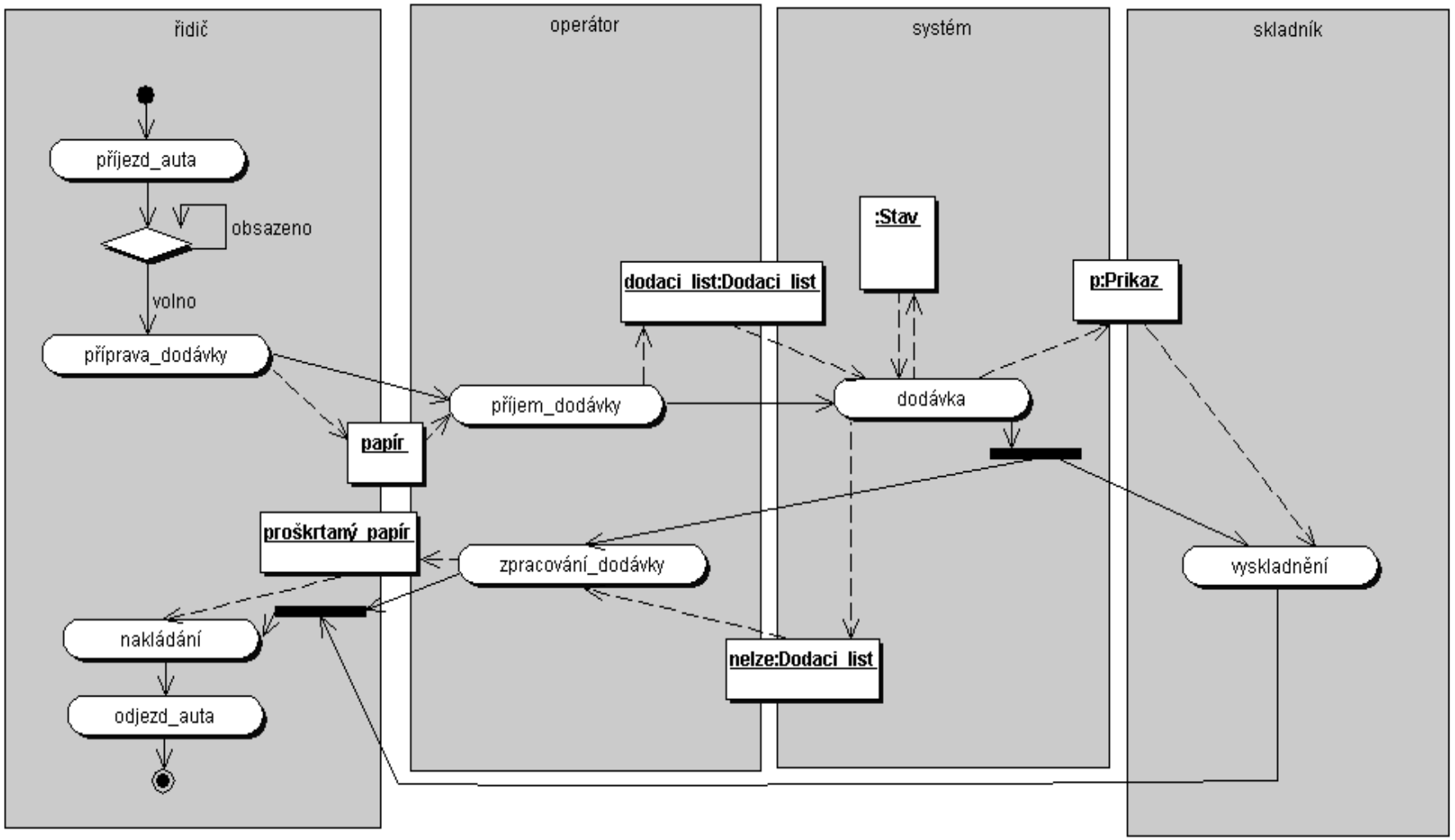


Diagram aktivity pro „dodávku“



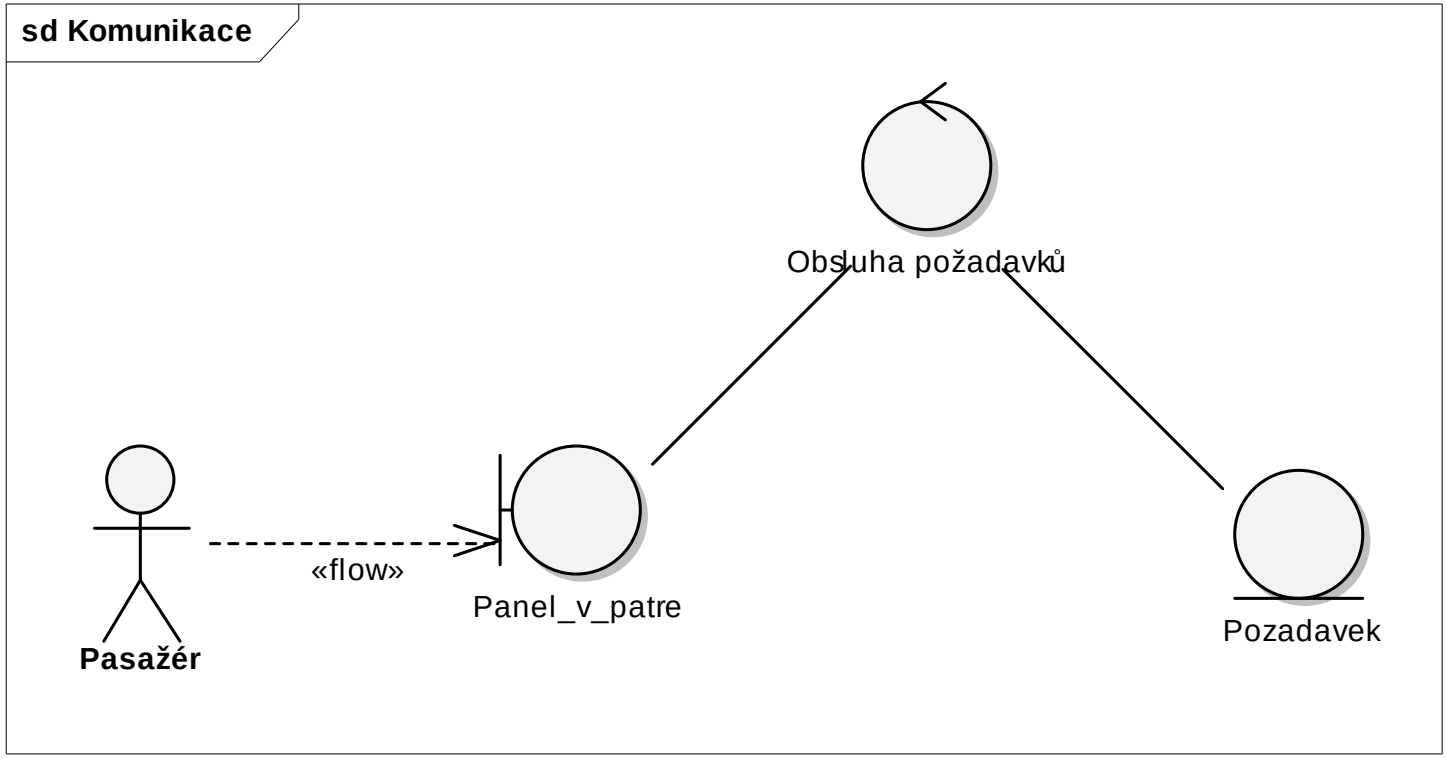
Diagramy komunikace (interakce)

(zachycení komunikace mezi objekty)

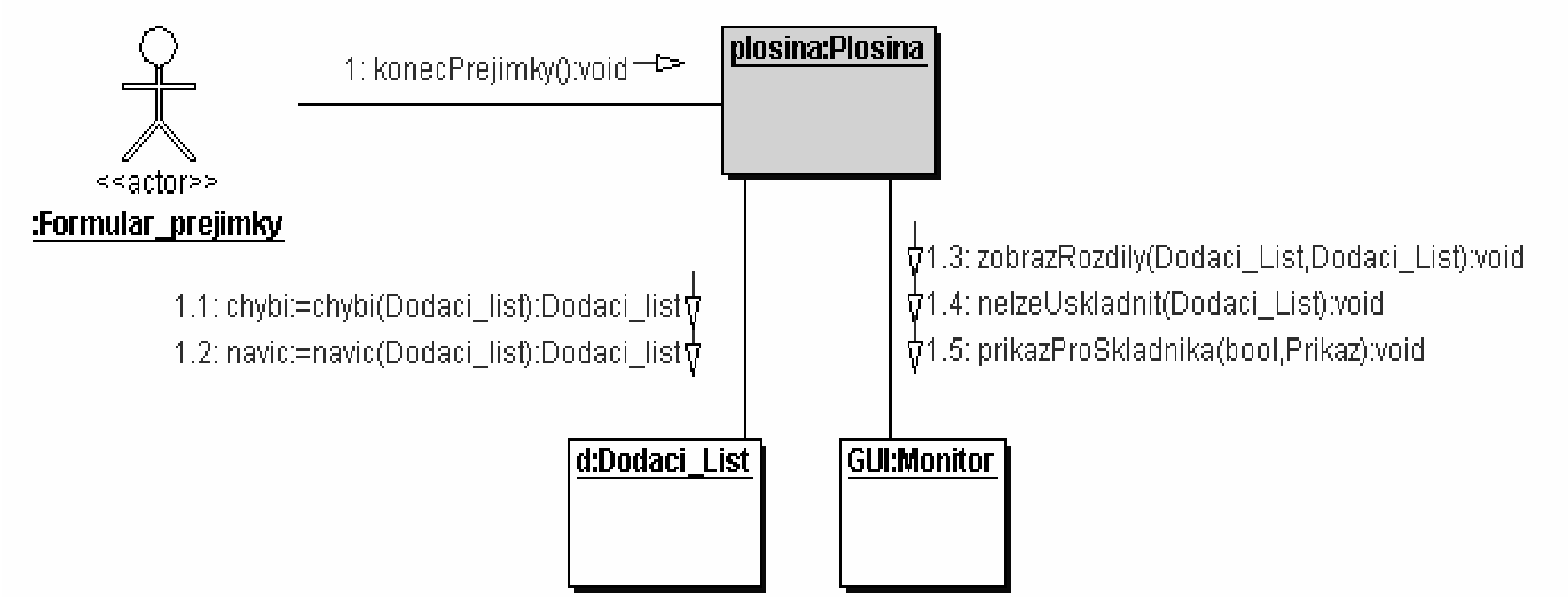
Prvky:

- **Objekty** - znázorněné jako obdélníky
- **Interakce mezi objekty** (stimuly) - orientovaná spojení mezi objekty
- **Zprávy** – dokumentace zpráv, které si objekty mezi sebou posílají, včetně parametrů a návratových hodnot - odezvy na události (výstupy)
- **Čas** – vyznačen číslováním

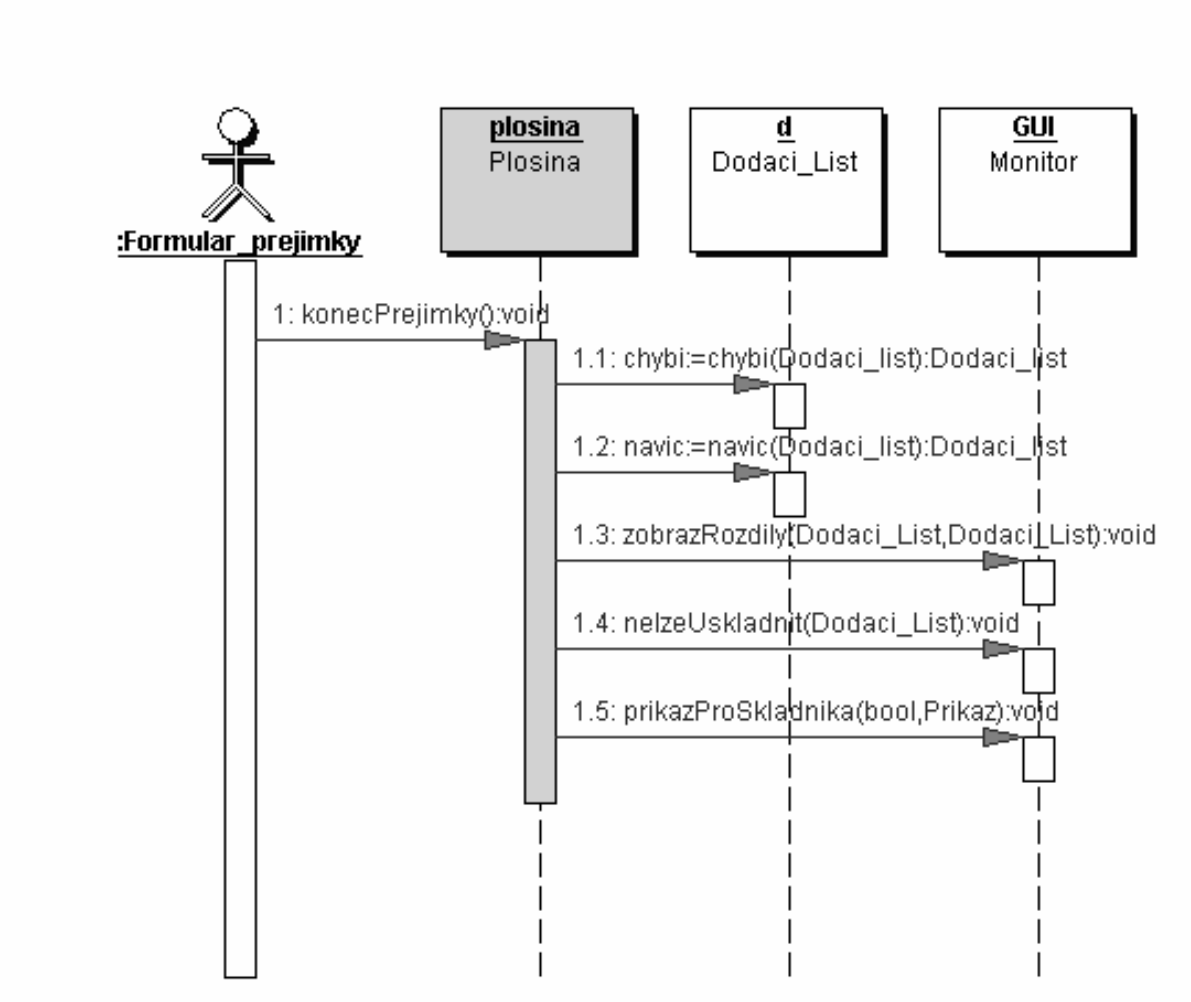
Triviálně


sd Komunikace
Obsluha požadavků
«flow»
Panel_v_patre
Pozadavek
Pasažér

Podrobněji

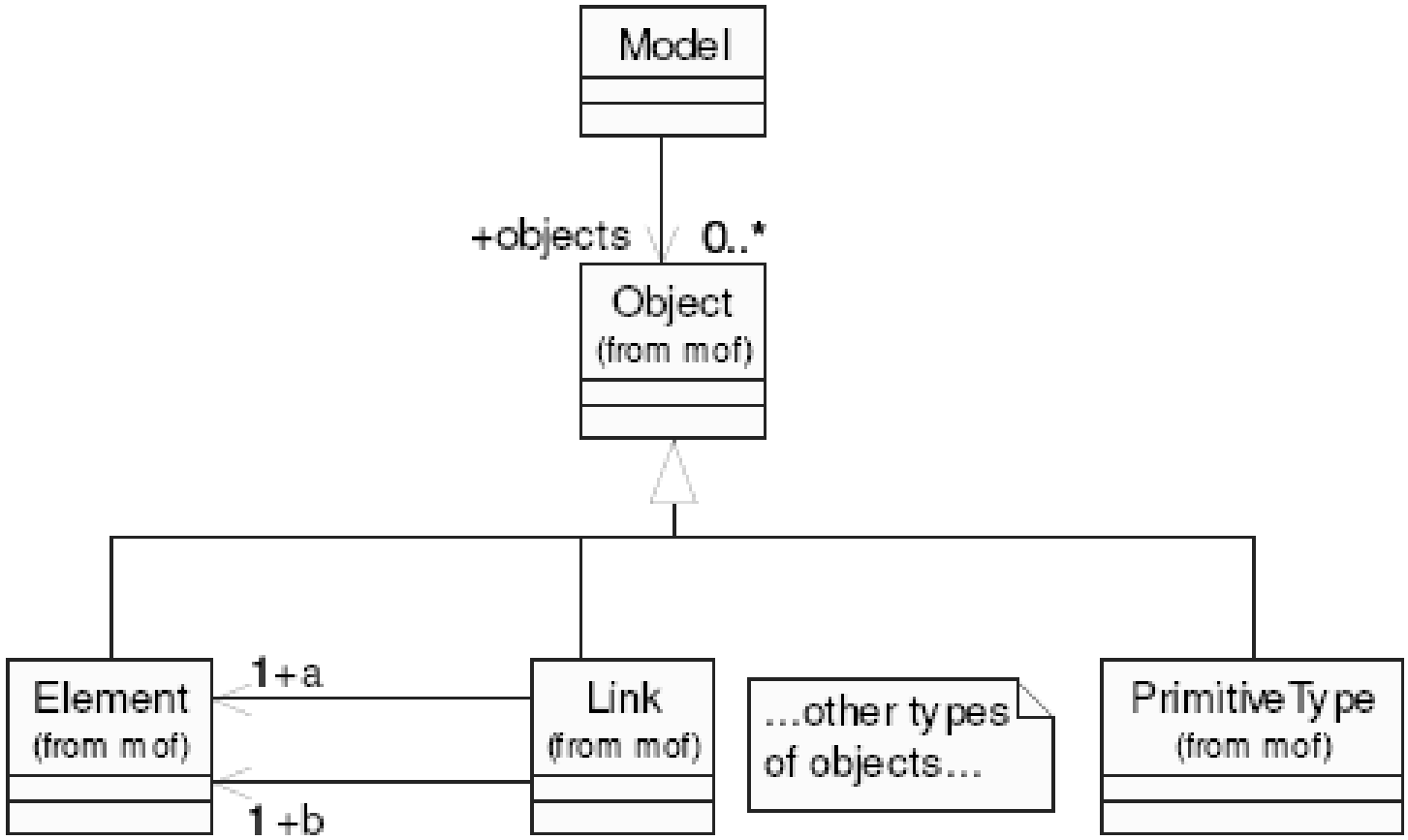

<<actor>>
:Formular_prejimky
1: konecPrejimky():void
plosina:Plosina
1.1: chybi:=chybi(Dodaci_list):Dodaci_list
1.2: navic:=navic(Dodaci_list):Dodaci_list
1.3: zobrazRozdily(Dodaci_List,Dodaci_List):void
1.4: nelzeUskladnit(Dodaci_List):void
1.5: prikazProSkladnika(bool,Prikaz):void
d:Dodaci_List
GUI:Monitor

Nebo jako scénář



Datový model

Př.: Model podle OMG MDA



Zdroj:http://www.omg.org/docs/ormsc/05-04-01.pdf

Datově orientovaná analýza

- Seznam událostí, kontext, datový slovník
- Identifikace dat, která s událostmi souvisí (identifikace základních objektů)
- Identifikace vztahů mezi objekty
- Scénáře jednání (původce, událost, akce, participanti, výstupy - reakce)
- Modelování životních cyklů objektů
- Popis akcí (minispecifikace základních akcí)

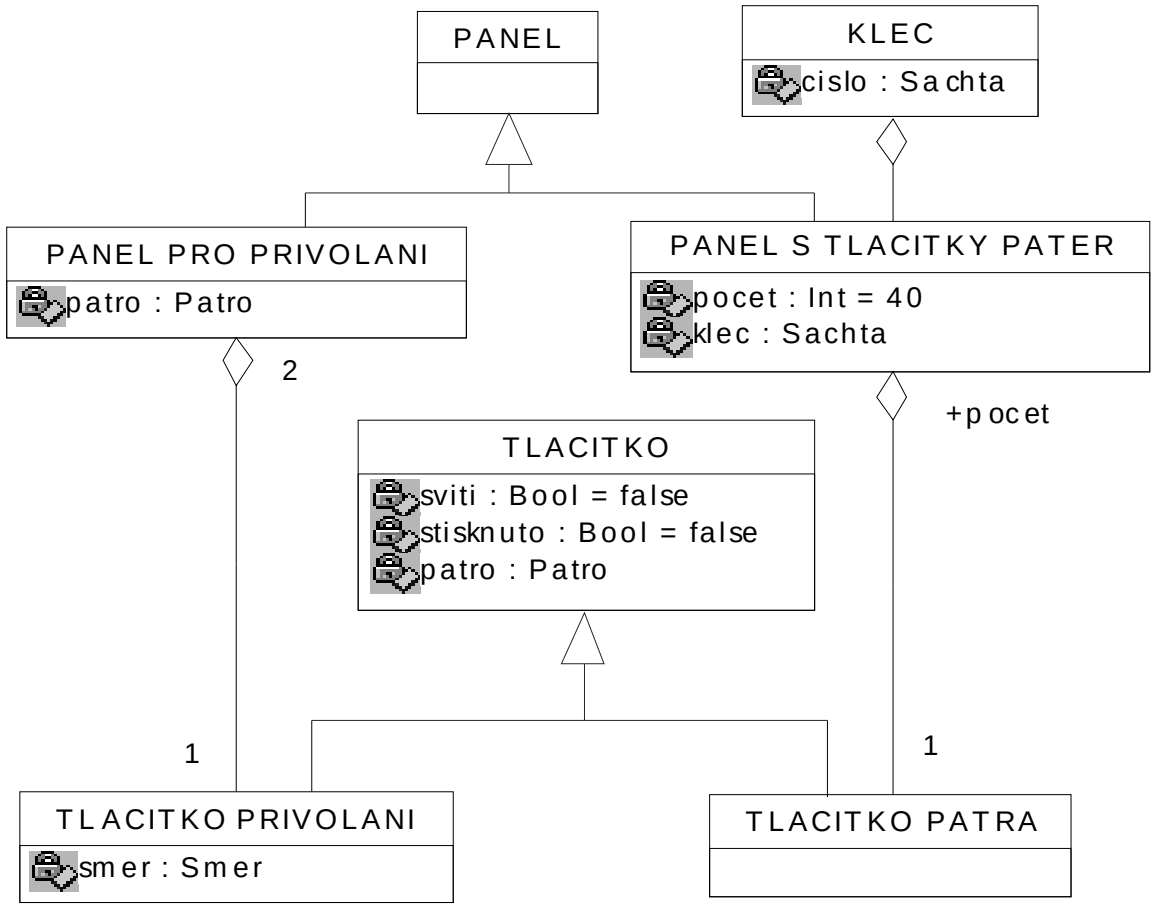
Datový model (konceptuální)

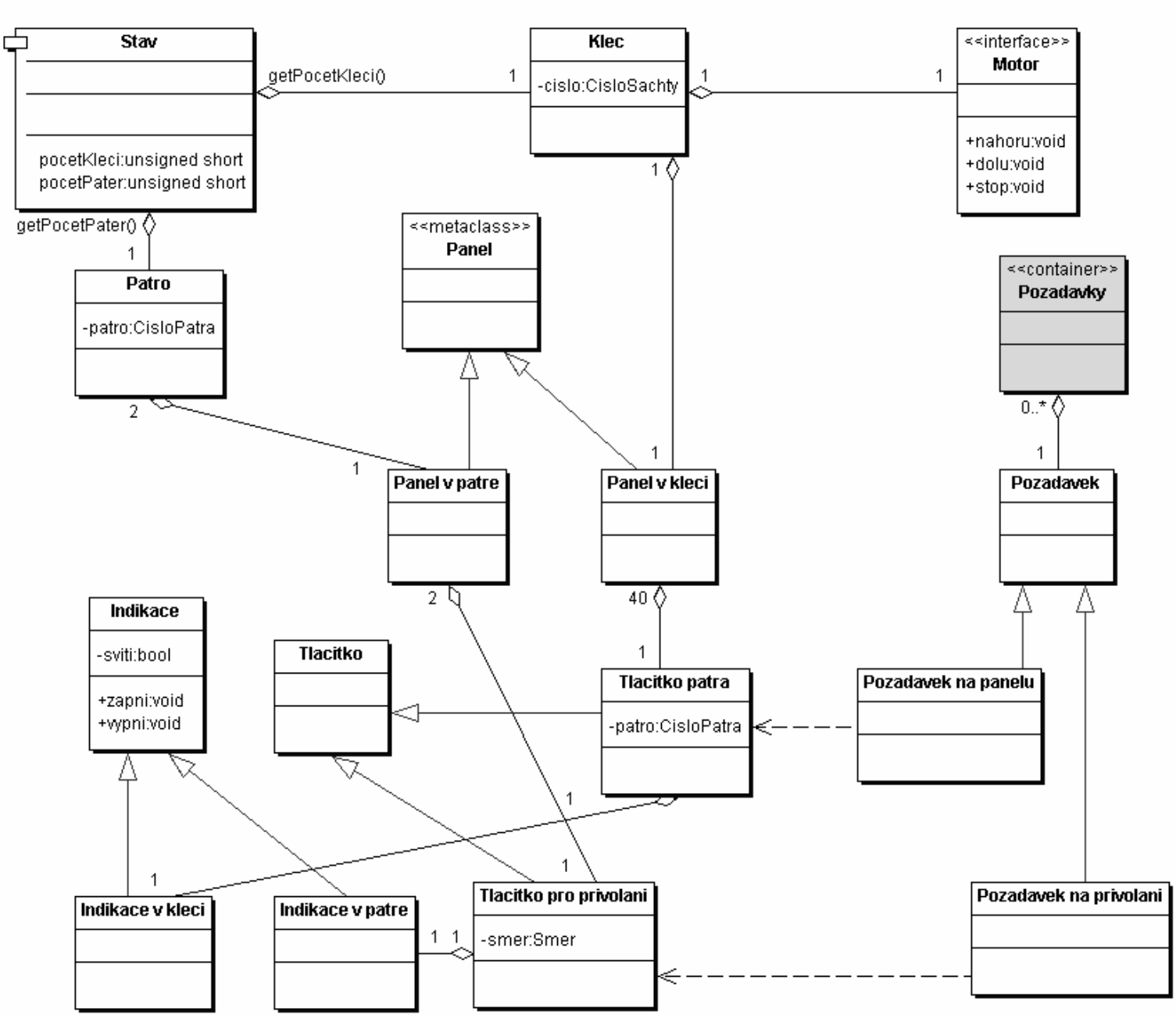
(zachycení analýzy dat)

Komponenty:

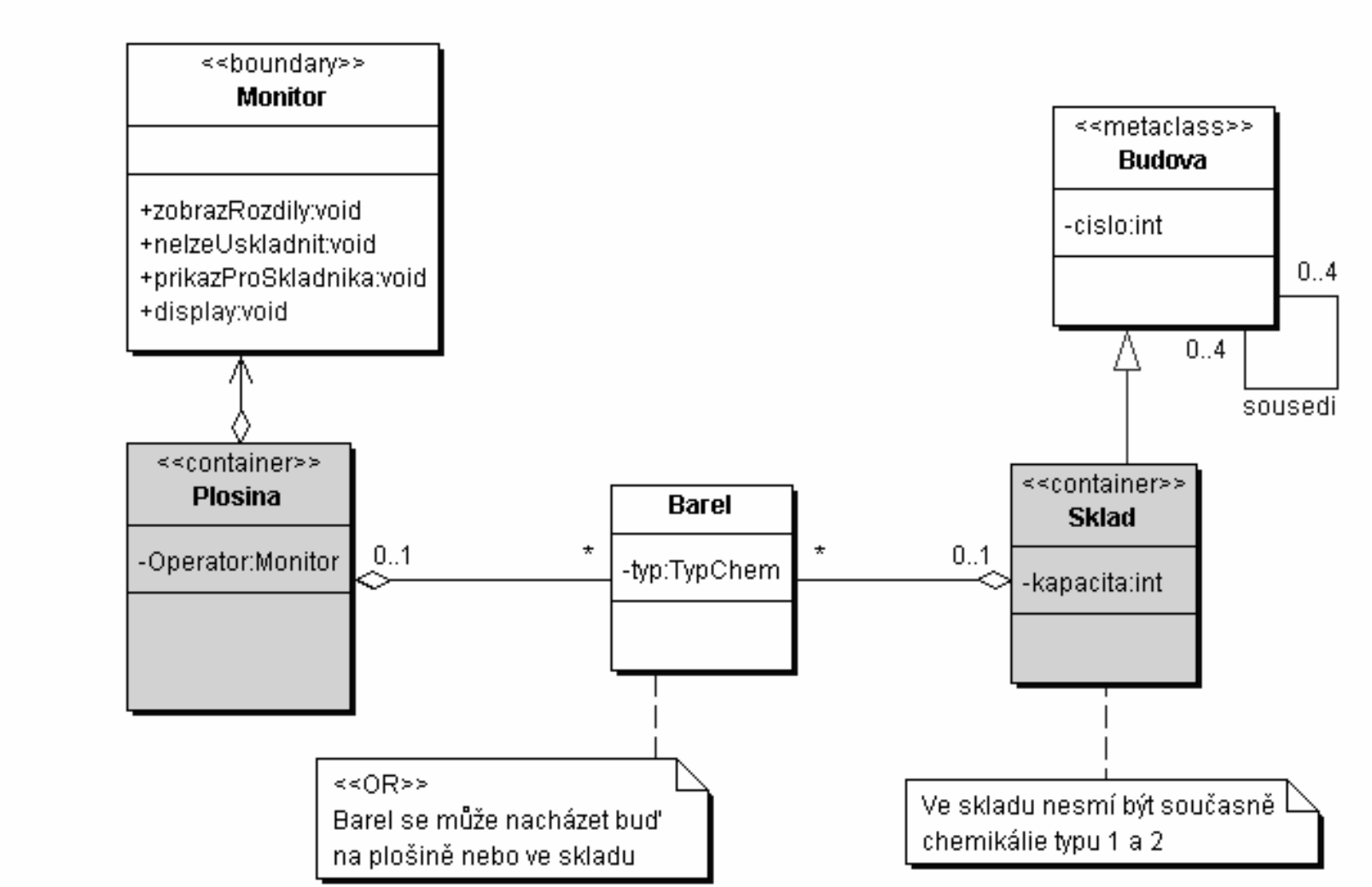
- **typy objektů (entity)** - entita = rozlišitelný identifikovatelný objekt
- **vztahy (relationships)** - množiny instancí reprezentujících vztahy mezi (2 a více) objekty
- **indikace přidružených objektů** - pro vztahy o nichž si potřebujeme něco pamatovat
- **indikace vztahů nadtyp-podtyp, celek-část**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

Datový model pro „Výtah“ (1.verze)




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)



Datově orientovaná analýza

- Vychází z představy, že základem IS jsou data. Služby IS slouží pro pořízení a exploraci dat.
- Doporučuje proto nejprve analyzovat požadavky a definovat konceptuální datový model řešeného systému.
- Konceptuální datový model musí postihovat data přicházející přes hranici systému jako vstupní data související s událostmi, dále data, která se v systému ukládají a nakonec rovněž data, která systém produkuje na výstupu.
- Teprve později doplníme model o další části.

Postup datově orient. analýzy

1. Seznam událostí, kontext, datový slovník
2. Identifikace dat, která s událostí souvisí (základních objektů)
3. Identifikace vztahů mezi objekty
4. Scénáře jednání (původce, událost, akce, participanti, výstupy - reakce)
5. Modelování životních cyklů objektů
6. Popis akcí (minispecifikace základních akcí)

Jak hledat data?

Doporučení č.1:

- Analyzujeme odborný článek, vybereme všechna podstatná jména.
- Roztřídíme je do skupin:
  - kandidáti na typy objektů (entity),
  - kandidáti na vlastnosti objektů (atributy),
  - ostatní (kandidáti na aktéry, smetí).

Příklad: Odborný článek pro „Výtah“

Systém “Výtah” slouží pro logické řízení obsluhy výtahu s jednou či více šachtami (předpokládají se 4 šachty a 40 úrovní). Systém zajišťuje efektivní plánování sběru a odvozu pasažérů mezi obsluhovanými patry podle požadavků (požadavek na přivolání výtahu pro jízdu směrem nahoru nebo dolů, požadavek na dopravení do určitého patra). Směr jízdy se nemění, dokud výtah nesplní objednávky v daném směru (výtah neví o pasažérech – neexistuje indikace prázdnosti klece). Přeplněný výtah nereaguje na výzvy (existuje indikace přetížení). Pro každou šachtu existuje samostatný motor ovládaný signály (povely UP, DOWN a STOP). Povel STOP způsobí zastavení výtahu v nejbližším patře v daném směru a otevření dveří výtahu (dveře se dají otevřít až v patře). Uvnitř klece je panel s tlačítky pater, indikace aktuální polohy a tlačítko STOP. Tlačítko STOP zabrání zavření dveří (jde mimo systém). Rovněž otevírání a zavírání dveří jde mimo systém (kvůli bezpečnosti). Příkazy pro systém jsou akceptovány až po zavření dveří. Operátor výtahu má k dispozici tlačítko ON/OFF, kterým zadává požadavek na zastavení pohybu výtahů.

Zpracovaný článek

systém “Výtah”
logické řízení
šachta
úroveň
pasažér
patro
požadavek
požadavek na přivolání výtahu pro jízdu směrem nahoru
požadavek na přivolání výtahu pro jízdu směrem dolů
požadavek na dopravení do patra
směr jízdy
objednávka
indikace prázdnosti klece
výzva
indikace přetížení
motor
signál
povel UP
povel DOWN
povel STOP
dveře výtahu
klec
panel s tlačítky pater
indikace aktuální polohy
tlačítko STOP
příkaz pro systém
operátor výtahu
tlačítko ON/OFF
požadavek na zastavení pohybu

Kandidáti na aktéry

pasažér

indikace přetížení

motor

indikace aktuální polohy (patra)

tlačítko STOP

operátor výtahu

tlačítko ON/OFF

Kandidáti na typy dat

šachta (atribut klece)

úroveň alias patro

požadavek alias objednávka alias příkaz pro systém alias výzva

požadavek na přivolání výtahu pro jízdu směrem nahoru

požadavek na přivolání výtahu pro jízdu směrem dolů

požadavek na dopravení do patra

směr jízdy (atribut)

indikace prázdnosti klece (neexistuje)

indikace přetížení

signál alias povel (pro motor)

povel UP

povel DOWN

povel STOP

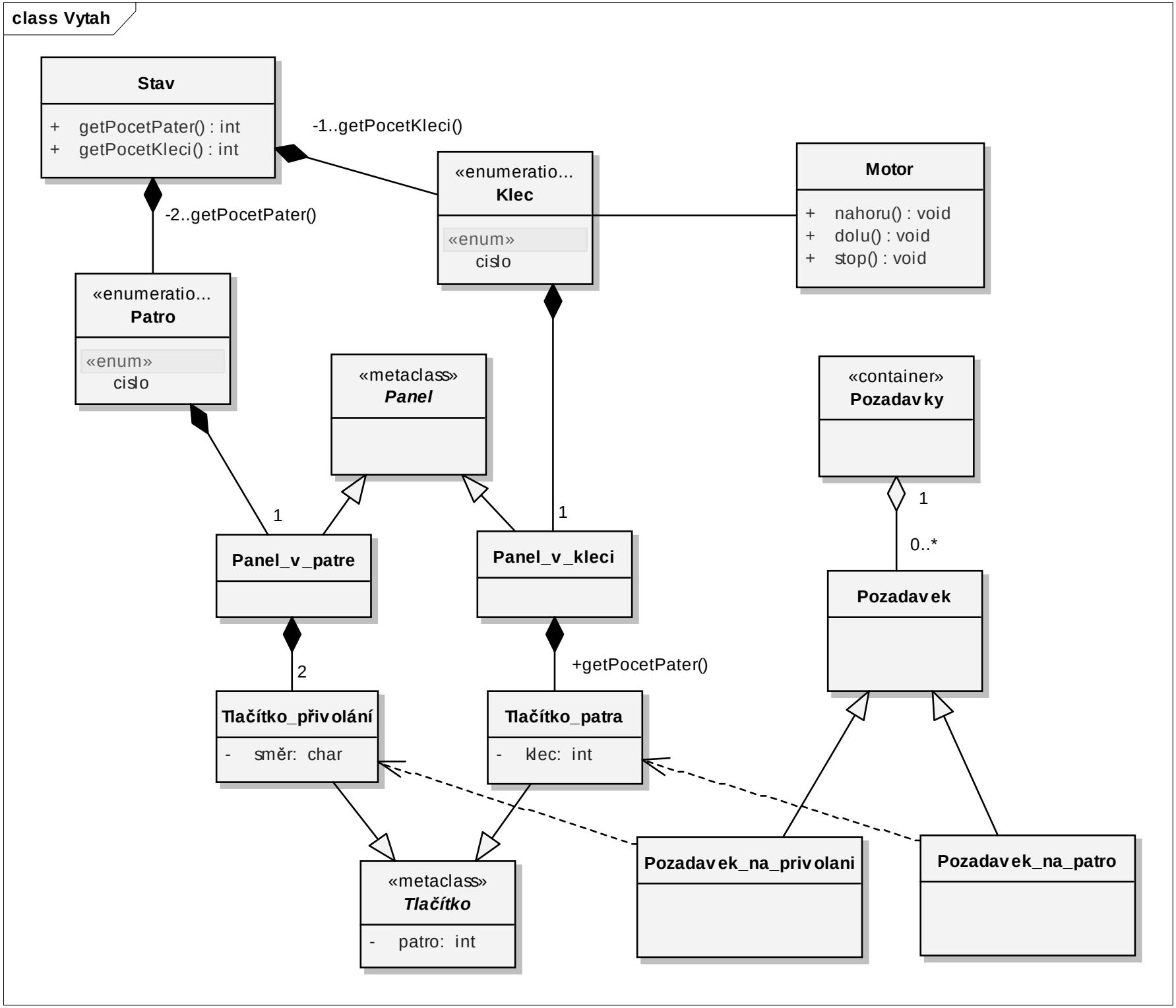
klec

panel s tlačítky pater

indikace aktuální polohy

tlačítko STOP (jde mimo systém)

tlačítko ON/OFF alias požadavek na zastavení pohybu

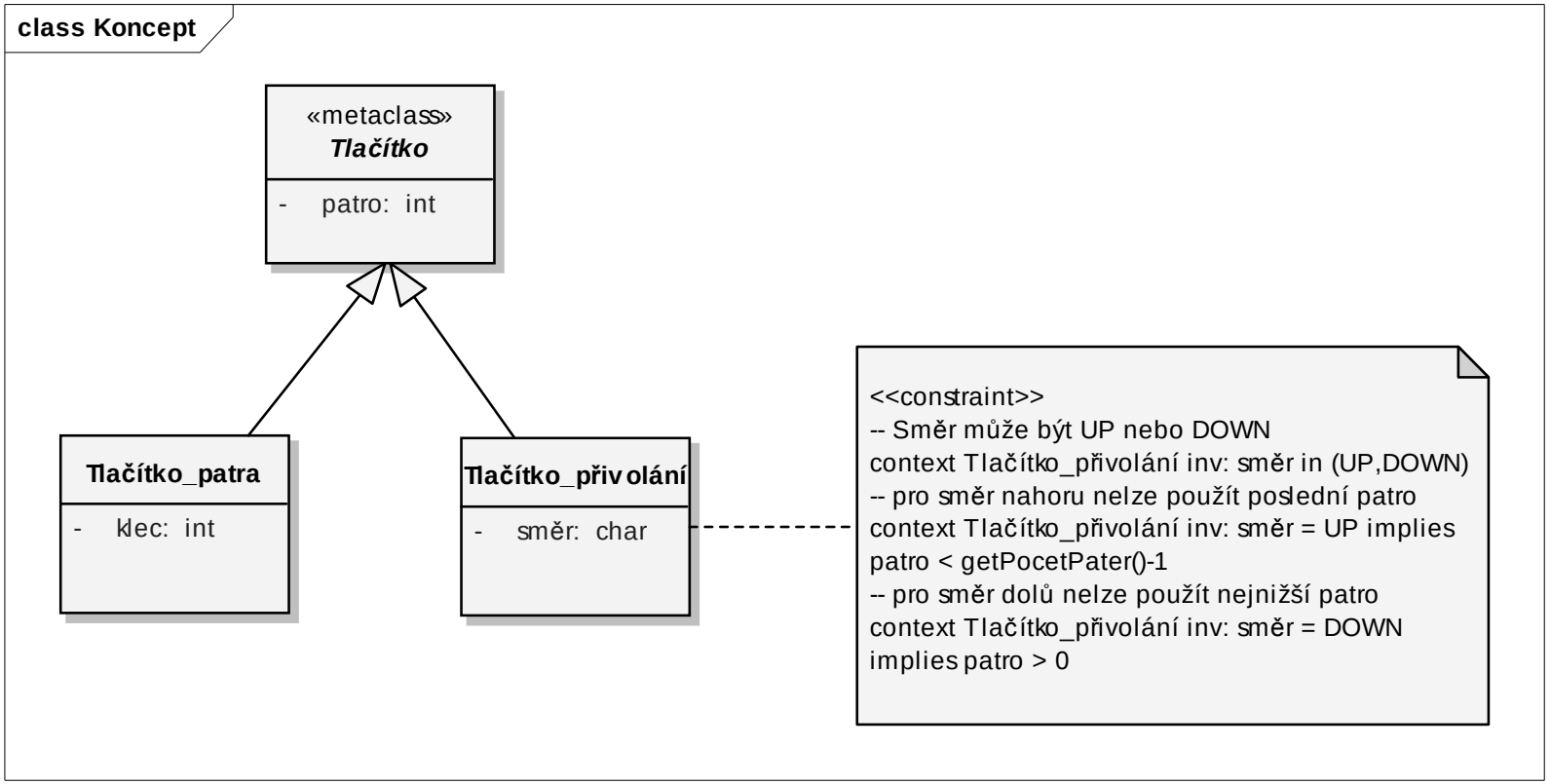

class Vytah
Stav
+ getPocetPater() : int
+ getPocetKleci() : int
-1..getPocetKleci()
-2..getPocetPater()
«enumeratio...
Klec
«enum»
cislo
Motor
+ nahoru() : void
+ dolu() : void
+ stop() : void
«enumeratio...
Patro
«enum»
cislo
«metaclass»
Panel
«container»
Pozadavky
1
1
1
0..*
Panel_v_patre
Panel_v_kleci
Pozadavek
2
+getPocetPater()
Tlačítko_přivolání
- směr: char
Tlačítko_patra
- klec: int
«metaclass»
Tlačítko
- patro: int
Pozadavek_na_privolani
Pozadavek_na_patro

Něco diagramem vyjádřit nelze

V příkladu systému Výtah je to např.:

- Tlačítko pro přivolání pro jízdu směrem nahoru na panelu v posledním patře, tj. když patro má hodnotu getPocetPater() neexistuje.
- Tlačítko pro přivolání pro jízdu směrem dolů na panelu v prvním patře neexistuje.

Připojení omezení k prvkům



Jak hledat data?

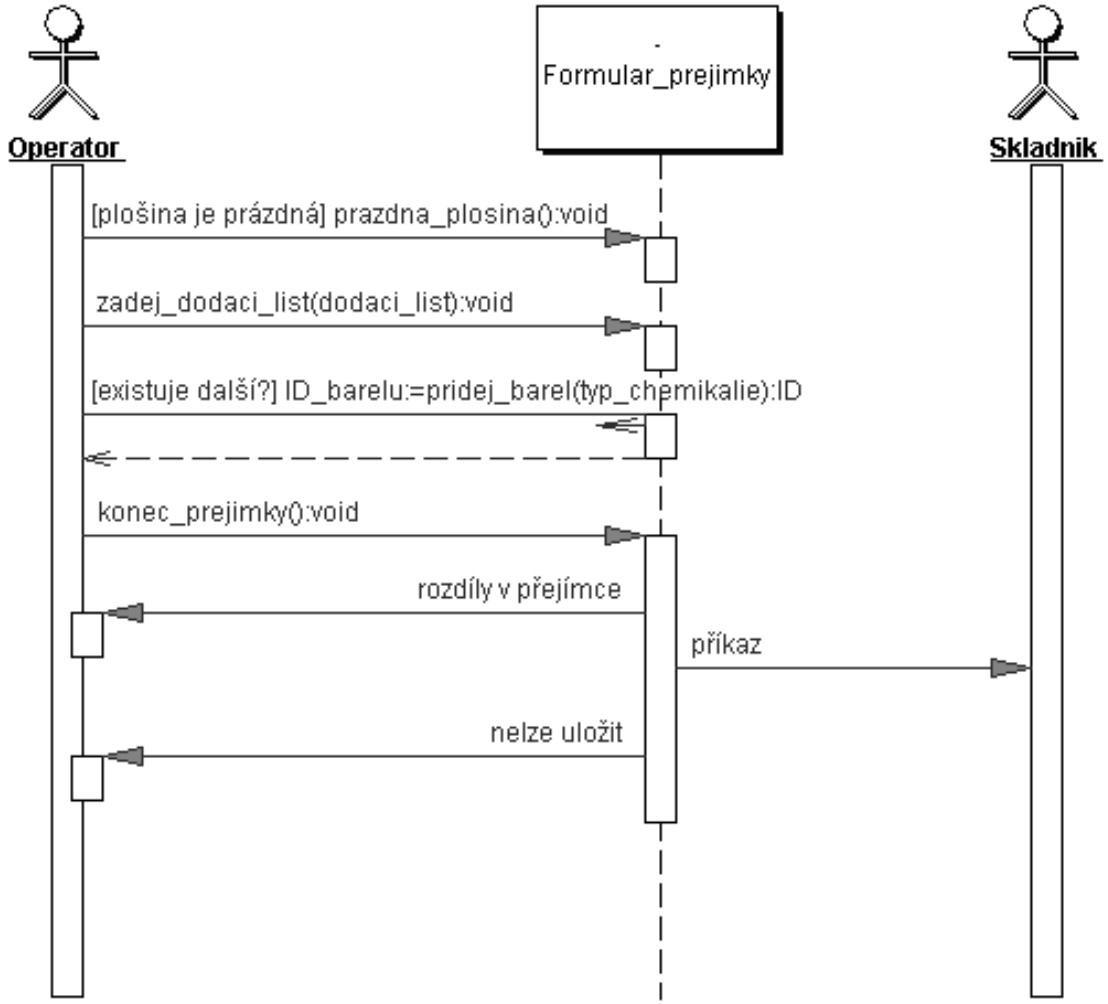
Doporučení č.2:

- Analyzujeme seznam událostí, rozpoznáváme data, která s událostmi souvisí.
- Roztřídíme je do skupin:
  - kandidáti na typy objektů (entity),
  - kandidáti na vlastnosti objektů (atributy).

Příklad: Události pro ECO sklad

- Operátor zahájil přejímku
- Operátor zahájil dodávku
- Manažer se ptá na stav skladu
- Manažer se ptá na bezpečnost skladu

Scénář pro přejímku



Kandidáti na typy dat

dodací list

barel

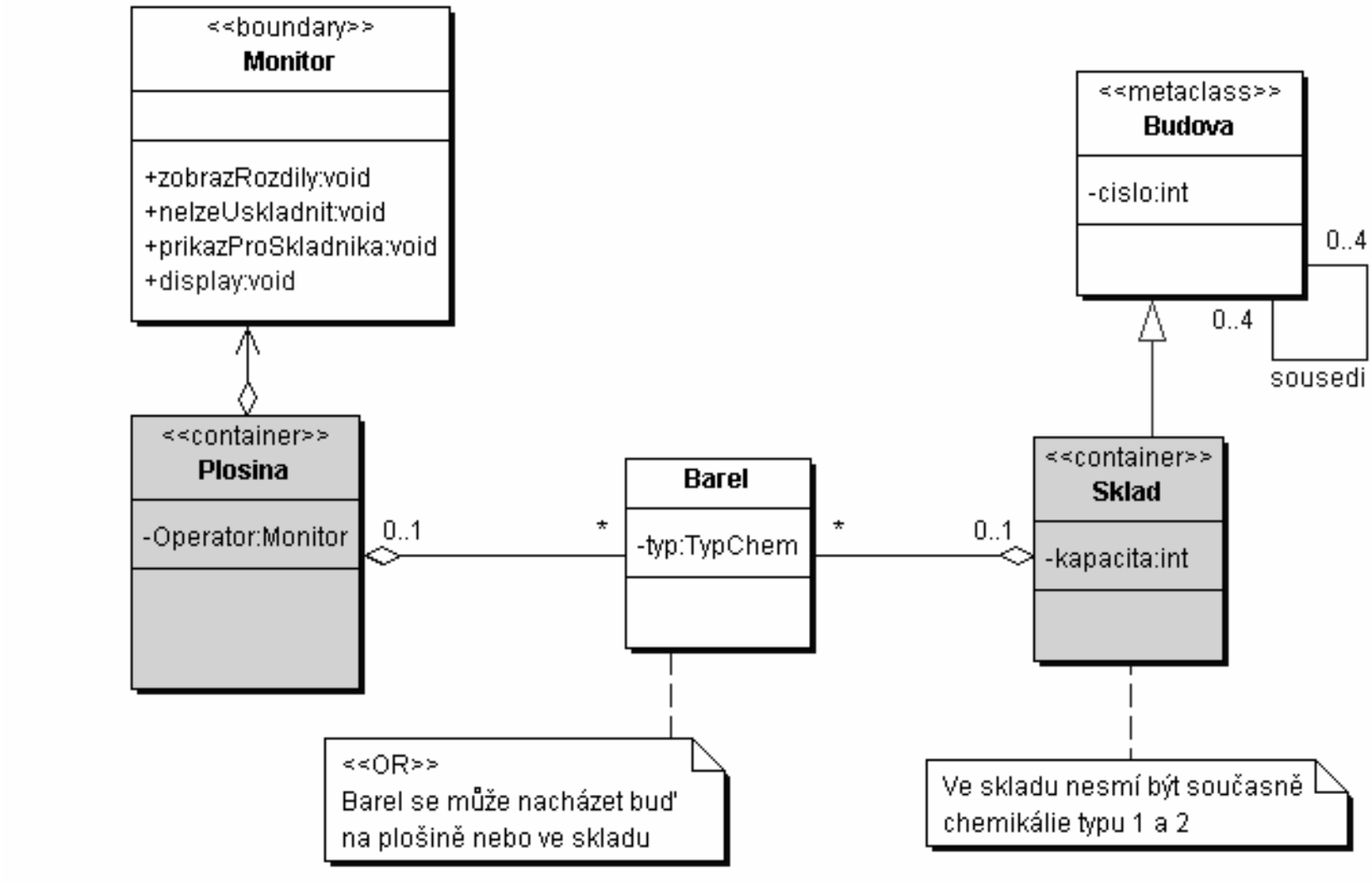
typ chemikálie

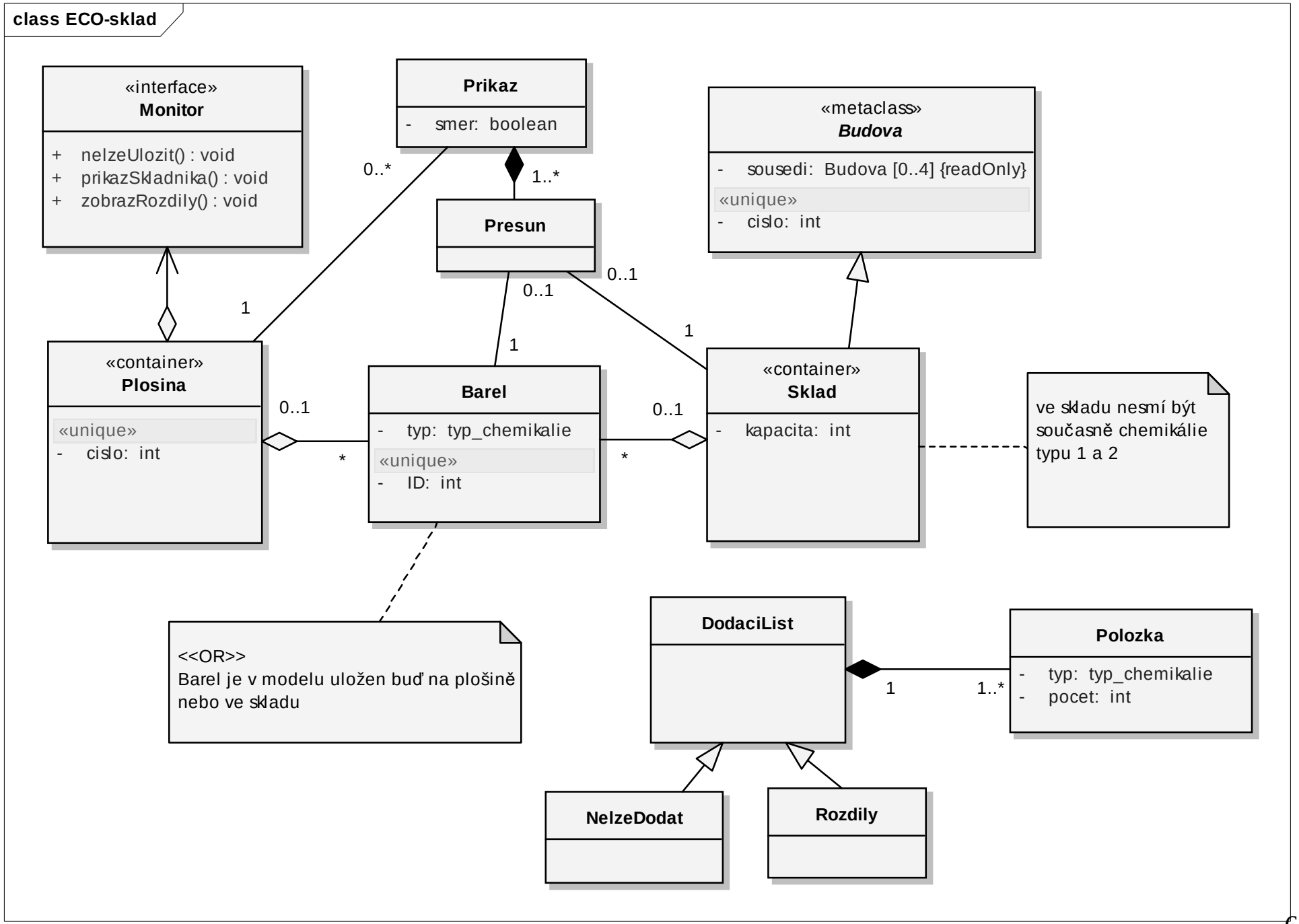
rozdíly v přejímce

nelze uložit

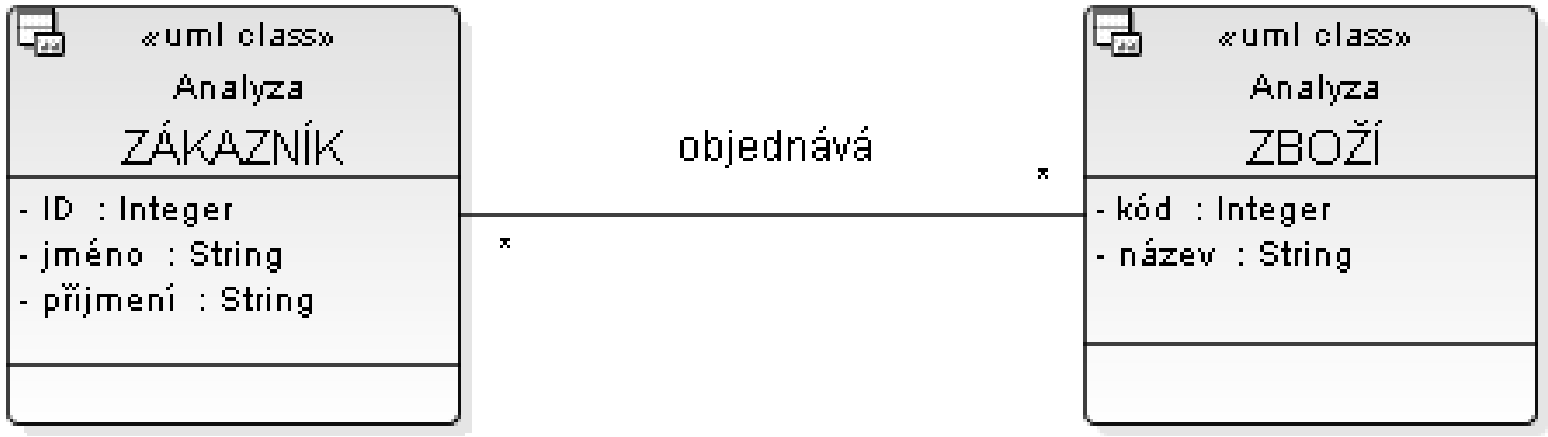
příkaz pro skladníka

Datový model pro ECO-sklad (1.)

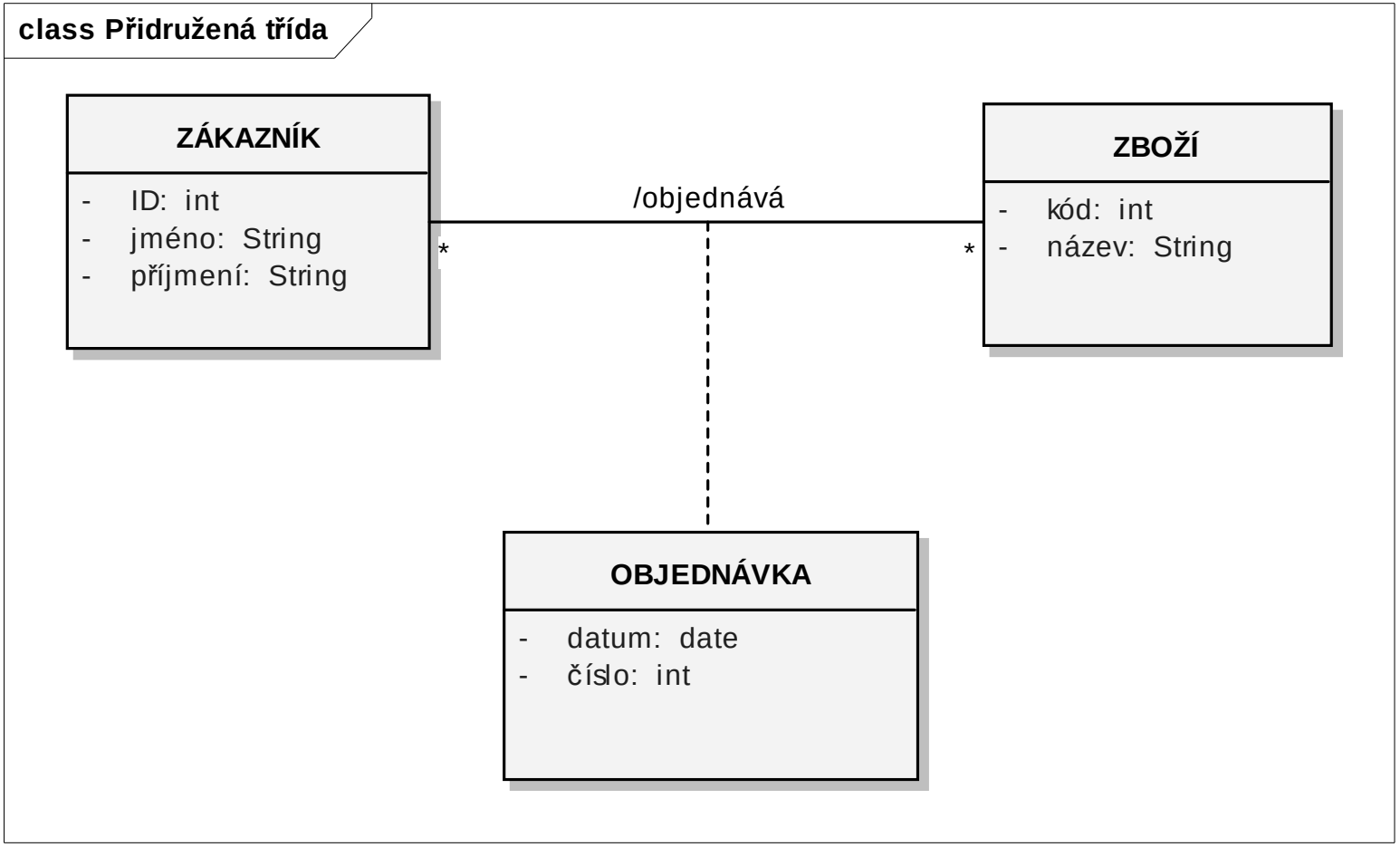



class ECO-sklad
«interface»
Monitor
+ nelzeUlozit() : void
+ prikazSkladnika() : void
+ zobrazRozdily() : void
Prikaz
- smer: boolean
0..*
1..*
Presun
«metaclass»
Budova
- sousedi: Budova [0..4] {readOnly}
«unique»
- cislo: int
0..1
0..1
1
1
1
«container»
Plosina
«unique»
- cislo: int
0..1
*
Barel
- typ: typ_chemikalie
«unique»
- ID: int
*
0..1
«container»
Sklad
- kapacita: int
ve skladu nesmí být
současně chemikálie
typu 1 a 2
<<OR>>
Barel je v modelu uložen buď na plošině
nebo ve skladu
DodaciList
1
1..*
Polozka
- typ: typ_chemikalie
- pocet: int
NelzeDodat
Rozdily

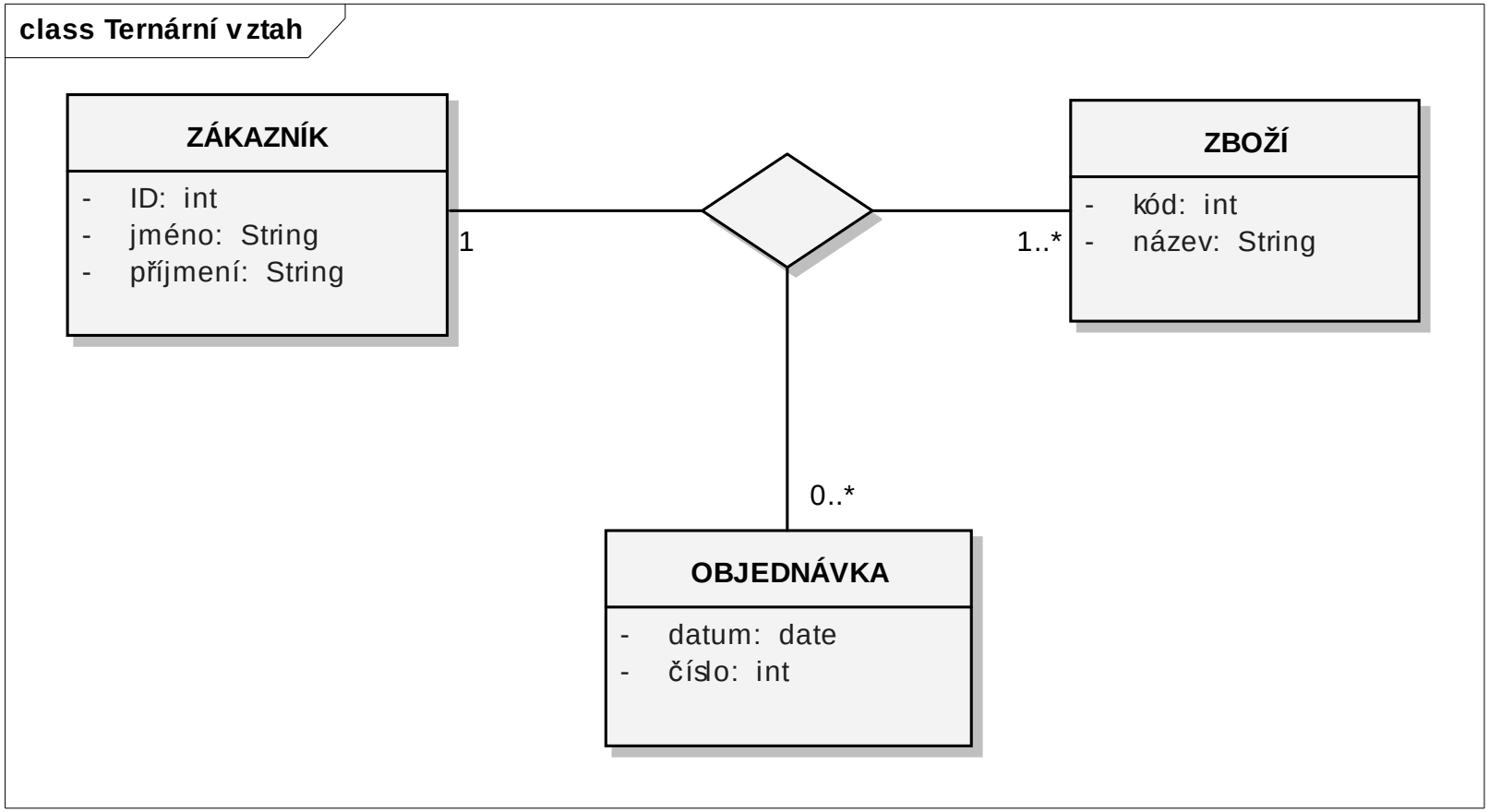
Pár poznámek – první model



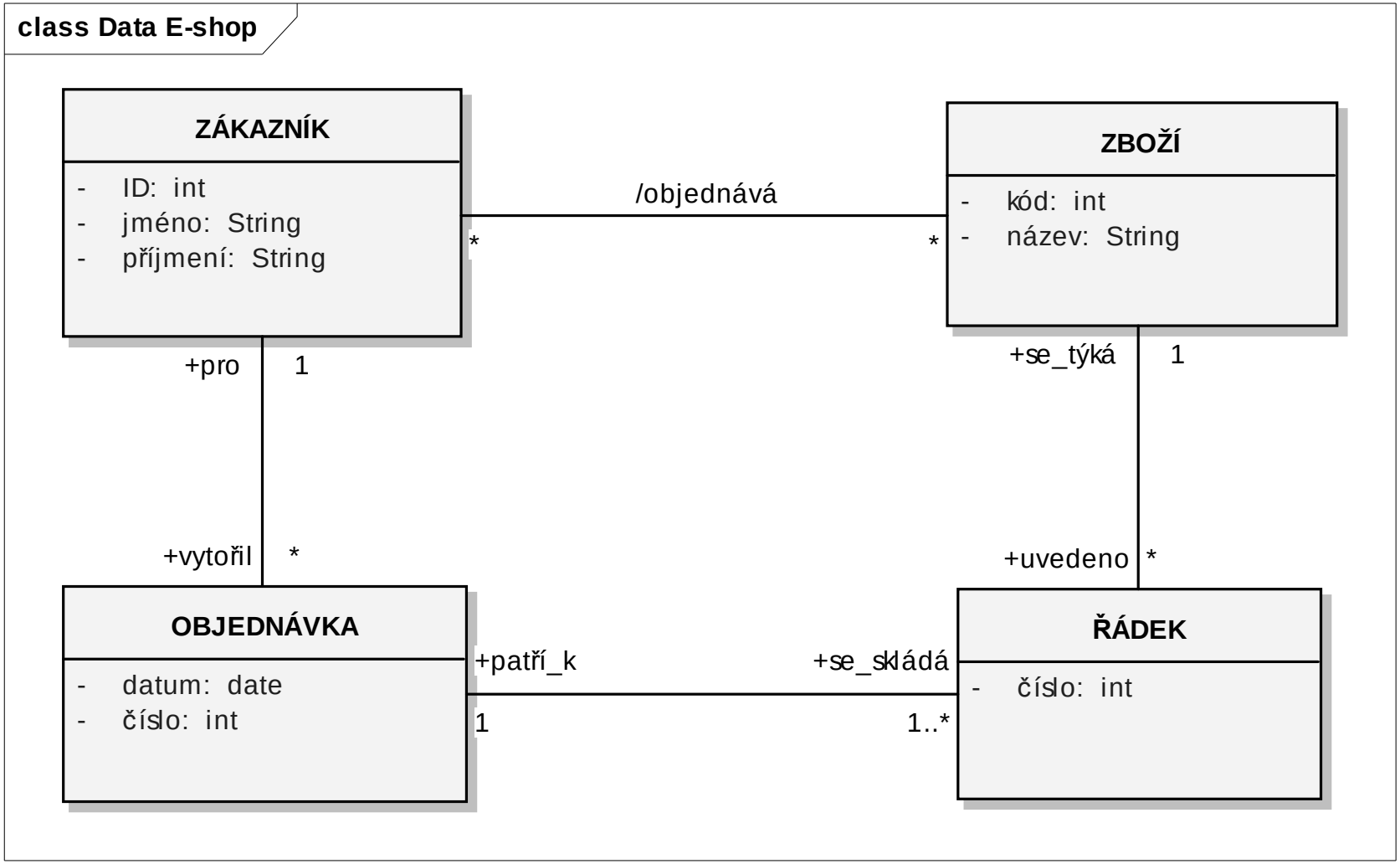
Přidružená třída



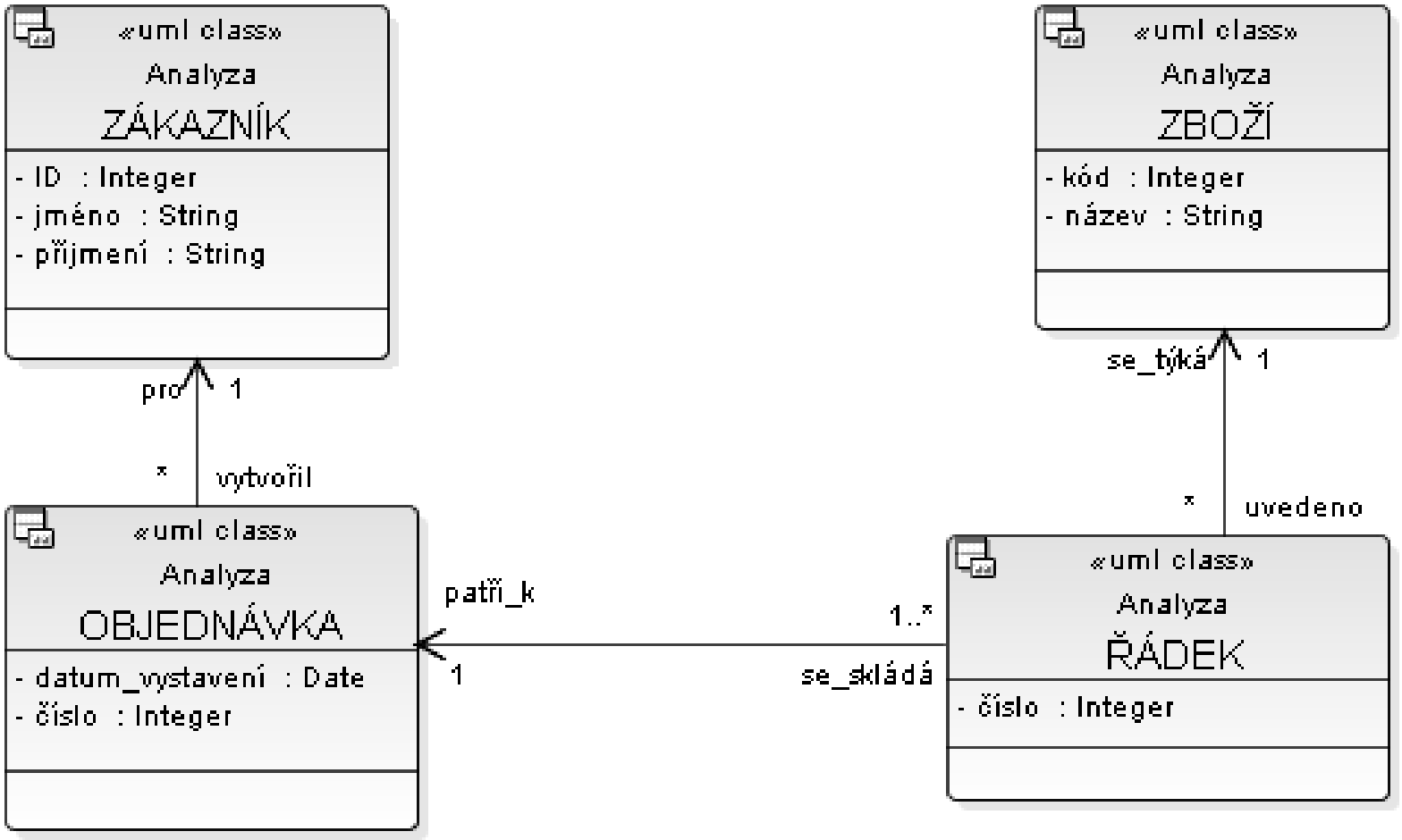
Nebo ternární vztah



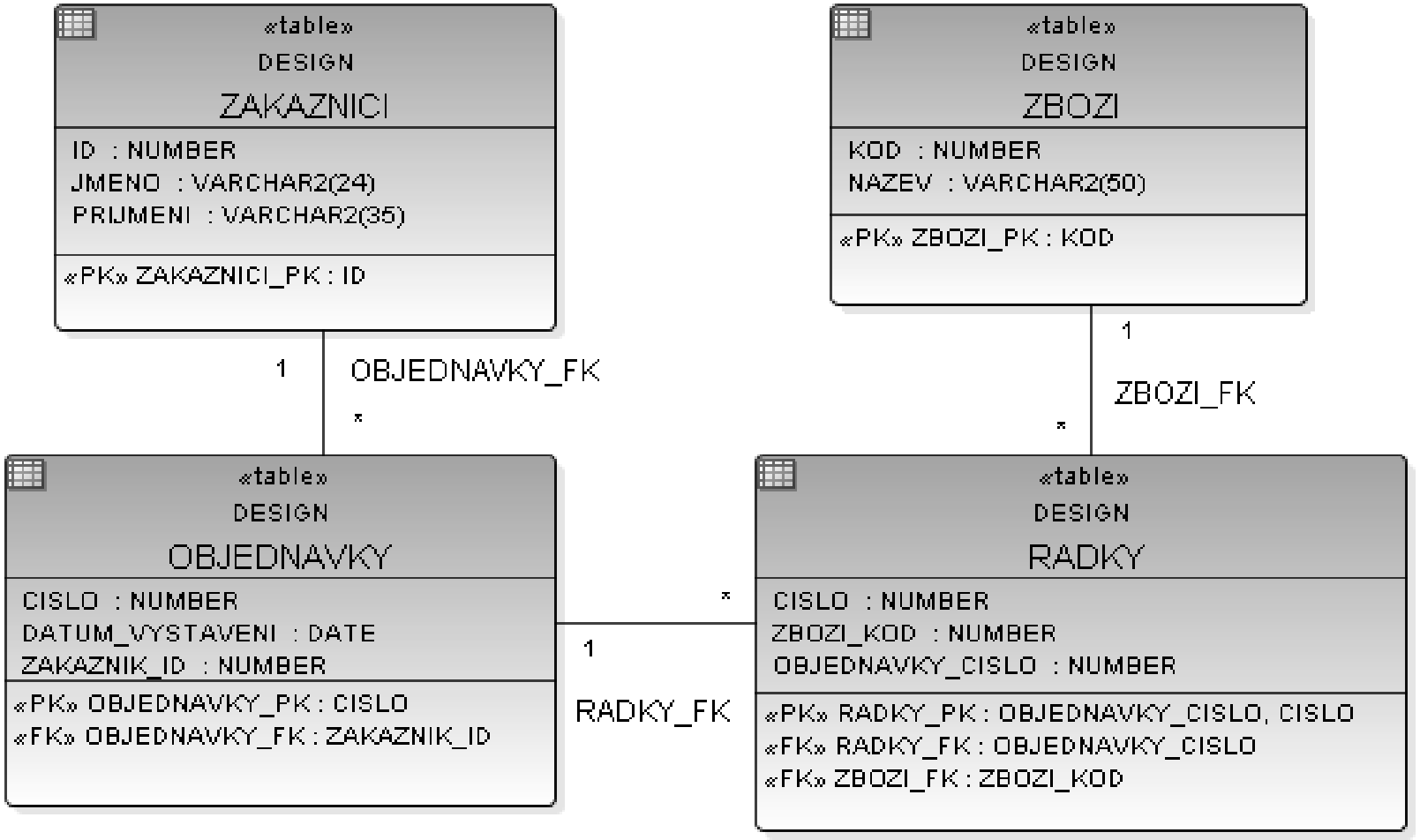
Podrobnější model



Používání šipek?



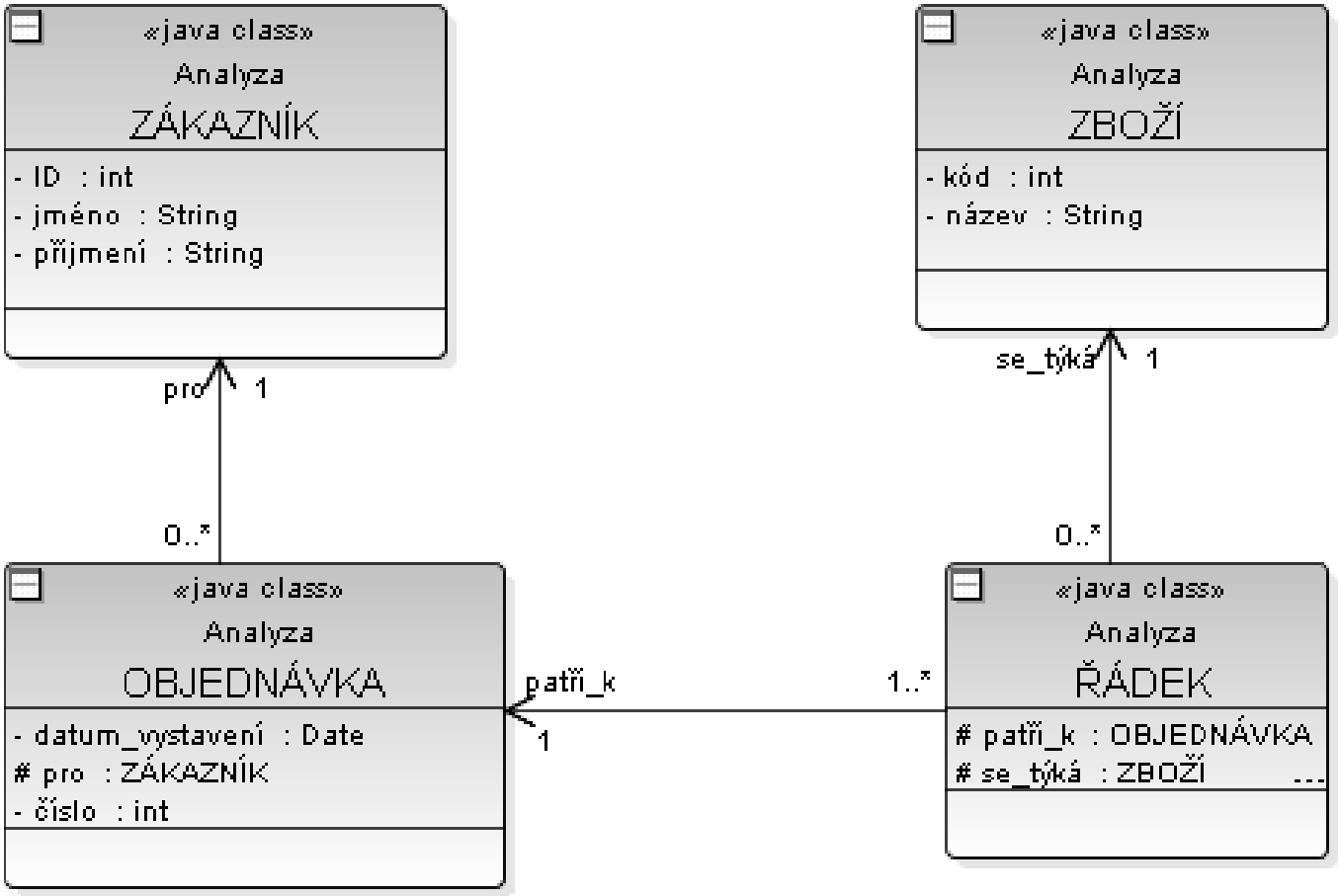
Generovaný logický model (SQL)



Generovaný kód

```
…
CREATE TABLE "RADKY„ (
  "CISLO" NUMBER NOT NULL,
  "ZBOZI_KOD" NUMBER NOT NULL,
  "OBJEDNAVKY_CISLO" NUMBER NOT NULL
);
ALTER TABLE "RADKY"
  ADD CONSTRAINT "RADKY_PK" PRIMARY KEY
    ("OBJEDNAVKY_CISLO","CISLO") ENABLE;
ALTER TABLE "RADKY"
  ADD CONSTRAINT "RADKY_FK" FOREIGN KEY
    ("OBJEDNAVKY_CISLO")
  REFERENCES "OBJEDNAVKY„ ("CISLO") ENABLE;
ALTER TABLE "RADKY"
  ADD CONSTRAINT "ZBOZI_FK" FOREIGN KEY ("ZBOZI_KOD")
  REFERENCES "ZBOZI„ ("KOD") ENABLE;
…
```

Generovaný logický model (Java)



Generovaný kód

```
…
package Analyza;

public class RADEK
{
  private int cislo;
  /* @label ObjednavkaZbozi
   * @clientCardinality 0..*
   */
  protected Analyza.ZBOZI se_tyka;
  /* @label obsahuje
   * @clientCardinality 1..*
   */
  protected Analyza.OBJEDNAVKA patri_k;
}
…
```

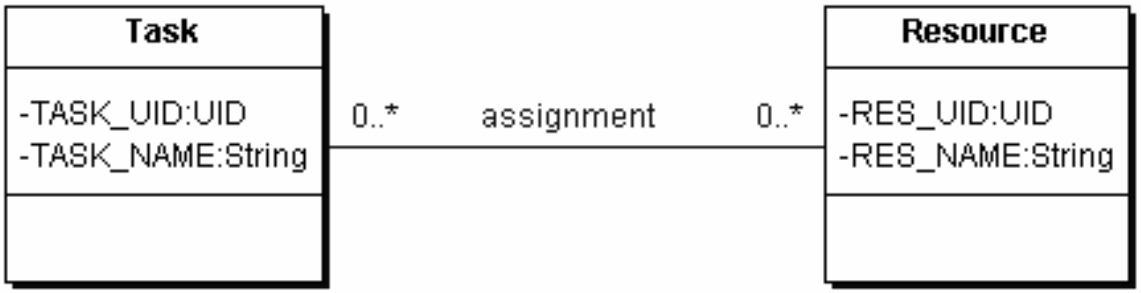
Příklad: MS Project

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

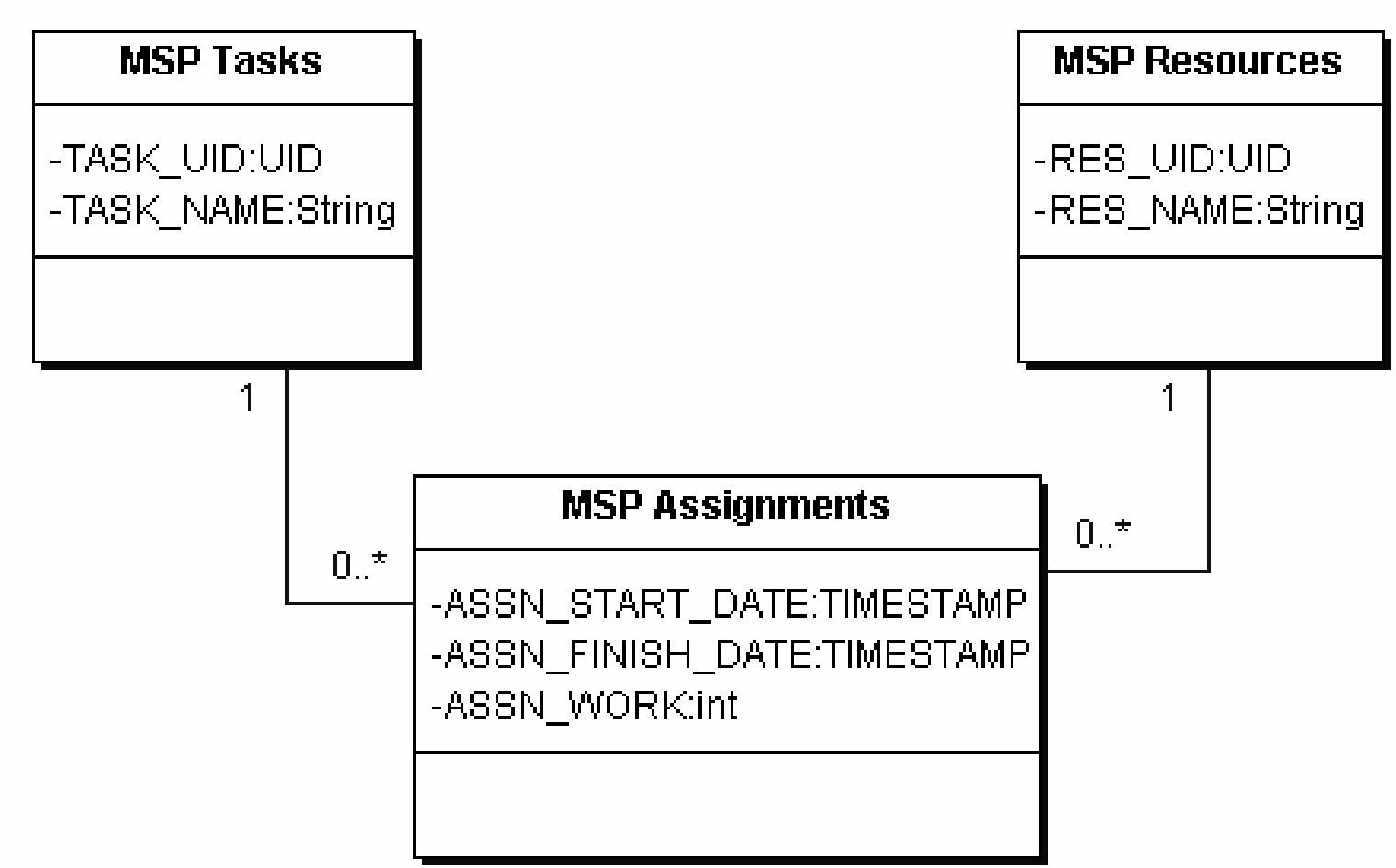
  Úloha

  Zdroj

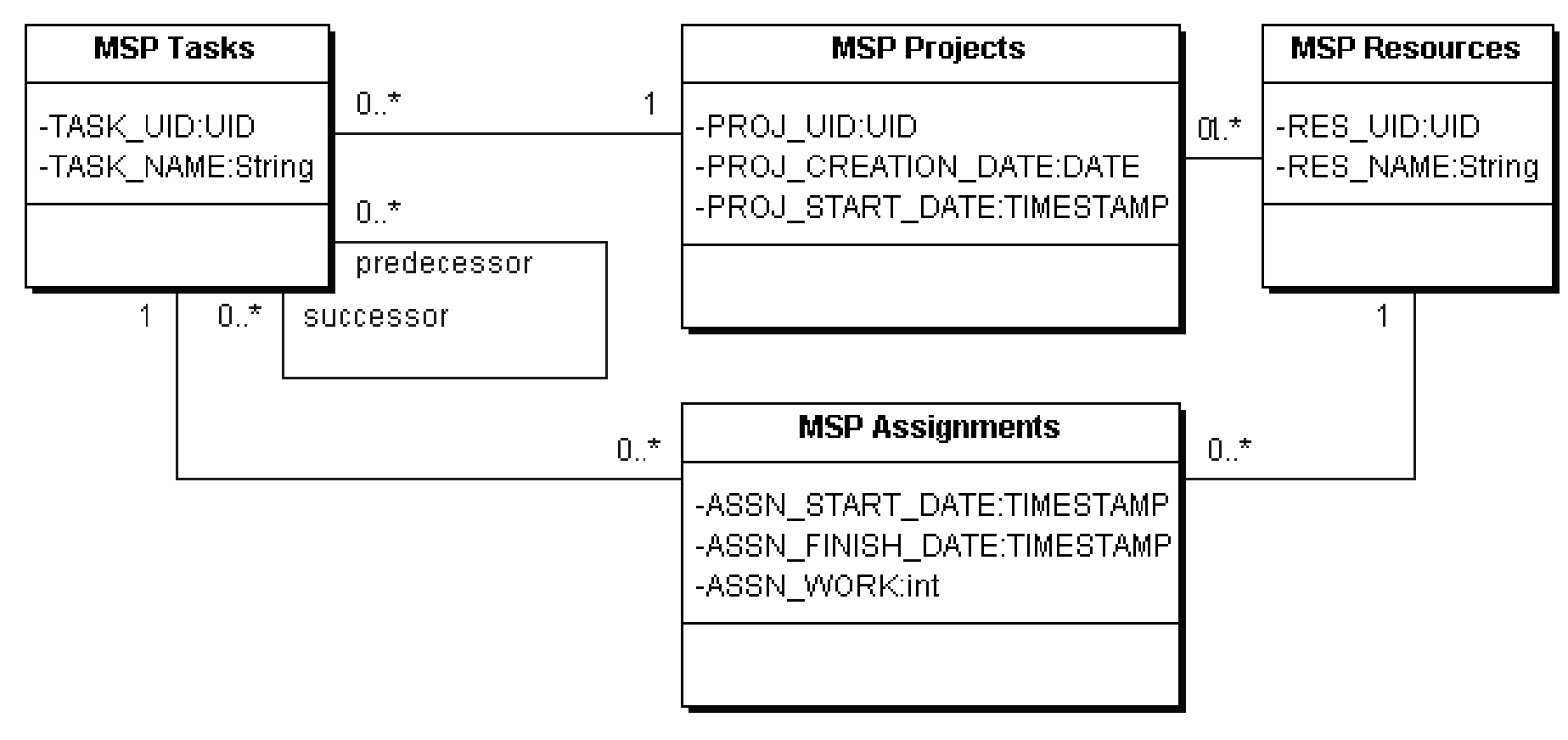
  Přiřazení



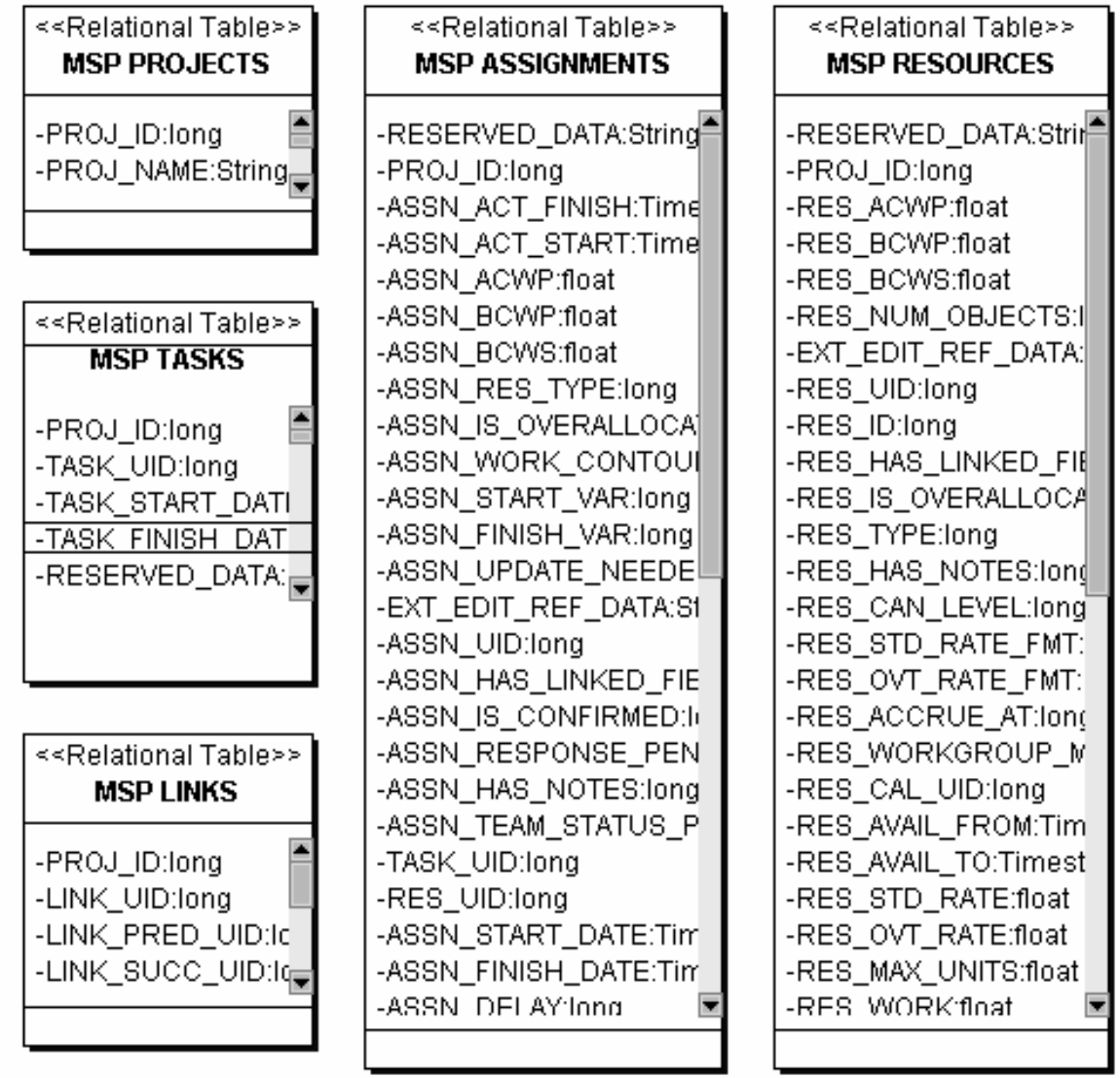
Podrobnější model



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) …
);
```

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?

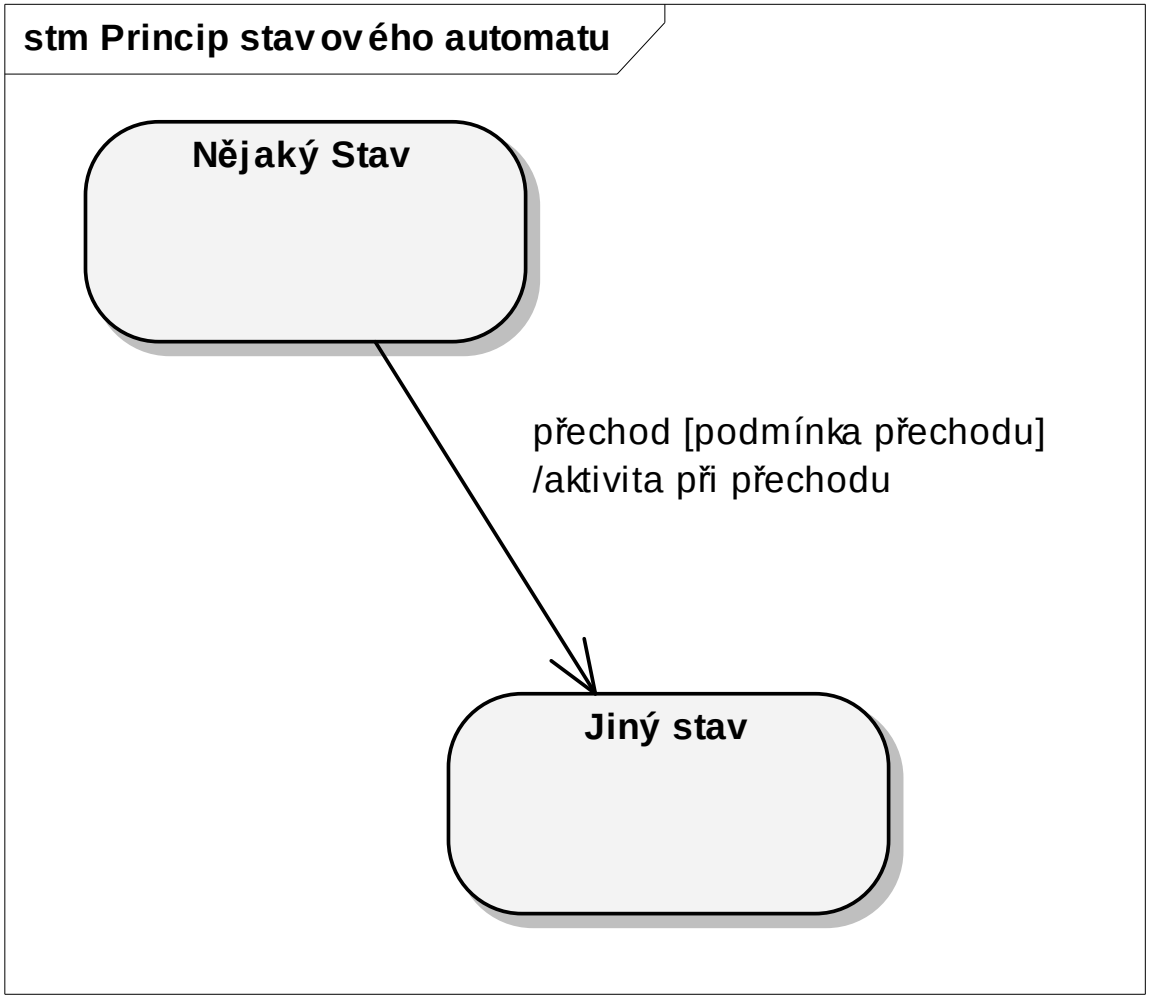
- Potřebujeme ještě v rámci nějaké funkce reprezentaci barelu zrušit.
- Mohla by to udělat funkce „dodávka“, neboť po vyskladnění barelu jeho životní cyklus končí.
- Doplníme tedy do popisu funkce dodávka požadavek „pokud v rámci dodávky využijeme některý barel, vymažeme jeho reprezentaci z obsahu skladu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

Dynamický model

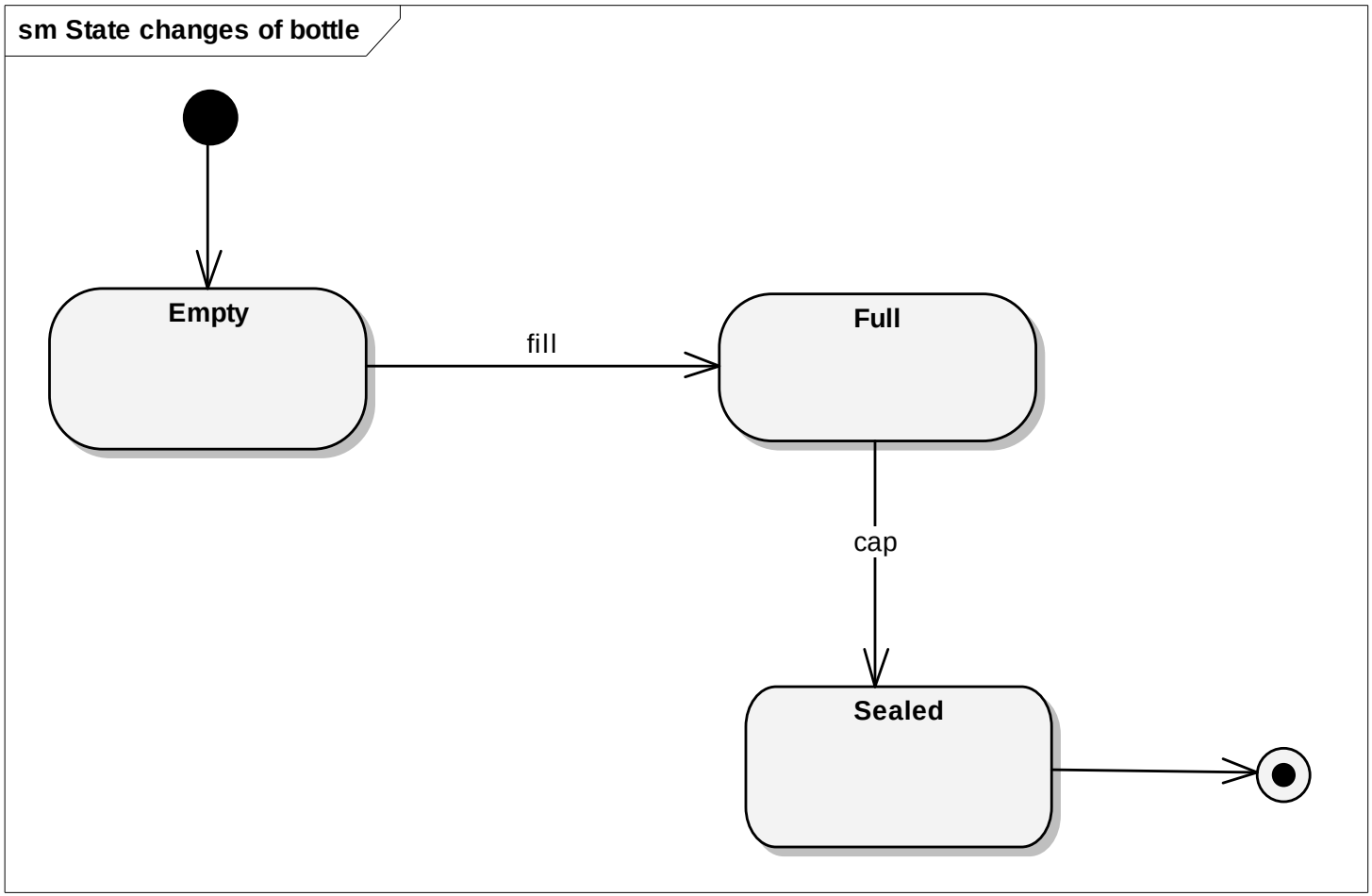
Stavové diagramy

- 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)

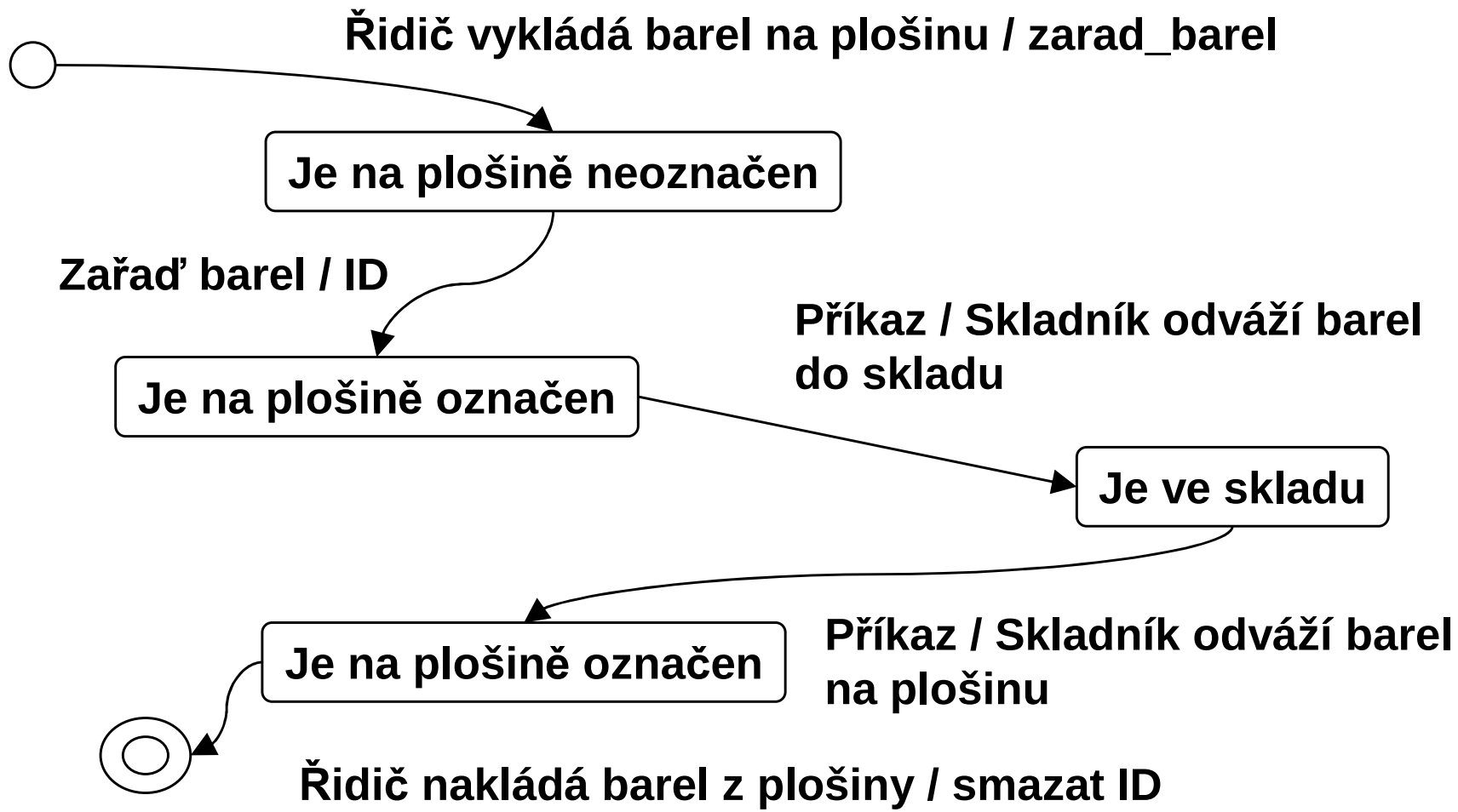
Princip stavového automatu



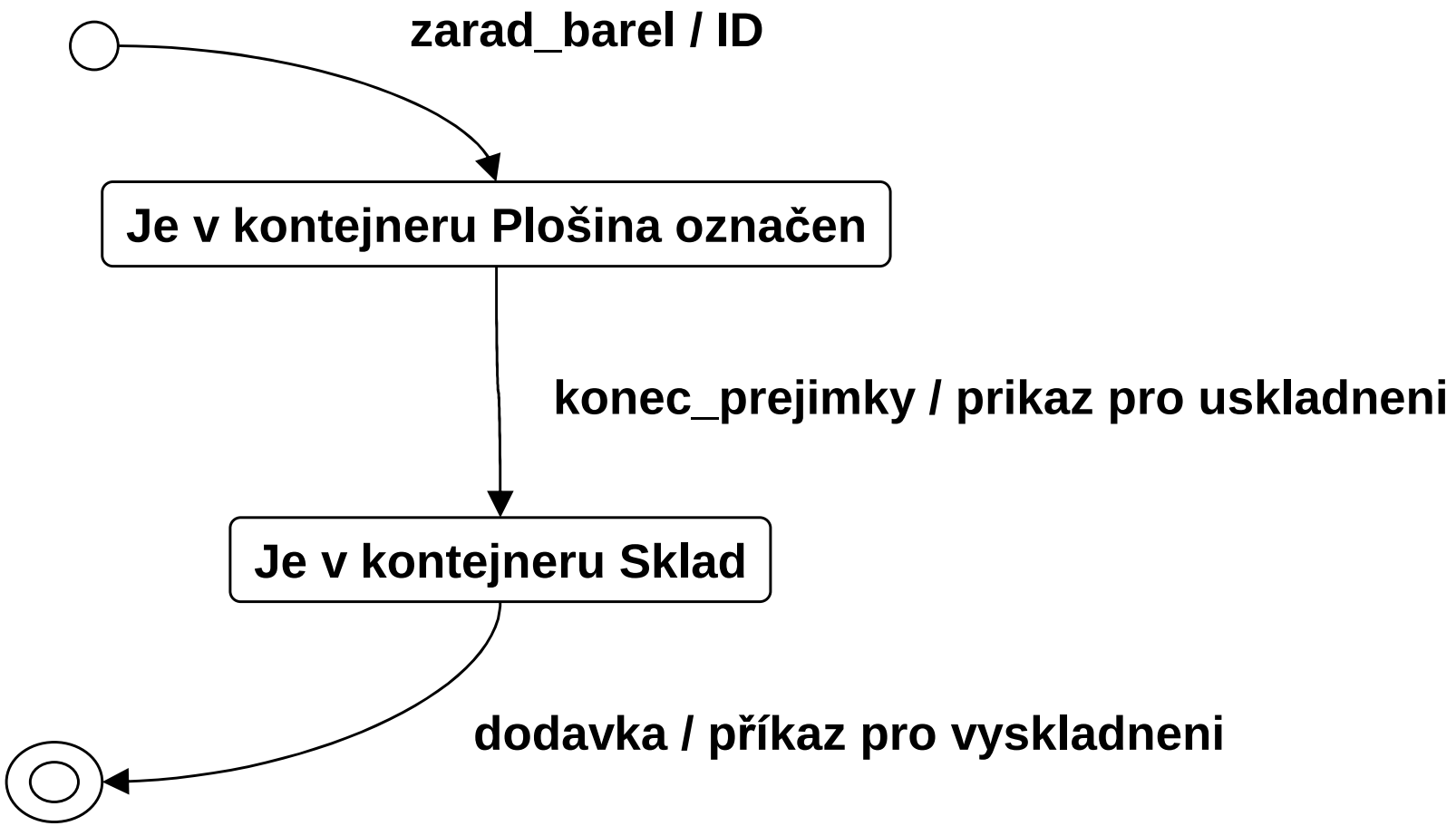
Životní cyklus jako stavový diag.



Životní cyklus skutečného „barelu”

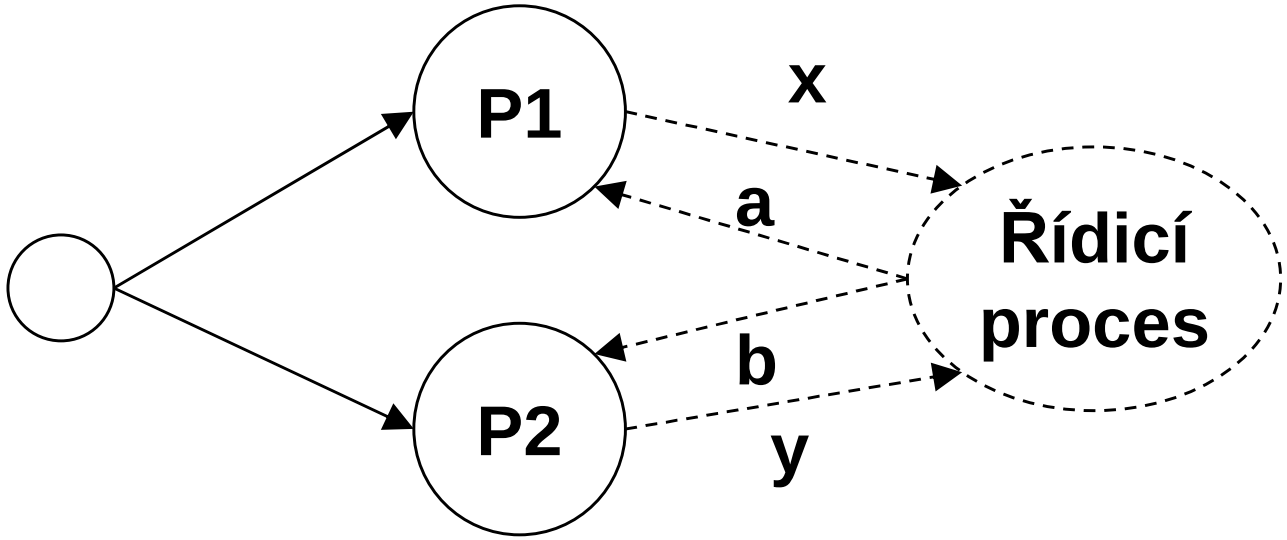


Životní cyklus entity „barel”

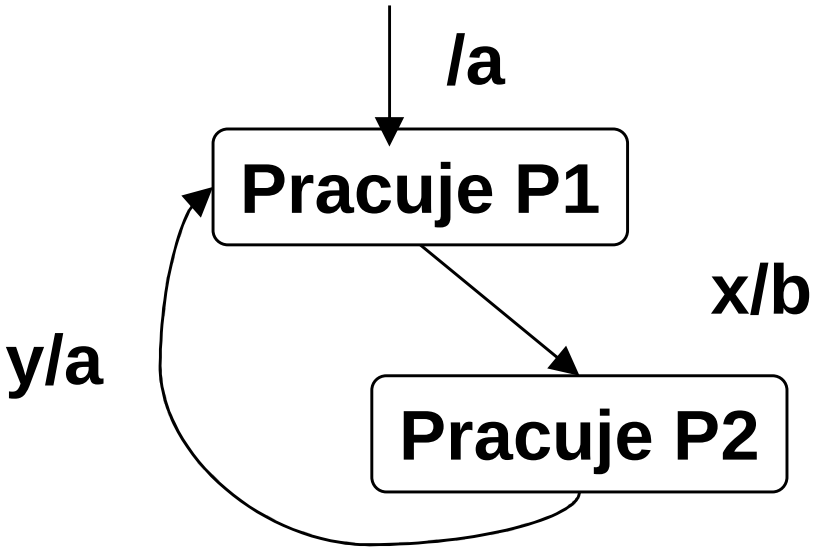


Popis řídicích procesů pomocí stavových diagramů

- 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.



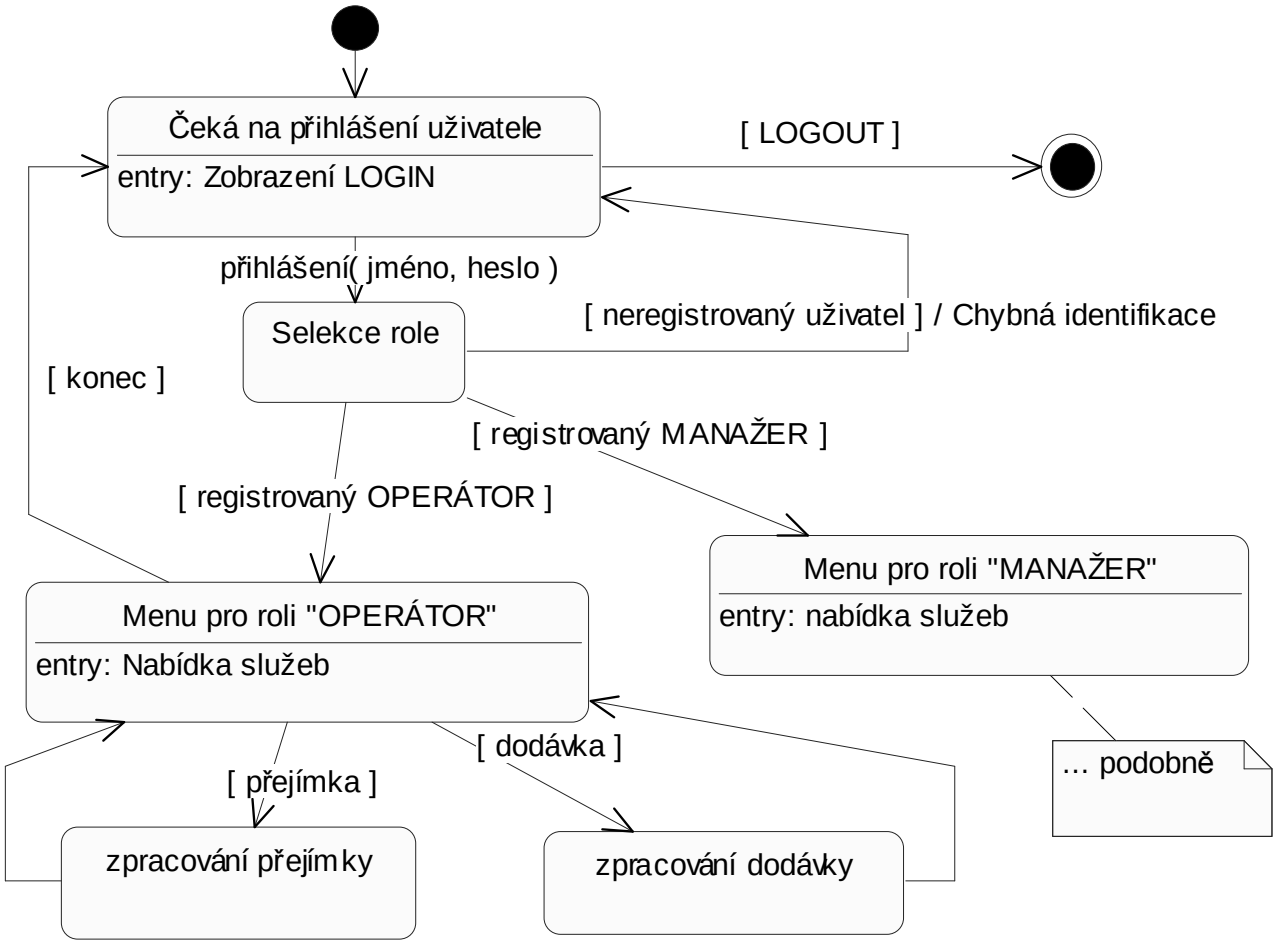
střídavě spouštíme P1 a P2



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 ECO-skladu



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?

Normalizace datového modelu

Předpoklad: všechny entity jsou jednoznačně identifikovatelné označenou kombinací atributů a/nebo vztahů

- 1.normální forma: entity neobsahují násobné atributy ani komponované atributy
- 2.normální forma (navíc): neklíčové atributy závisí pouze na celém klíči
- 3.normální forma (navíc): neklíčové atributy nejsou závislé na neklíčových položkách

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

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?

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 pro „Výtah”

Lifecycle Výtah:

(požadavek)* || (ON/OFF)*

požadavek = [požadavek na přivolání | požadavek na patro]

požadavek na přivolání = [požadavek na jízdu dolů | požadavek na jízdu nahoru]

požadavek na jízdu dolů = tlačítko pro jízdu dolů . #indikace

požadavek na jízdu nahoru = tlačítko pro jízdu nahoru . #indikace

požadavek na patro = tlačítko patra . #indikace

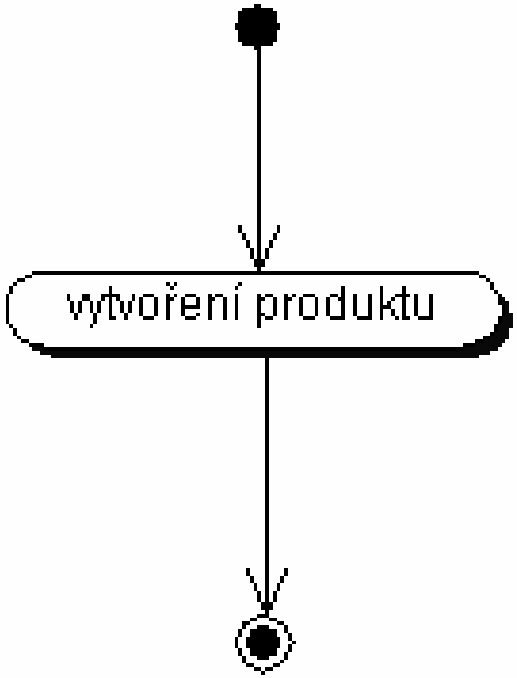
Poznámky k úvodní studii

Proč vytvářet úvodní studii?

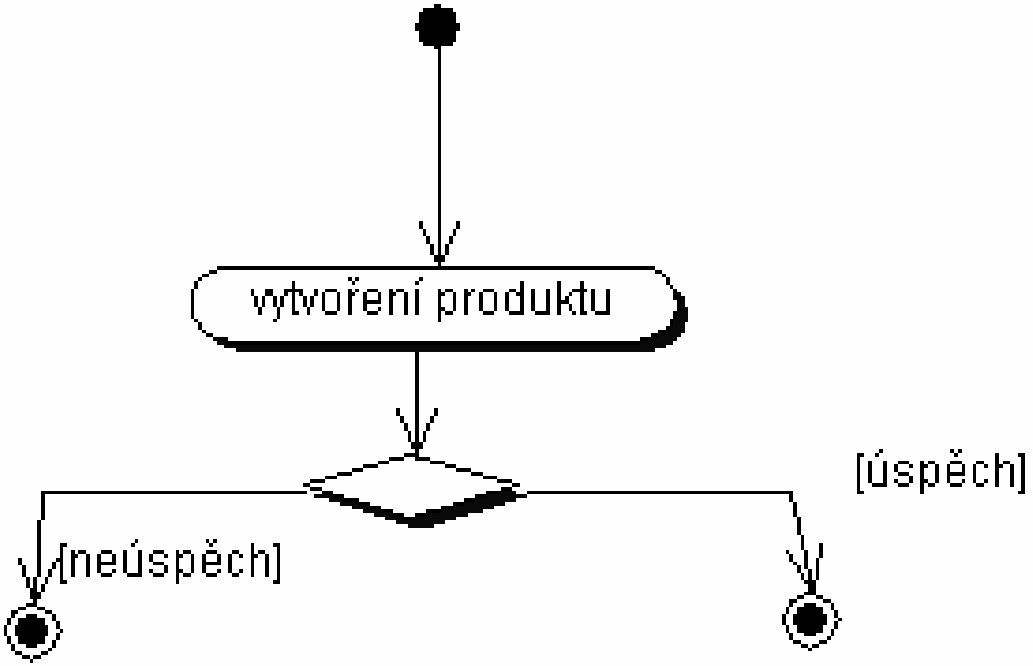
A. Protože to Richta chce.
B. Protože se to v komunitě informatiků sluší.
C. Protože to může ušetřit výdaje.

C je správně !!

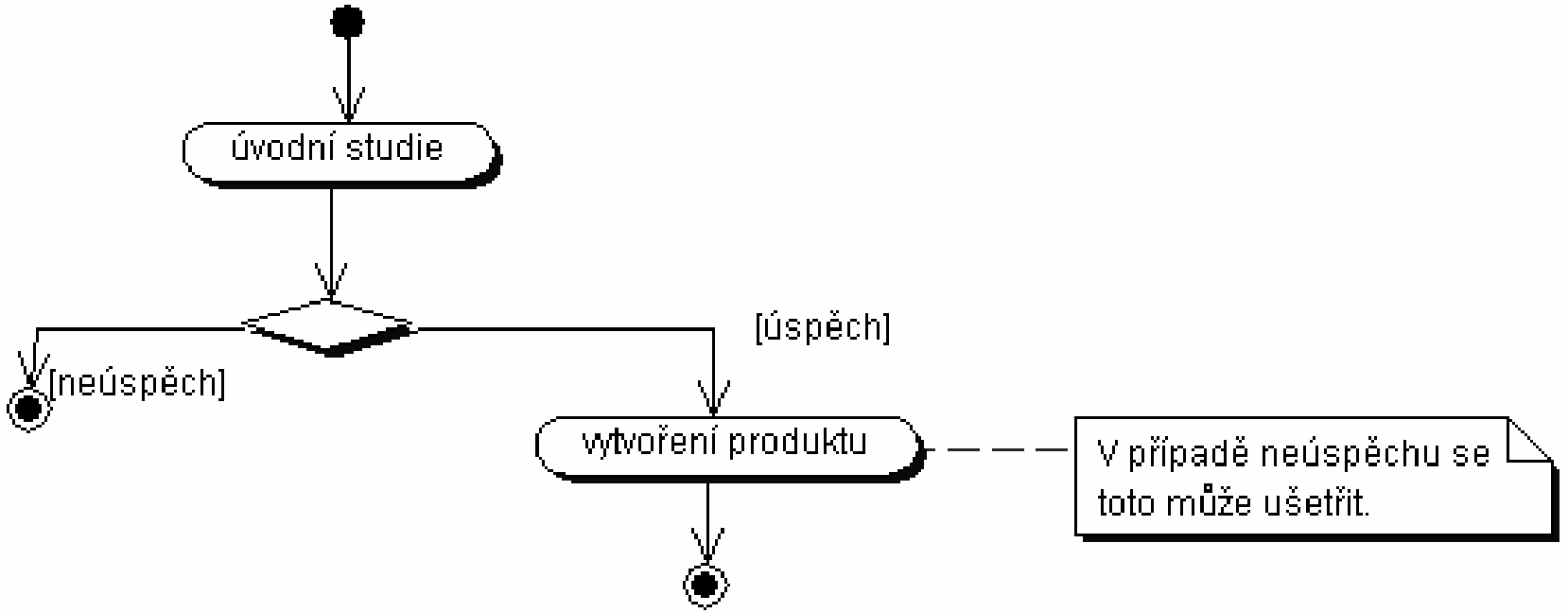
Jak to je ideální



Ale nemusí to vždy dopadnout



Úvodní studie může něco ušetřit



Jak se úvodní studie zakončí?

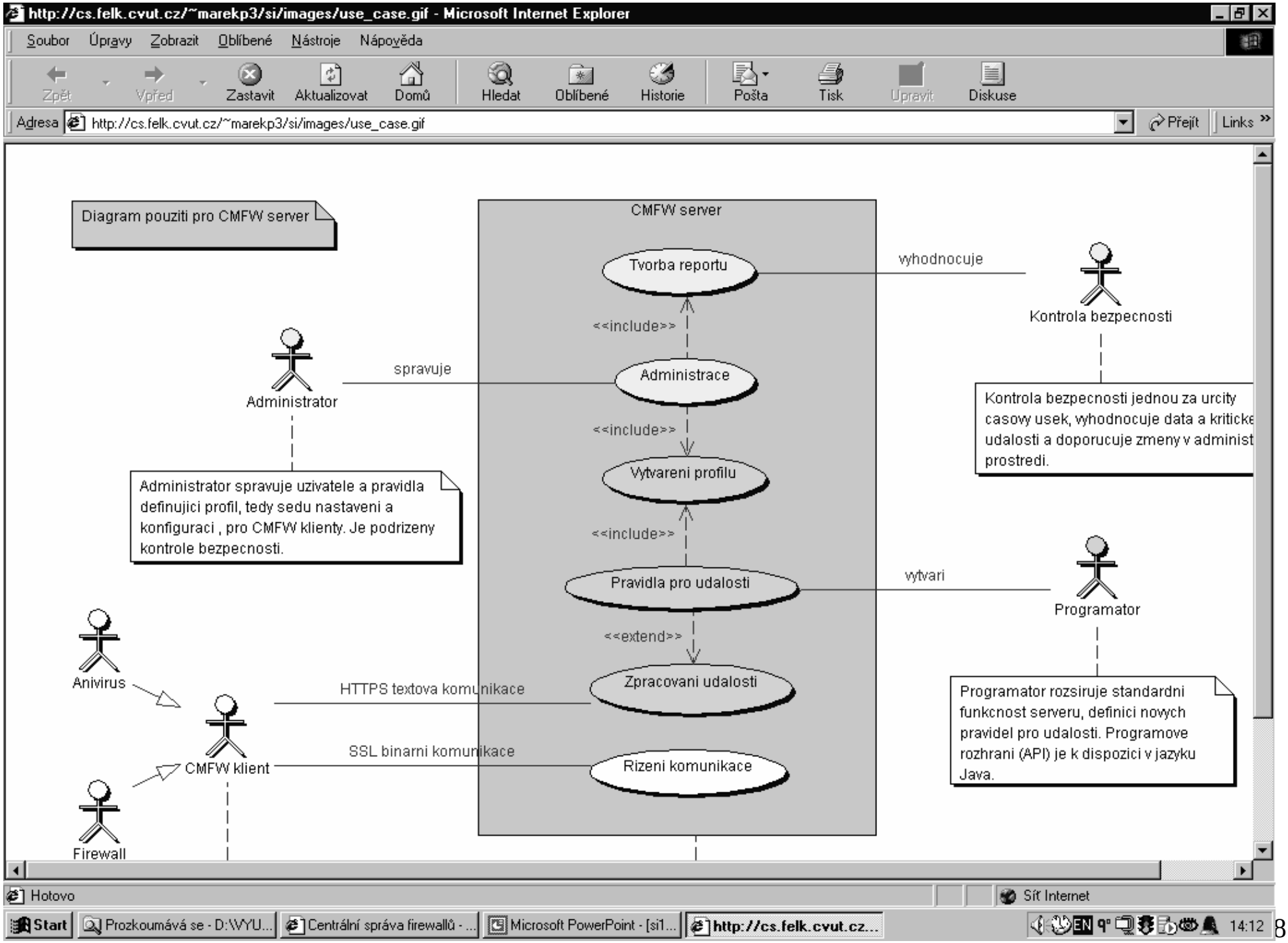
- Umístěním **dokumentace** na stránky projektu (deklarace záměru, odborný článek, model jednání/kontext, datový slovník, tým, kontakt, harmonogram (plán), matice zodpovědností, požadavky na HW a SW, rozpočet - 2x)
- Přípravou **prezentace** (umístí se rovněž na stránku projektu)
- Předvedením prezentace

Co to je prezentace úvodní studie?

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

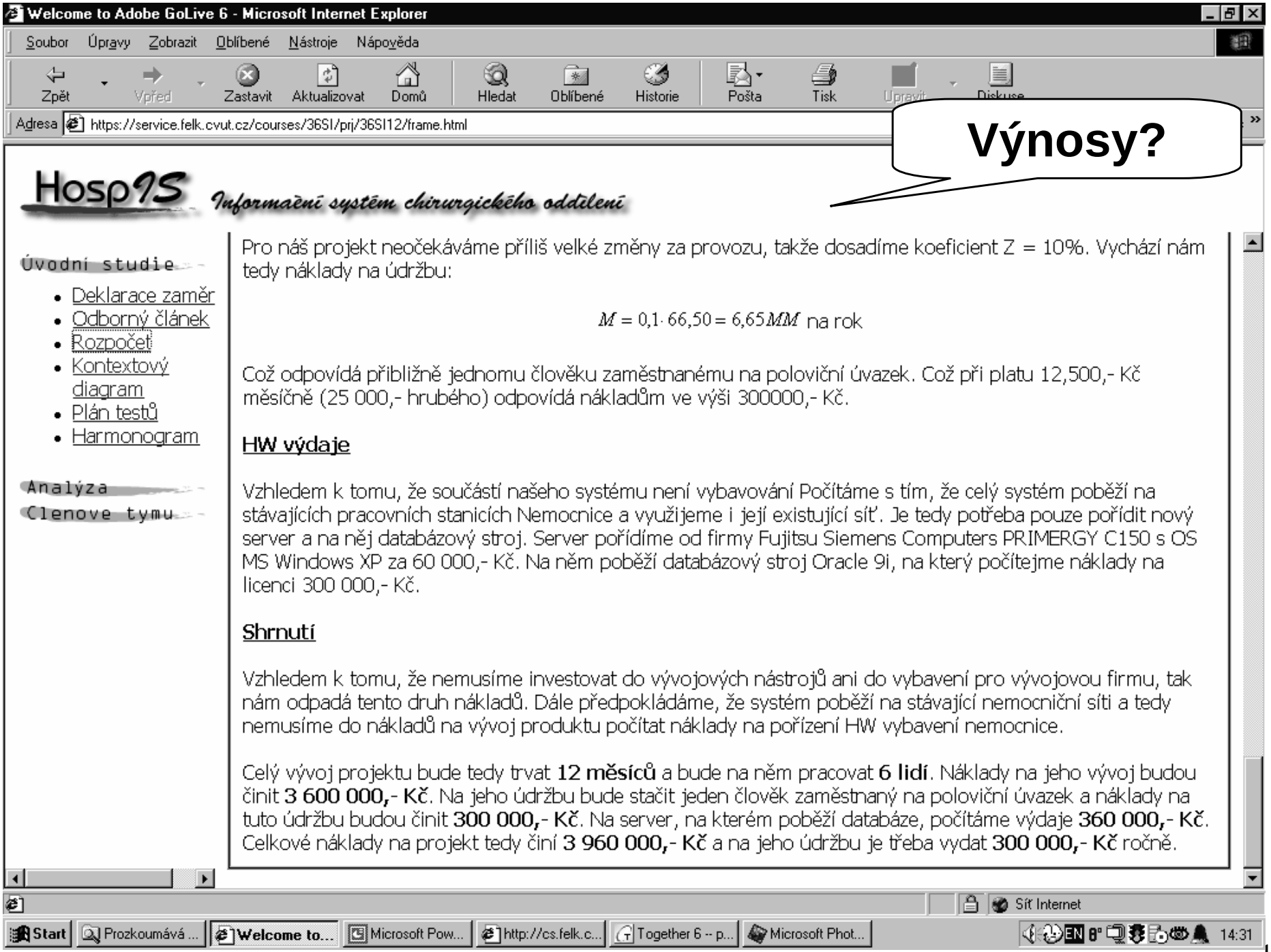
Stručná definice hranice systému

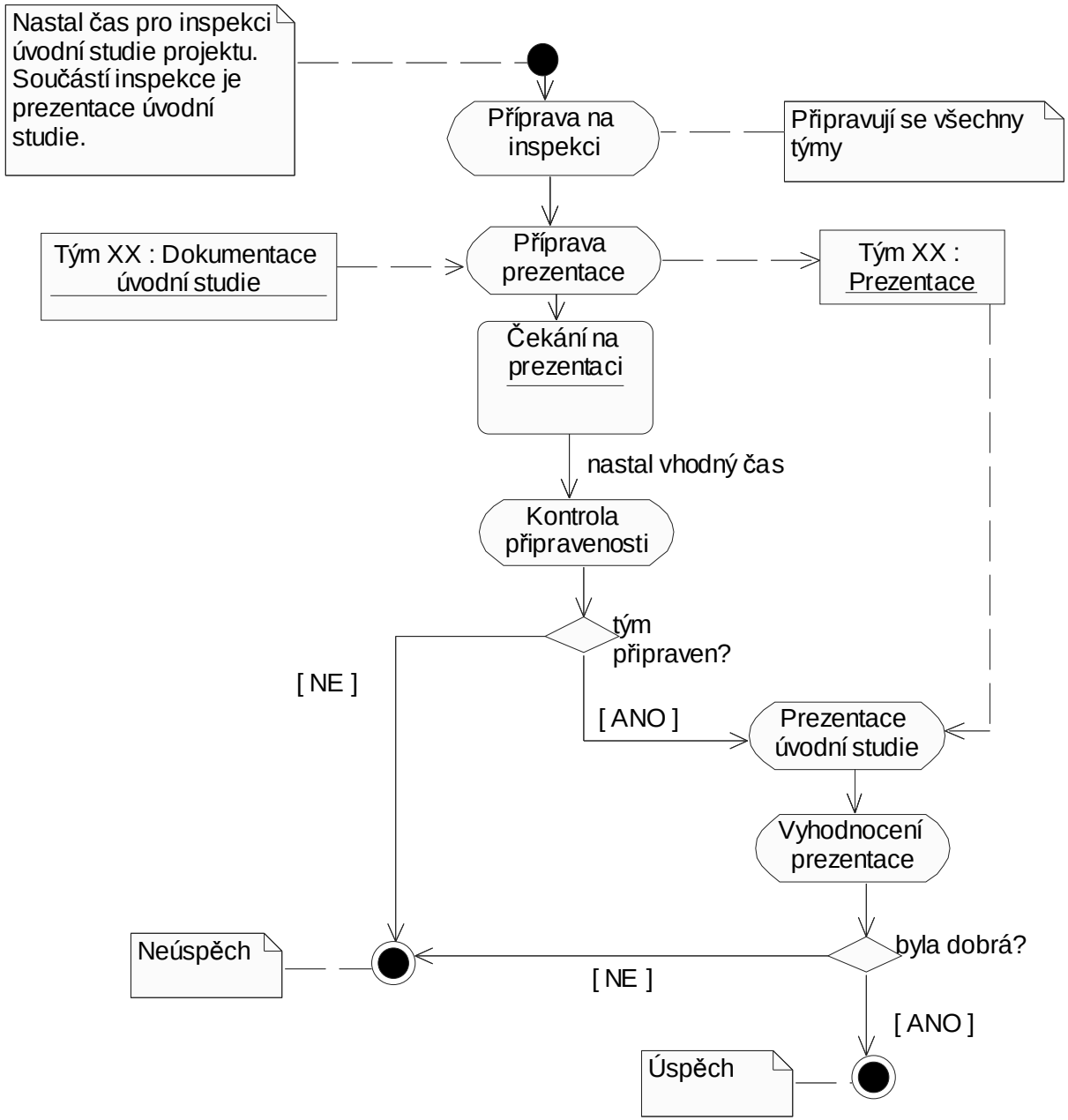
- Jen stručná informace, bez zbytečných podrobností (ty si každý může nalézt v úvodní studii, která je k dispozici na stránce projektu).


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>>
Administrace
spravuje
Administrator
<<include>>
Vytvareni profilu
Kontrola bezpecnosti jednou za urcity casovy usek, vyhodnocuje data a kritické udalosti a doporucuje zmeny v administ prostredi.
<<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
Zpracovani udalosti
HTTPS textova komunikace
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...
14:12

Prezentace úvodní studie slouží zejména pro zviditelnění projektu a jako podklad pro rozhodování o financování projektu

hranice produktu – náklady - výnosy


Welcome to Adobe GoLive 6 - 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/36SI12/frame.html
Výnosy?
HospIS
Informační systém chirurgického oddělení
Úvodní studie
Deklarace záměr
Odborný článek
Rozpočet
Kontextový diagram
Plán testů
Harmonogram
Analýza
Clenove tymu
Pro náš projekt neočekáváme příliš velké změny za provozu, takže dosadíme koeficient Z = 10%. Vychází nám tedy náklady na údržbu:
$M = 0{,}1 \cdot 66{,}50 = 6{,}65 MM$ na rok
Což odpovídá přibližně jednomu člověku zaměstnanému na poloviční úvazek. Což při platu 12,500,- Kč měsíčně (25 000,- hrubého) odpovídá nákladům ve výši 300000,- Kč.
HW výdaje
Vzhledem k tomu, že součástí našeho systému není vybavování Počítáme s tím, že celý systém poběží na stávajících pracovních stanicích Nemocnice a využijeme i její existující síť. Je tedy potřeba pouze pořídit nový server a na něj databázový stroj. Server pořídíme od firmy Fujitsu Siemens Computers PRIMERGY C150 s OS MS Windows XP za 60 000,- Kč. Na něm poběží databázový stroj Oracle 9i, na který počítejme náklady na licenci 300 000,- Kč.
Shrnutí
Vzhledem k tomu, že nemusíme investovat do vývojových nástrojů ani do vybavení pro vývojovou firmu, tak nám odpadá tento druh nákladů. Dále předpokládáme, že systém poběží na stávající nemocniční síti a tedy nemusíme do nákladů na vývoj produktu počítat náklady na pořízení HW vybavení nemocnice.
Celý vývoj projektu bude tedy trvat 12 měsíců a bude na něm pracovat 6 lidí. Náklady na jeho vývoj budou činit 3 600 000,- Kč. Na jeho údržbu bude stačit jeden člověk zaměstnaný na poloviční úvazek a náklady na tuto údržbu budou činit 300 000,- Kč. Na server, na kterém poběží databáze, počítáme výdaje 360 000,- Kč. Celkové náklady na projekt tedy činí 3 960 000,- Kč a na jeho údržbu je třeba vydat 300 000,- Kč ročně.
Síť Internet
Start Prozkoumává ... Welcome to... Microsoft Pow... http://cs.felk.c... Together 6 -- p... Microsoft Phot... 14:31



Prezentace úvodní studie

Kontrolní body pro úvod.studii

- Má projekt **kvalitní stránku**? Jsou zde uvedeni řešitelé a jejich e-maily? Je zde možnost poslat zprávu všem? Je na stránkách projektu uvedeno logo a název projektu? Je na stránkách projektu projektový deník?
- Je zpracována **deklarace záměru**?
- Je na stránkách projektu **odborný článek**?
- Existuje **katalog požadavků**? Kvalita funkčních požadavků, kvalita nefunkčních požadavků?
- Seznam a popis **případů užití** (USE-CASE), detailně rozpracované USE-CASE včetně scénářů?
- Je na stránkách k dispozici **harmonogram** práce týmu? Je na stránkách projektu **matice zodpovědnosti**? Jsou v úvodní studii specifikovány požadavky na HW a SW? Je v úvodní studii uveden rozpočet projektu dekomposicí (MS Project)? Je v úvodní studii uveden rozpočet projektu výpočtem (COCOMO)?
- Seznam a **popis entit systému** (= slovníček pojmů)? **Model entit systému** (= diagram)?
- Je k dispozici **prezentace** úvodní studie projektu?

Příklad

Prezentace úvodní studie projektu SPU

The End

Návrh

JAK se to udělá

Návaznosti

- Dříve:
  - Analýza
- Dnes:
  - Návrh
- Příště:
  - Návrhové vzory

Kroky návrhu

- návrh architektury systému
- návrh uživatelského vzhledu
- návrh komponent
- návrh komunikace mezi komponentami
- návrh způsobu integrace komponent a testování celku

Základní technologická rozhodnutí ve fázi návrhu

- architektura systému
- datové zdroje, přístupové mechanismy k nim
- distribuce programových modulů, komunikační mechanismy
- typy a formy výstupů
- uživatelské rozhraní
- vývojové prostředí

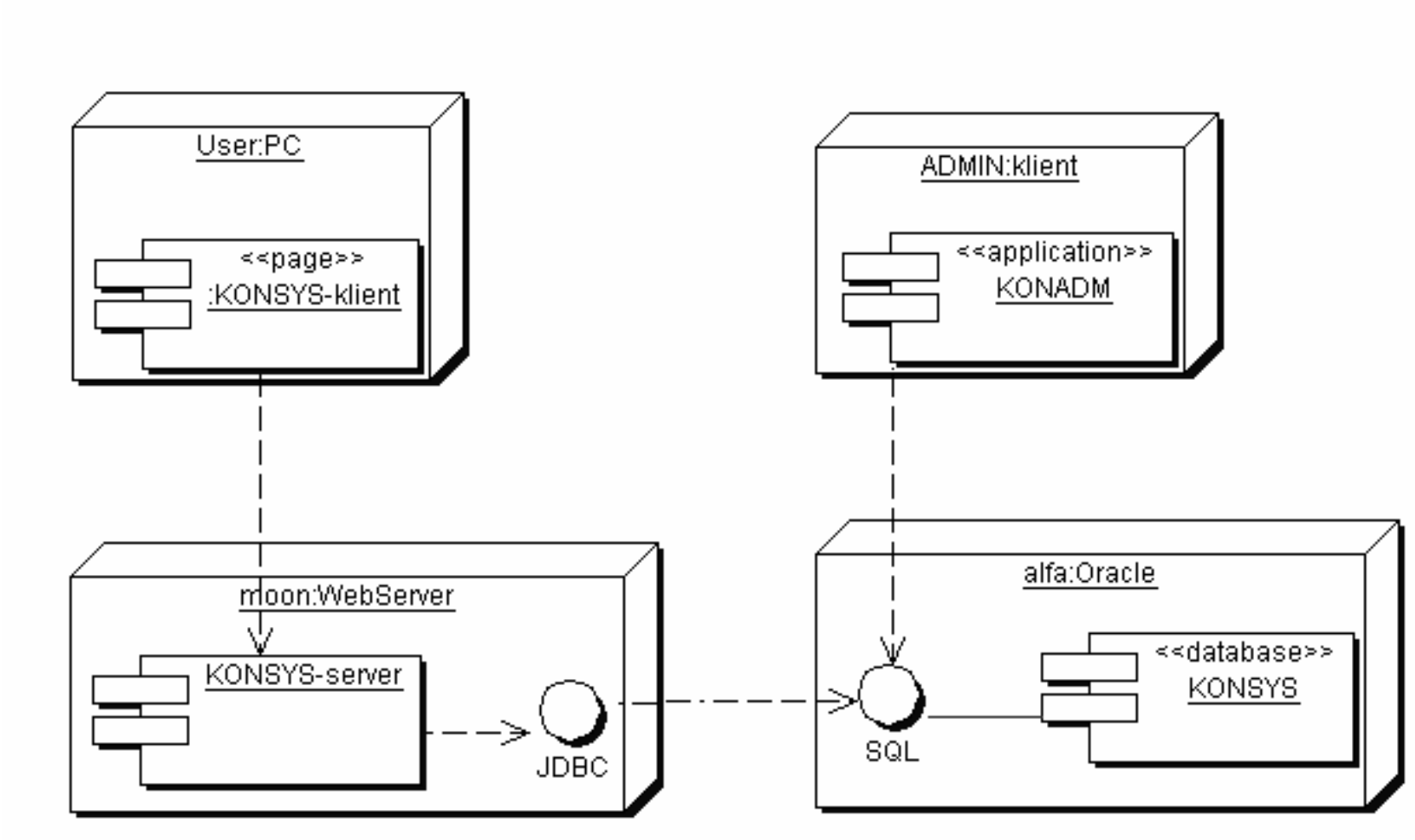
Vstupy pro návrh

- Analytický model chování systému

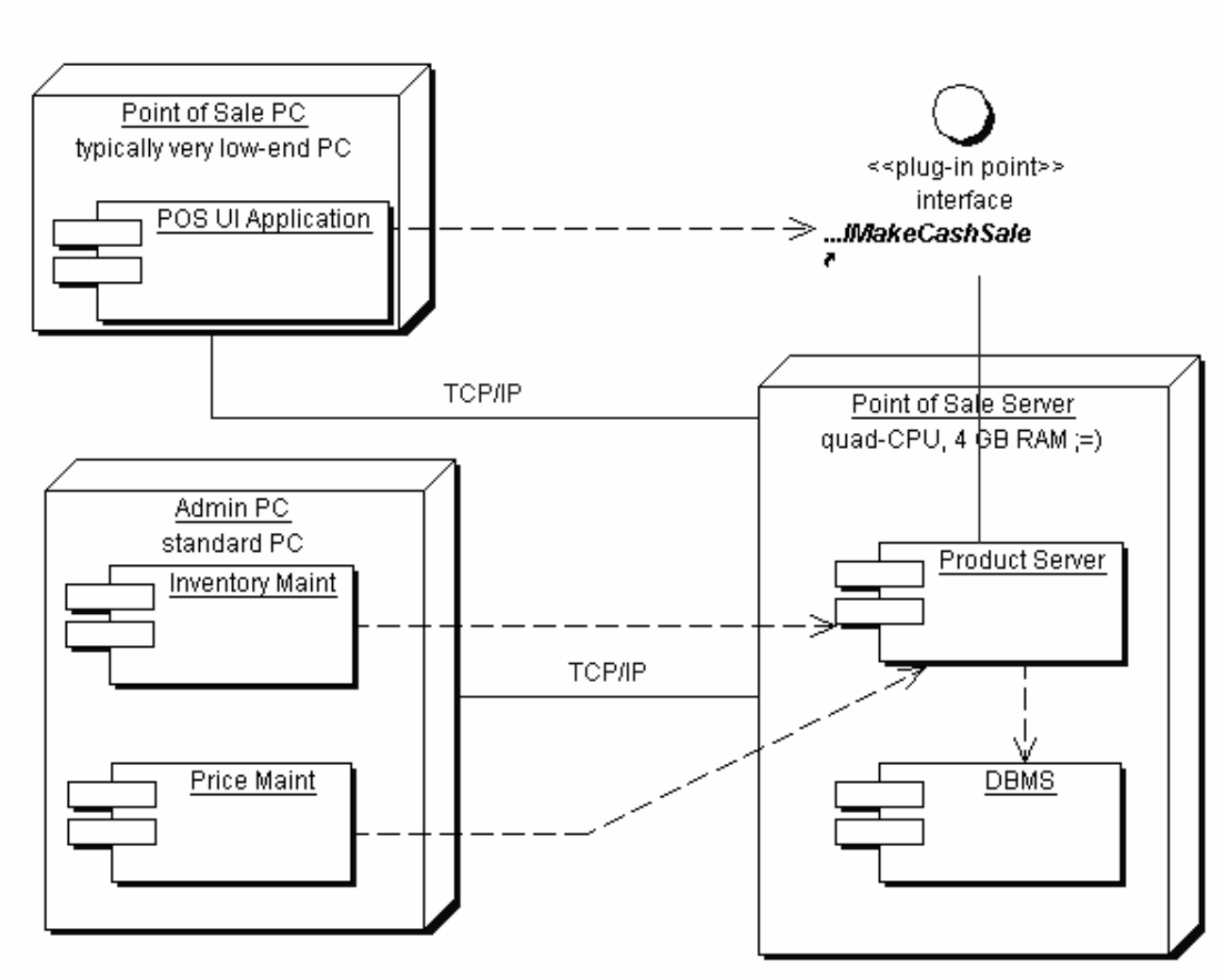
Výstupní dokumenty návrhu

- Architektura systému (HW,SW)
- Popis implementace dat (logický datový model)
- Popis komponent (modulů)
- Projektová dokumentace návrhu

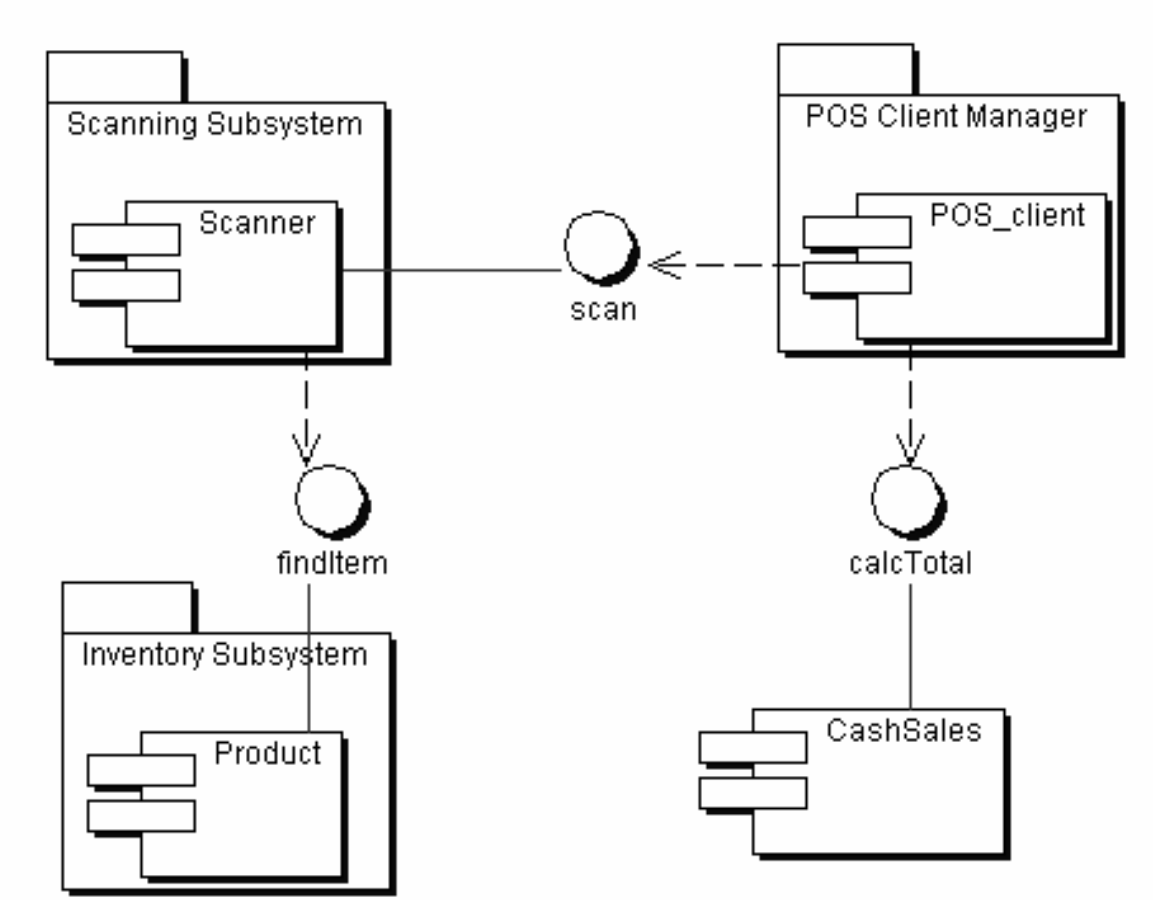
Příklad návrhu architektury



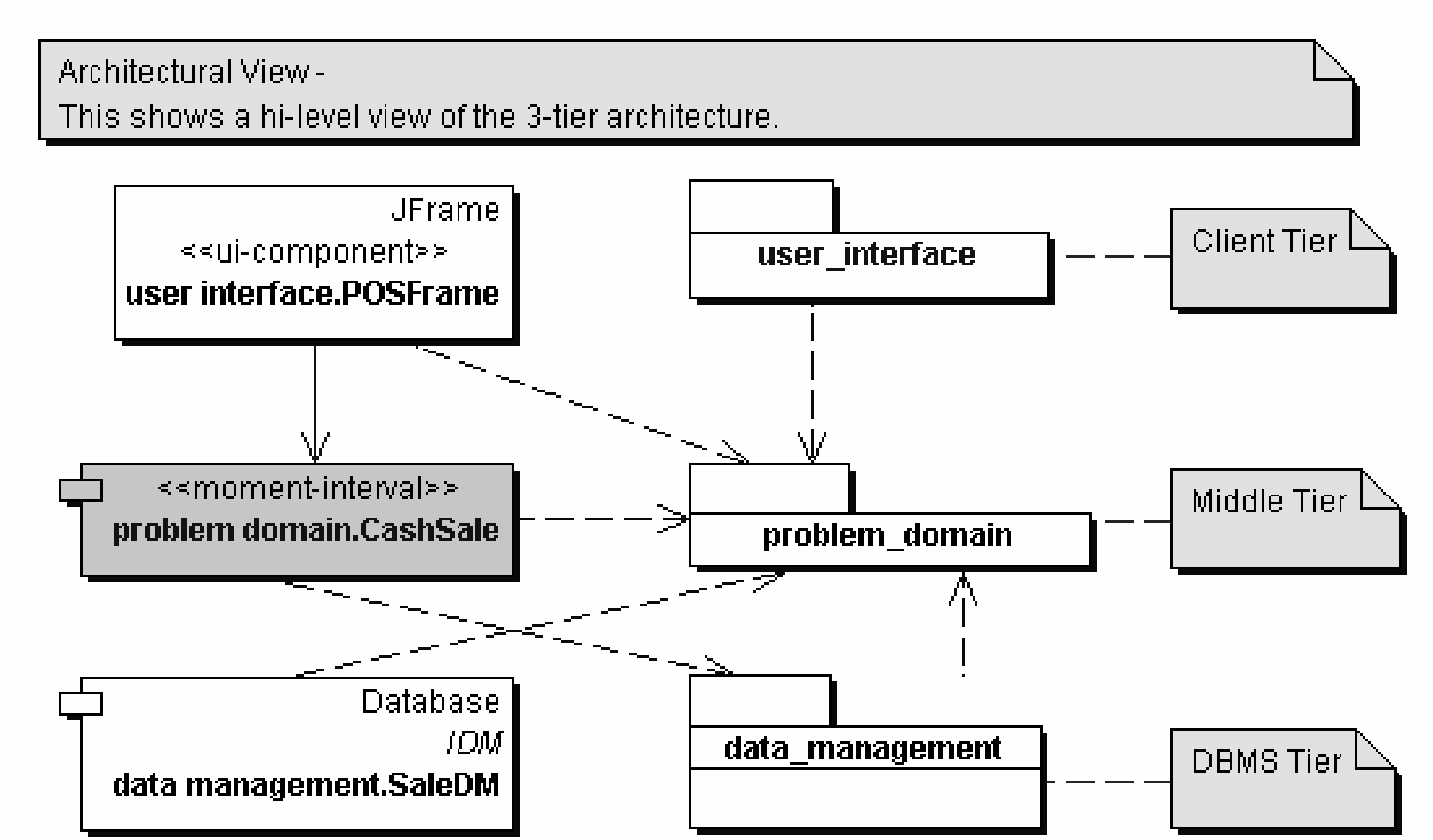
Jiný příklad: Elektronická pokladna



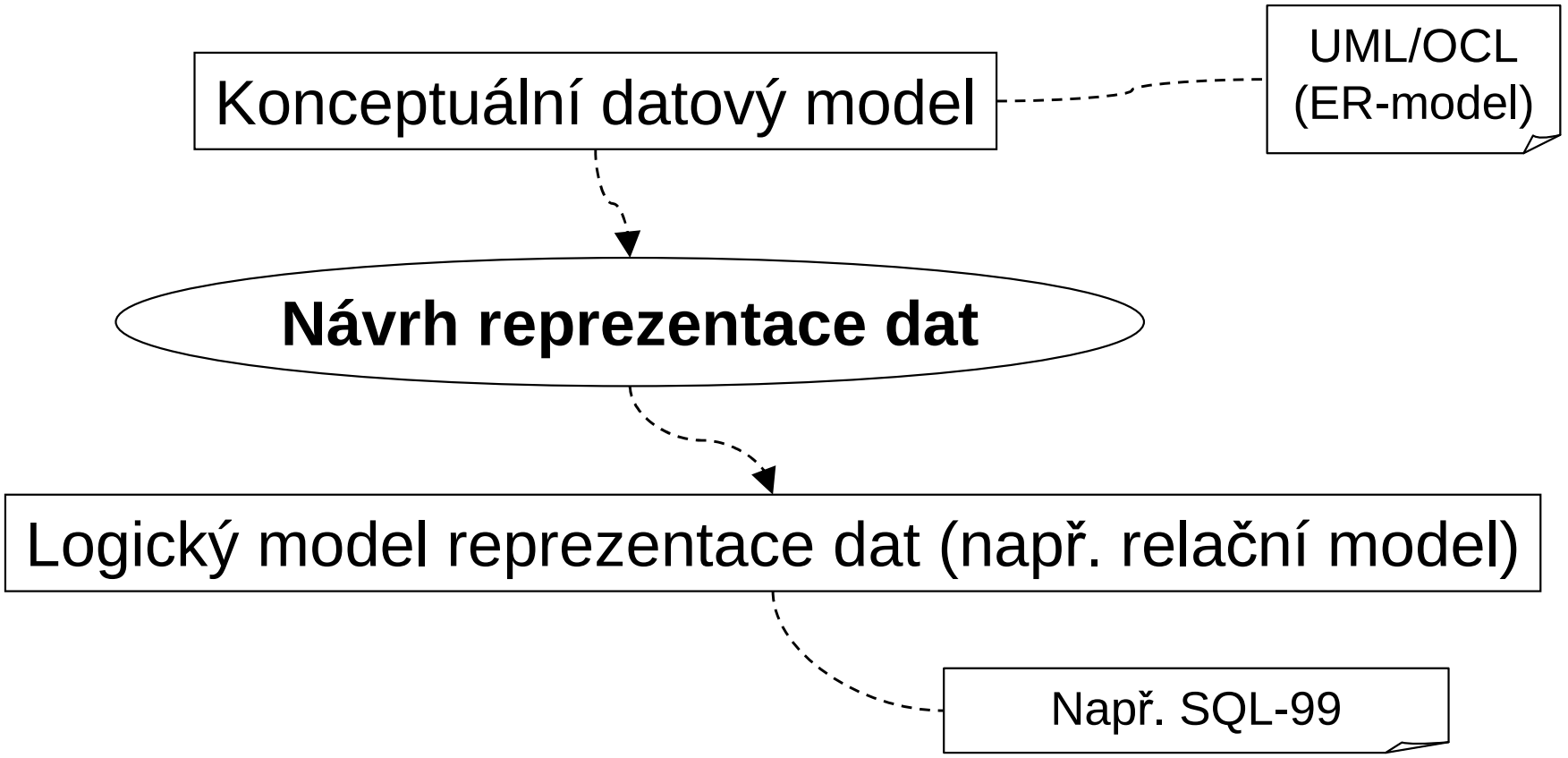
Elektronická pokladna (komponenty)



Pokladna (architektura)



Návrh reprezentace dat

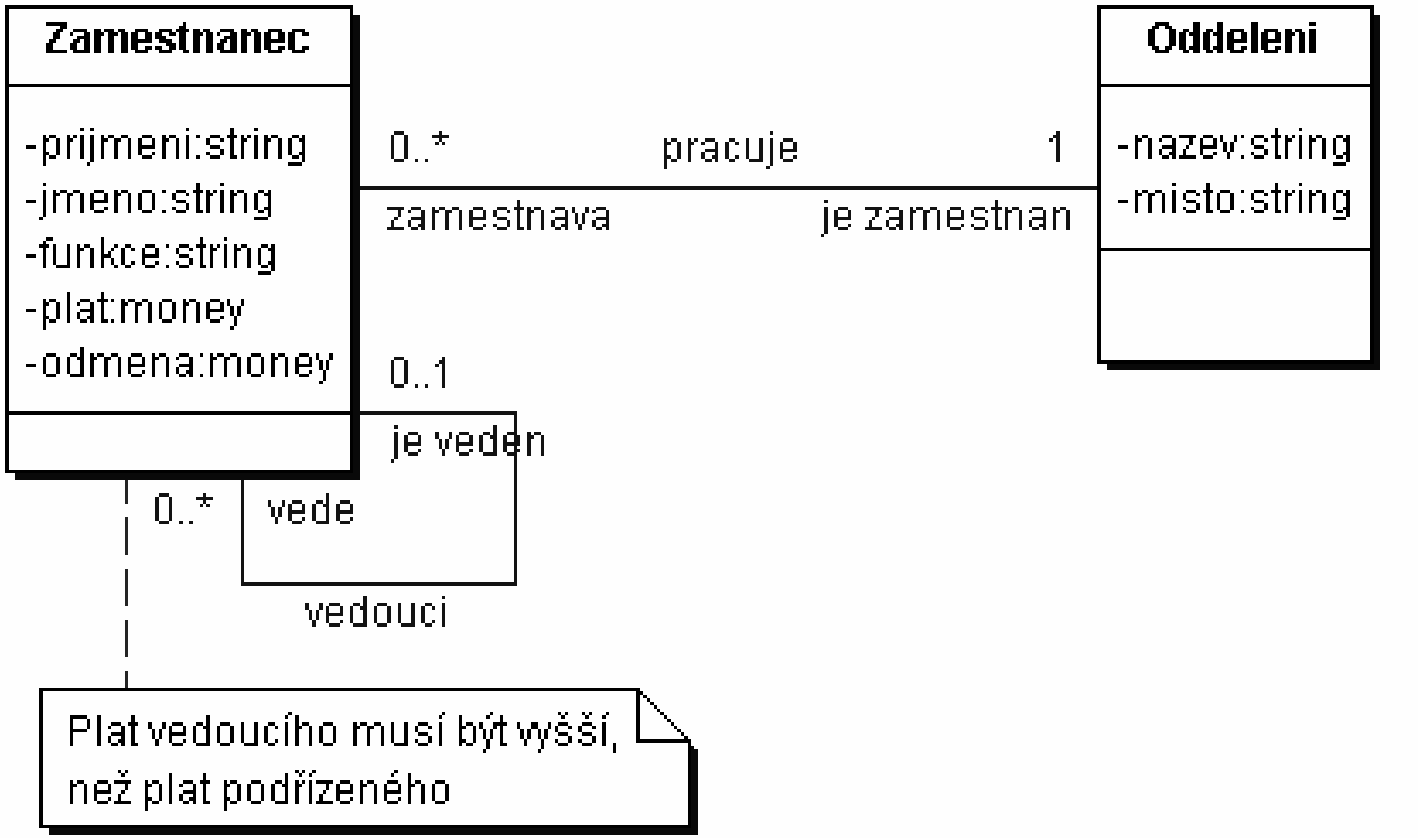


Konceptuální datový model specifikuje

- Typy dat (které entity/třídy, atributy…)
- Vztahy mezi nimi
- Další logická omezení (integrity constraints)

Pozn: Model tříd popisuje i operace, ale ty se často k modelu připojí až v návrhu

Příklad: 1.verze dat.modelu



Integritní omezení jsou součástí specifikace

- Integritní omezení zajišťují, aby popisovaná data (data uložená v databázi) byla pokud možno korektní a úplná.
- V UML můžeme pro specifikaci integritních omezení použít jazyk OCL (Object Constraint Language).

Příklad

„Plat vedoucího musí být vyšší, než plat jeho podřízených“.

OCL:

```
context ZAMESTNANEC inv C1 :
  forAll(ZAMESTNANEC e1,e2 | e1 = e2.vedouci
    implies e1.plat > e2.plat)
```

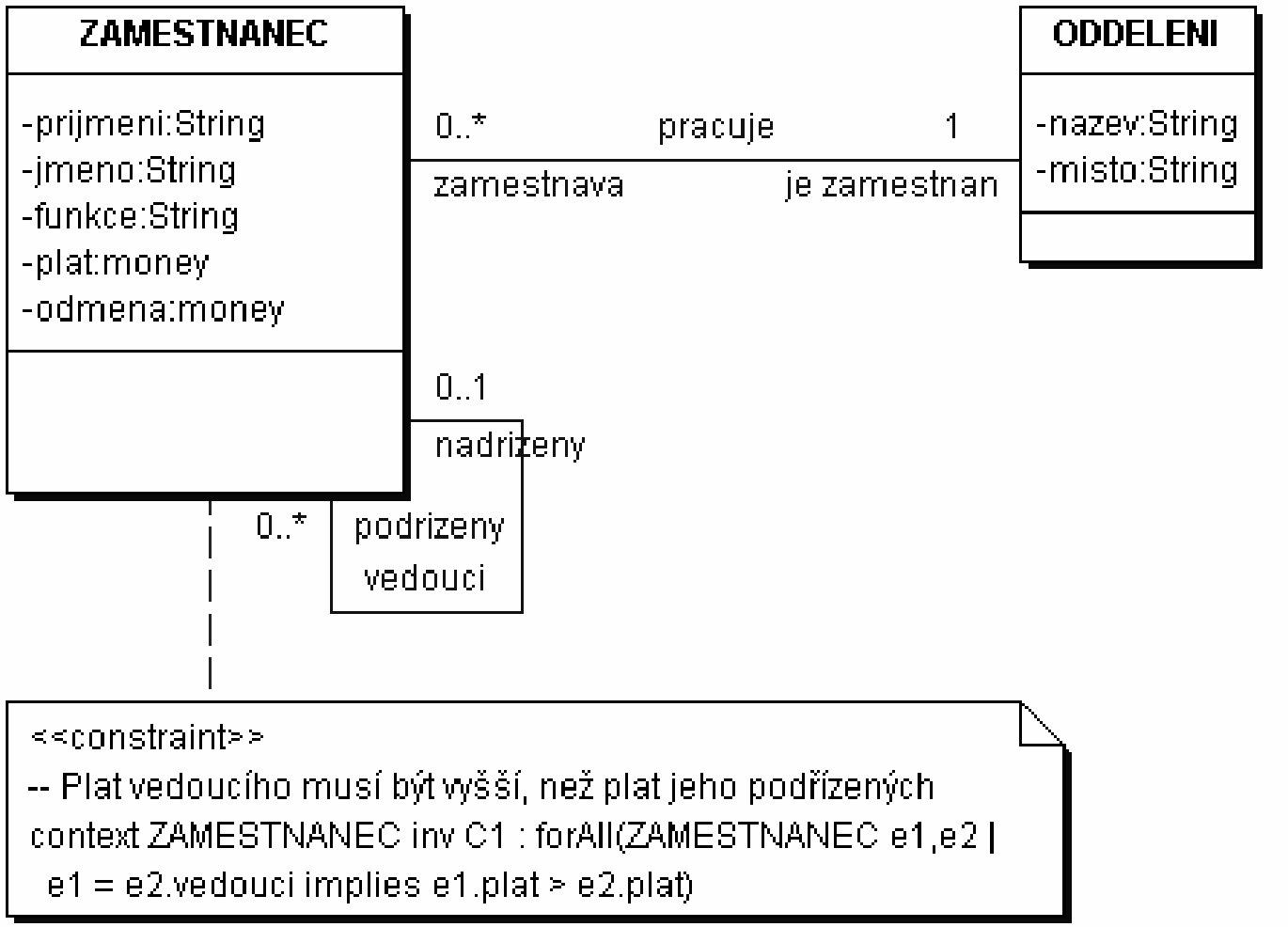
Ekvivalentní logická formule:

(C1) $\forall$**e1,e2** $\in$ **ZAMESTNANEC : e1 = e2.vedouci**

$\Rightarrow$

e1.plat > e2.plat

Příklad: 2.verze dat.modelu



Návrh reprezentace dat

- **Pro každou datovou paměť (úložiště) musíme navrhnout způsob reprezentace - může to být systém ovládání souborů, systém řízení báze dat (relační, objektové, objektově-relační), speciální datový stroj (SW, SW+HW).**
- **Následuje převod konceptuálního modelu do logického.**
- **Součástí převodu je i návrh zajištění integrity dat**
- **Návrh zajištění konzistence dat, zálohování, archivace apod.**

Návrh reprezentace dat pomocí relačního databázového systému

Vstup: konceptuální datový model (diagram tříd + popis integritních omezení)

Výstup: logický relační datový model (SQL-1999), včetně návrhu realizace integritních omezení

Pozn.: Výstupem je obecné SQL, při skutečné implementaci návrhu musí být ještě výstup přizpůsoben konkrétnímu stroji

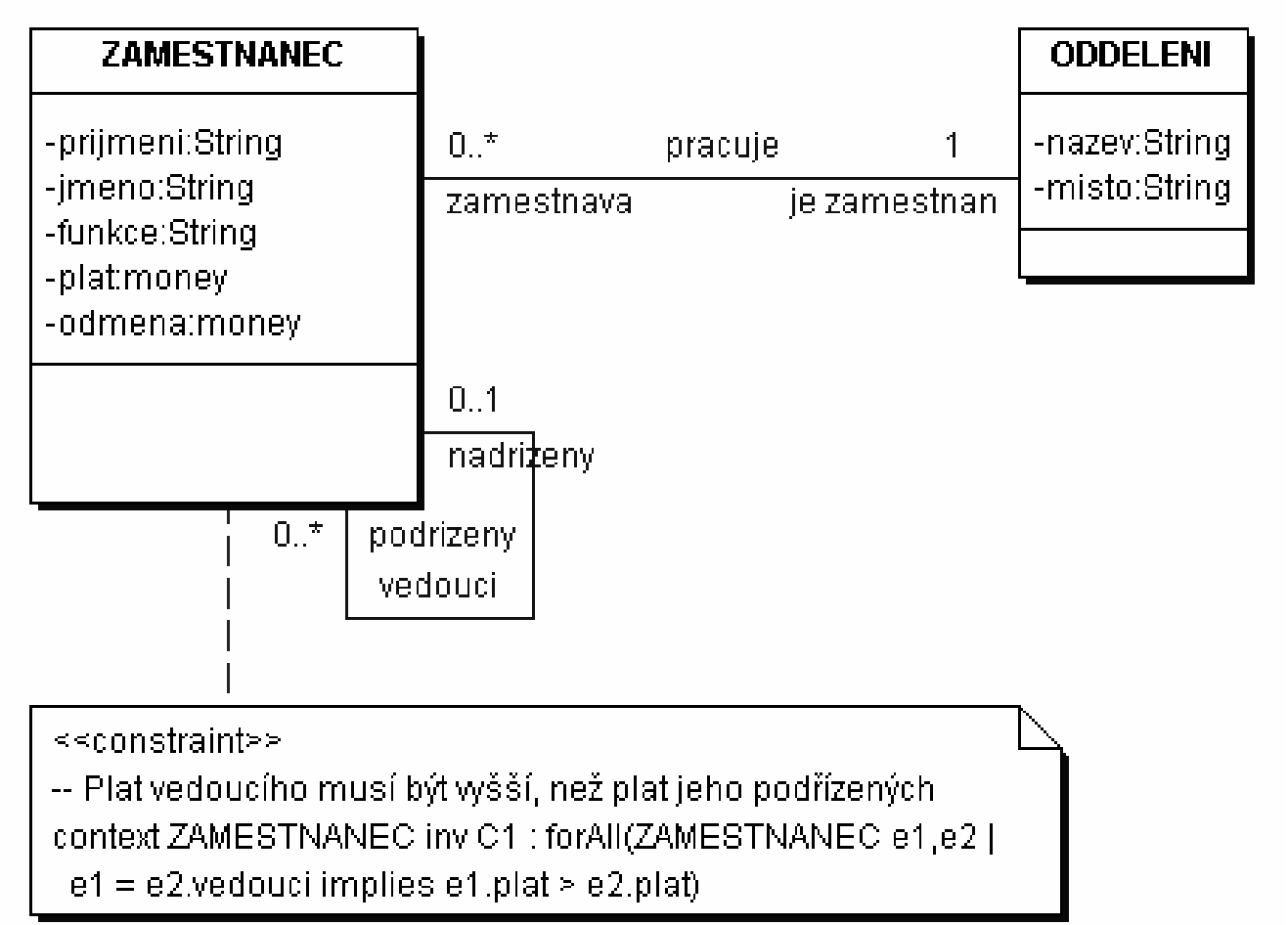
Postup návrhu

- Úprava (normalizace) konceptuálního modelu
- Návrh reprezentace typů (entit)
- Návrh reprezentace vztahů
- Návrh reprezentace integritních omezení

Úprava (normalizace) konceptuálního modelu

- Vyloučení multihodnotových a násobných atributů
- Vyloučení funkčních závislostí (odstranění redundance dat) – převod modelu do 3-NF (příp. 4-té, či 5-té normální formy)
- Náhrada nebinárních vztahů binárními
- Náhrada vztahů typu M:N přidruženými třídami

Normalizovaný datový model



Návrh reprezentace

- Pro každou jednoduchou entitu (typ) navrhneme tabulku, jméno tabulky bude množné číslo jména typu.
- Návrh jmen sloupců pro reprezentaci atributů a odpovídajících domén.
- Doplníme informace o volitelnosti formátu sloupců.
- Z nejčastěji používané unikátní identifikace vytvoříme primární klíč, nebo zavedeme nový identifikační sloupec (OID).

Návrh reprezentace (pokr.)

- Pro N-konce vztahů přidáme k tabulce jednoznačné identifikace z tabulky na 1-konci (volitelné vztahy indikují nepovinnost. Současně přidáme odpovídající cizí klíče.
- Pro každý vztah typu nadtyp/podtyp navrhneme reprezentaci (společná tabulka s rozlišovací položkou, samostatné tabulky).
- Pro každý vztah typu celek/část navrhneme reprezentaci (společná tabulka s rozlišovací položkou, samostatné tabulky).

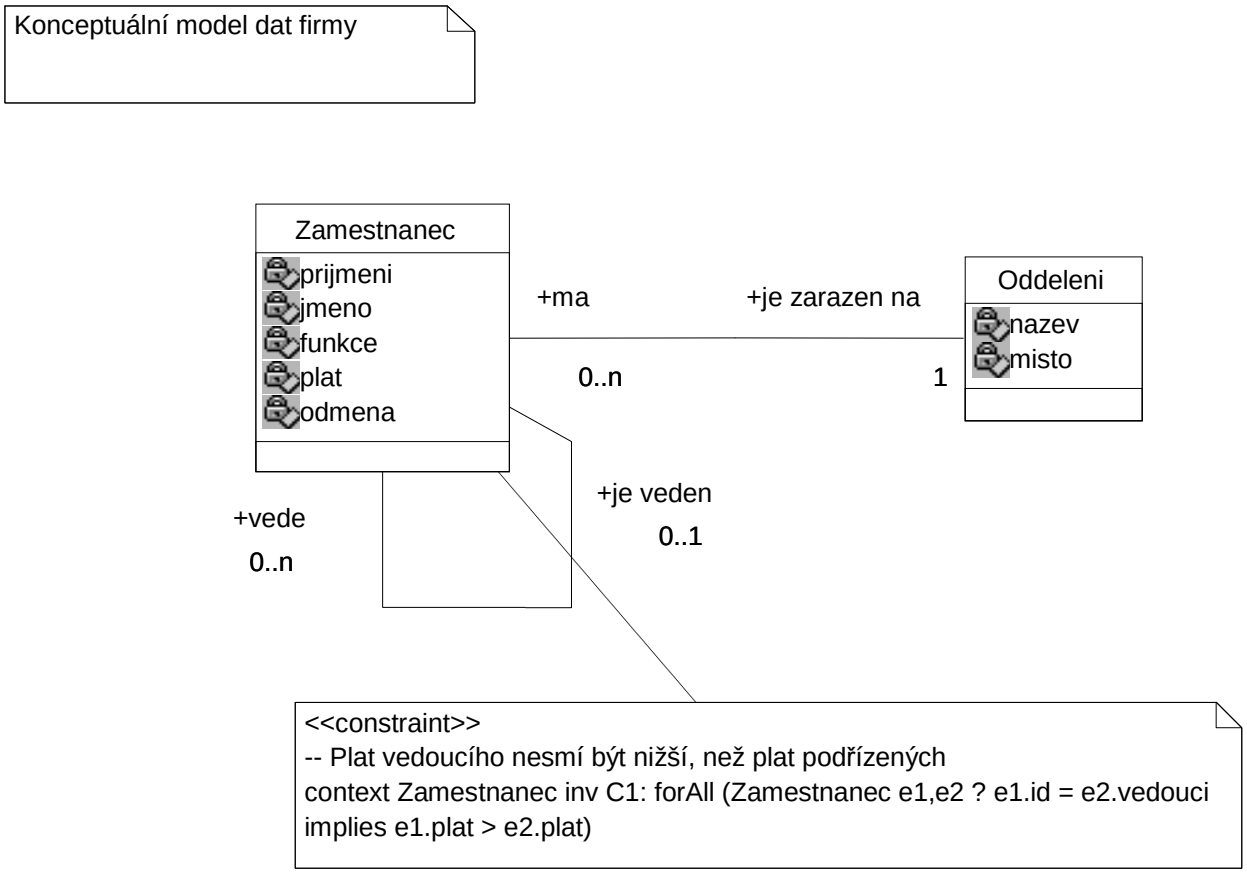
Návrh reprezentace (pokr.)

- Pro každý exkluzivní vztah (exkluzivní podtypy) rozhodneme, zda se má řešit společnou doménou, nebo explicitními cizími klíči.
- Doplníme sloupce odpovídající často používaným odvozeným atributům a navrhneme mechanizmus jejich údržby.
- Navrhneme indexy pro často využívané unikátní kombinace, které nejsou realizovány jako primární klíče. Indexy rovněž vytvoříme pro cizí klíče.

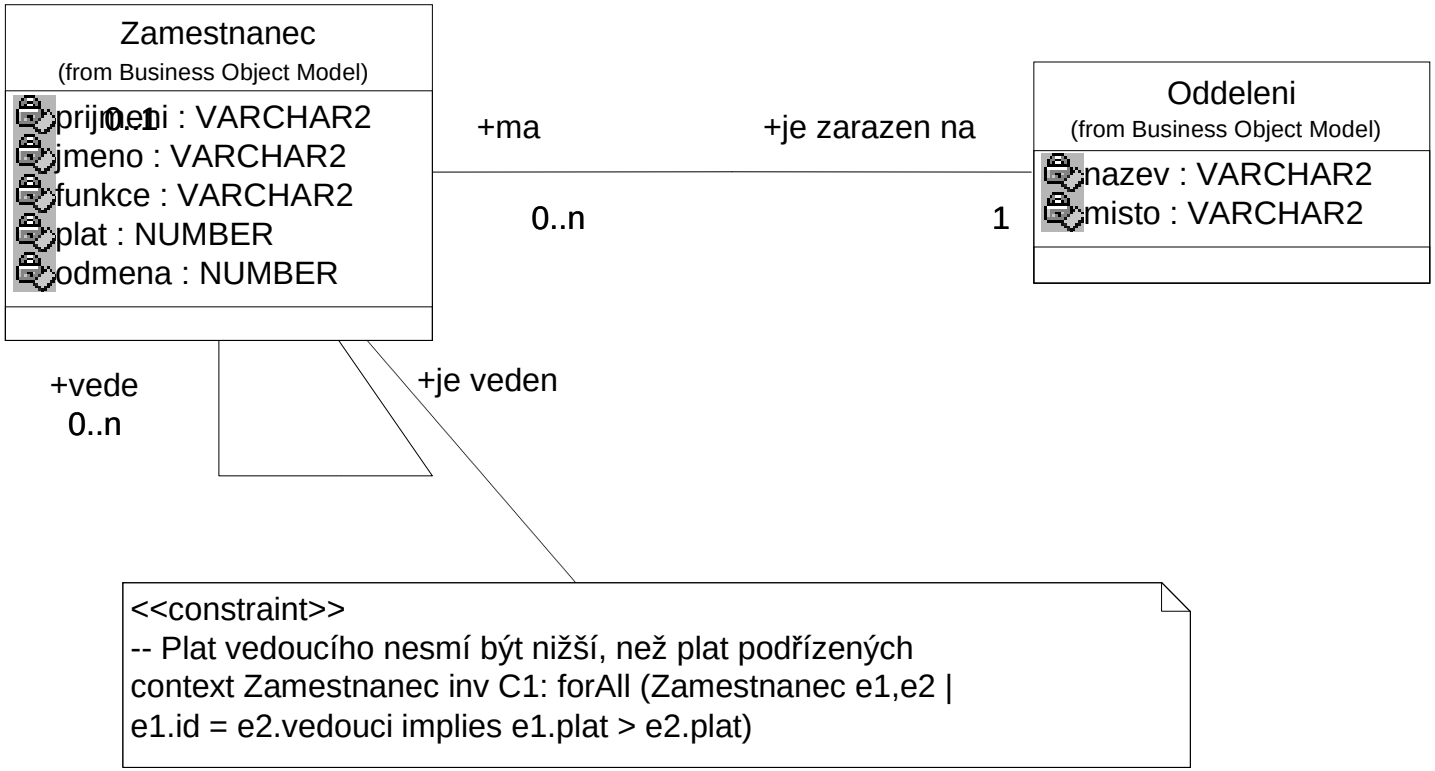
Návrh reprezentace (pokr.)

- Přidáme definice pohledů (zejména pro nadtypy, podtypy, celky a části).
- Pro generované primární klíče přidáme definice sekvencí pro jejich generování (může být implementačně závislé).
- Navrhneme řešení integritních omezení (použijeme deklarativní relační integritní omezení, nebo navrhneme „triggery“).

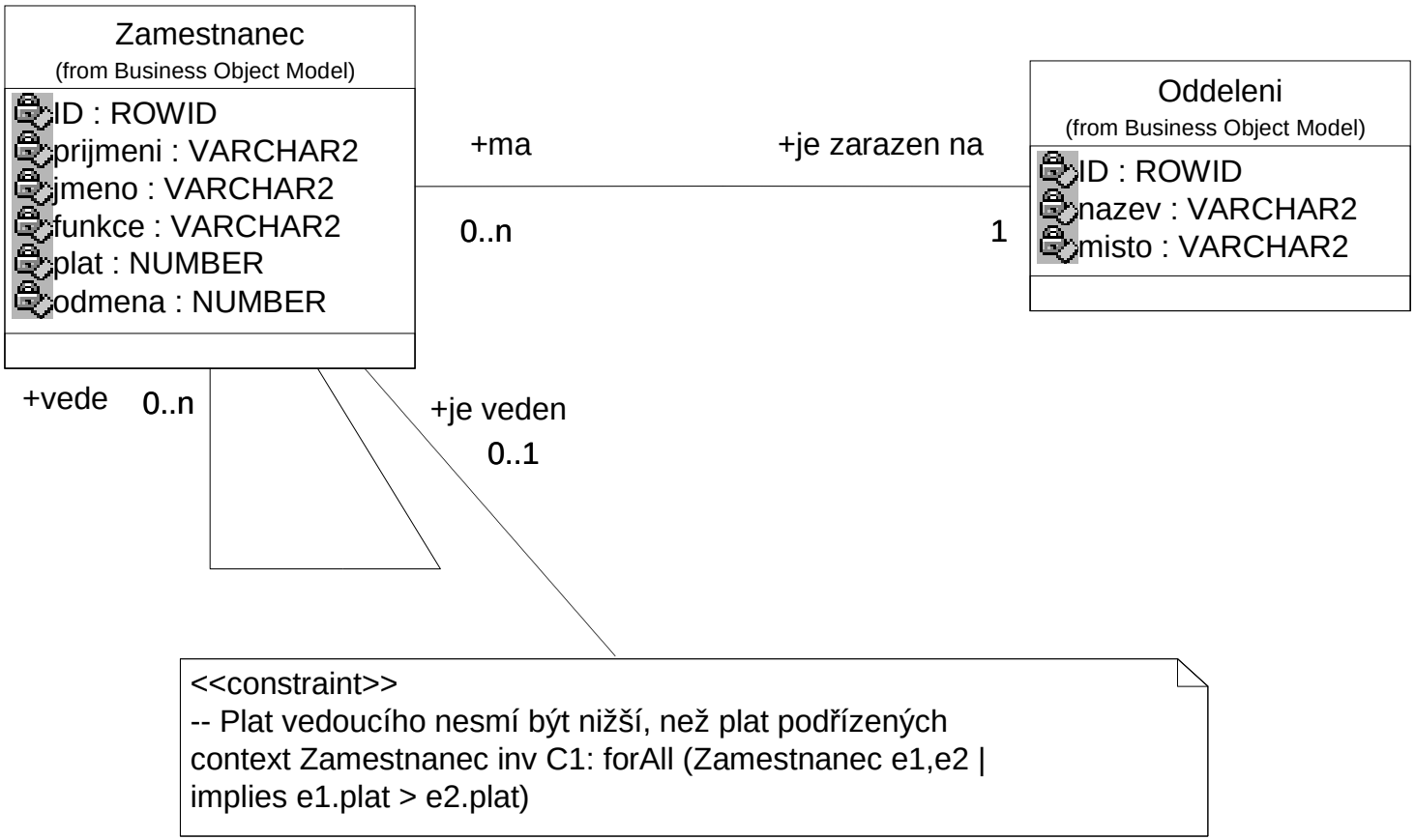
Konceptuální model



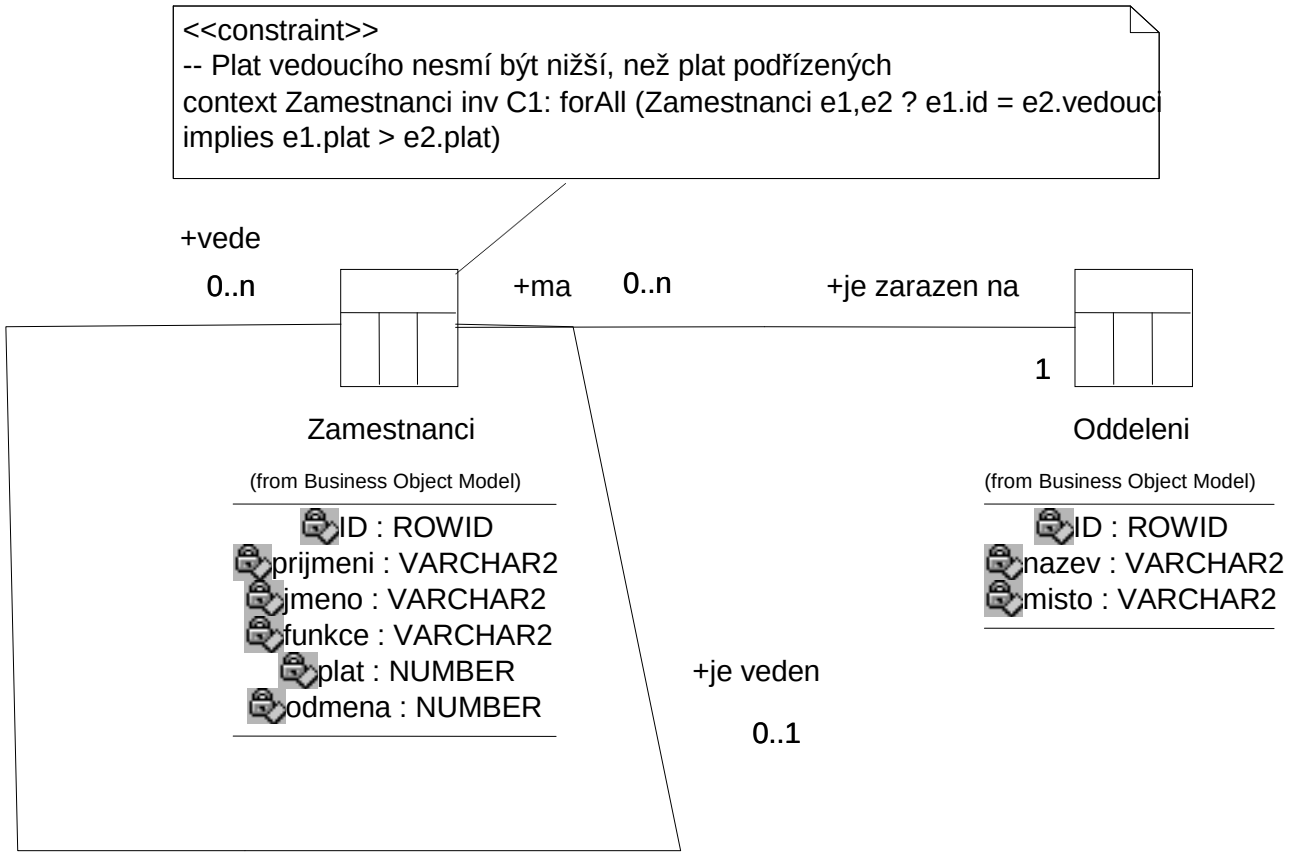
Jména a domény



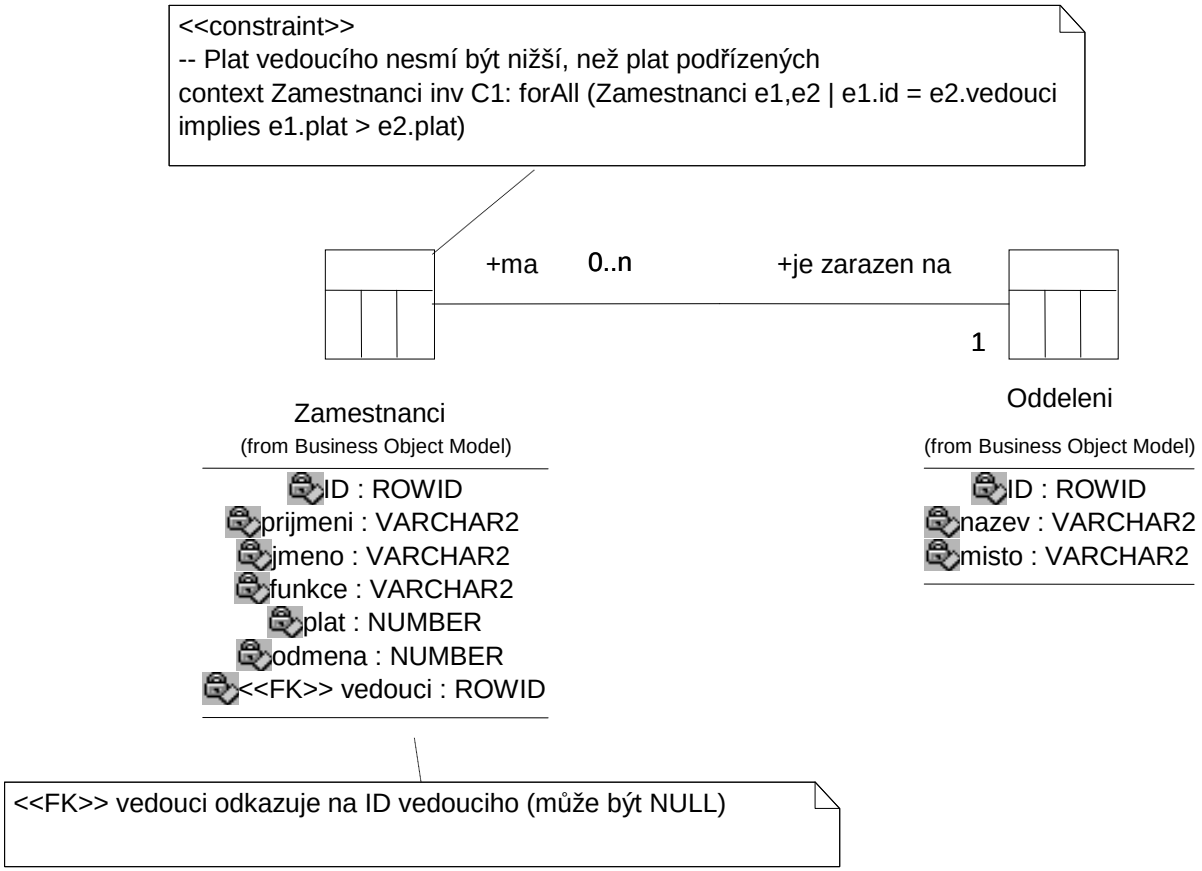
Primární identifikace



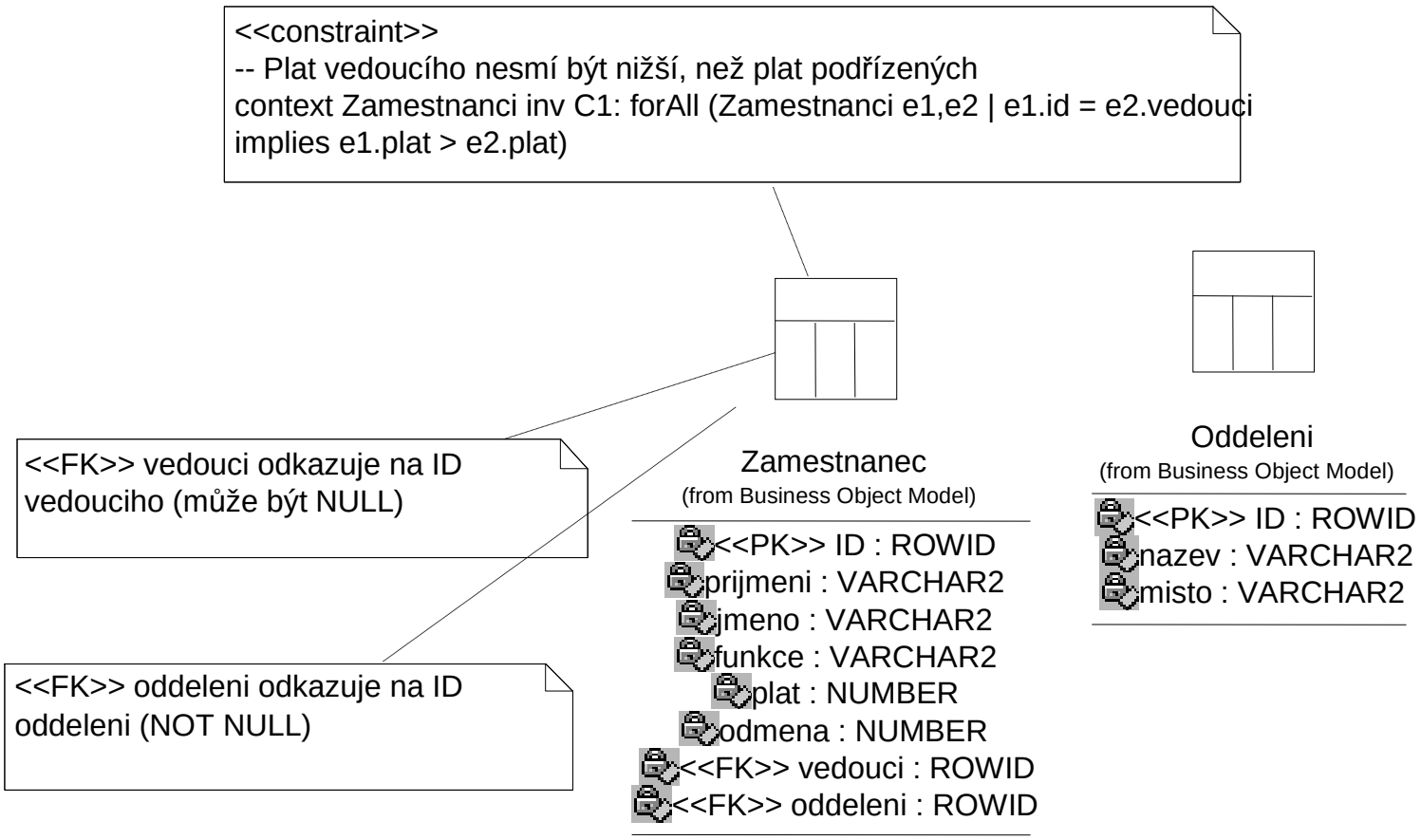
Vytvoříme tabulky



Cizí klíč pro “vedoucího”



Cizí klíč pro “zaměstnance”



Návrh reprezentace integritních omezení

- Zkusíme vytvořit deklarativní omezení.
- Pokud by nefungovala, musíme navrhnout „triggery“.

Příklad: "forAll"

```
context Zamestnanci inv C1: forAll
  (Zamestnanci e1,e2 | e1.id =
  e2.vedouci implies e1.plat > e2.plat)
```

⇒

```
alter table Zamestnanci
 add constraint C1 check (not exists
  (select 'X'
   from Zamestnanci e1, Zamestnanci e2
   where e1.id = e2.vedouci
   and e2.plat >= e1.plat));
```

Odvozené SQL I. (tabulka Oddeleni)

```
create table Oddeleni (
  ID ROWID primary key,
  nazev VARCHAR2(20),
  misto VARCHAR2(20)
);
```

Odvozené SQL II. (table EMP)

```
create table Zamestnanci (
  ID ROWID primary key,
  prijmeni VARCHAR2(35),
  jmeno VARCHAR2(35),
  funkce VARCHAR2(10),
  plat NUMBER(9,2),
  odmena NUMBER(9,2),
  vedouci ROWID references EMP(id),
  oddeleni ROWID not null references
  Oddeleni(ID)
);
```

Odvozené SQL III. (ostatní)

```
alter table Zamestnanci add constraint C1
check (not exists
  (select ‘X’
   from Zamestnanci e1, Zamestnanci e2
   where e1.ID = e2.vedouci
   and e2.plat > e1.plat));
```

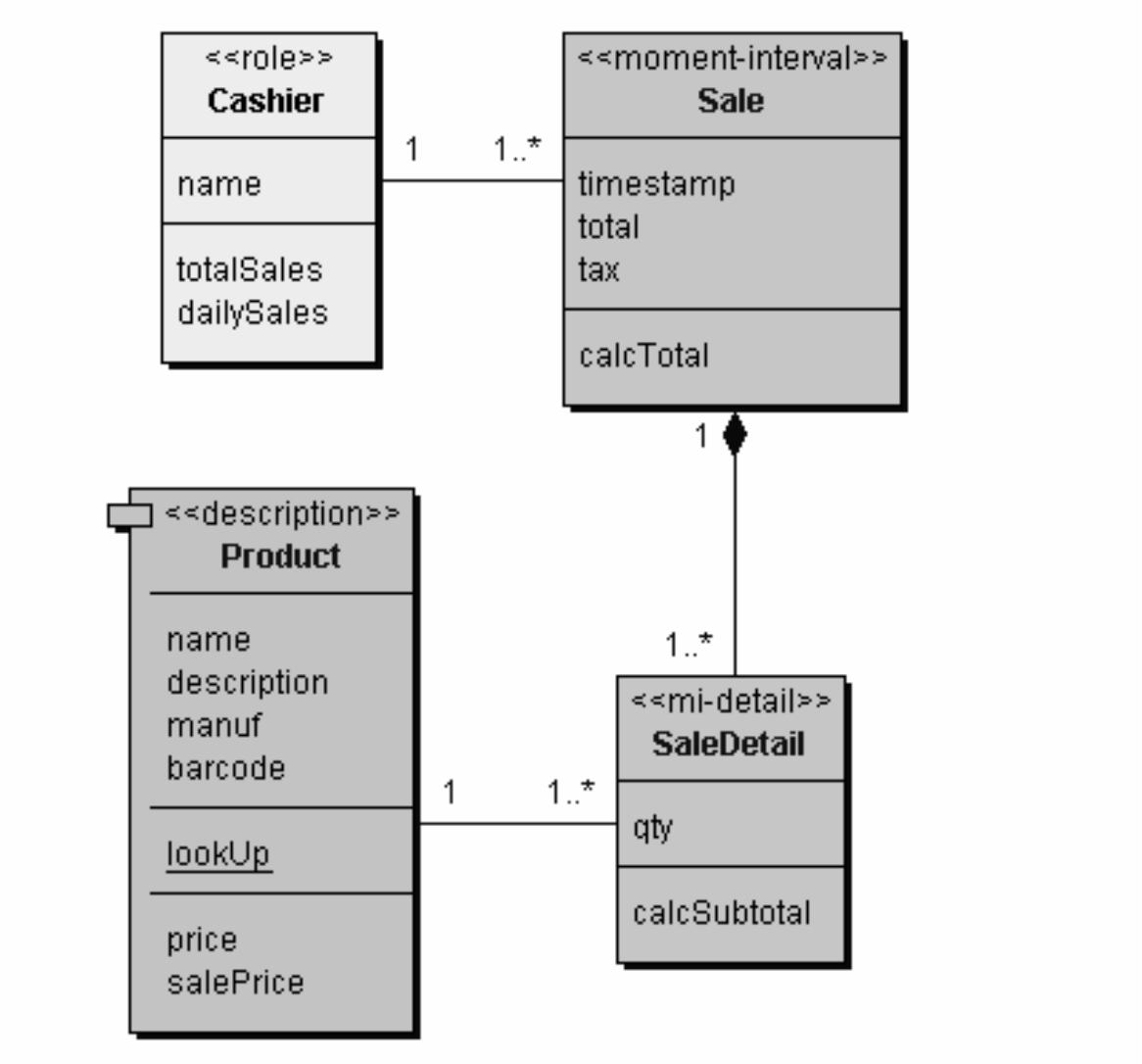
Ale problém – někde to nefunguje !!!

Zkusíme trigger !!!

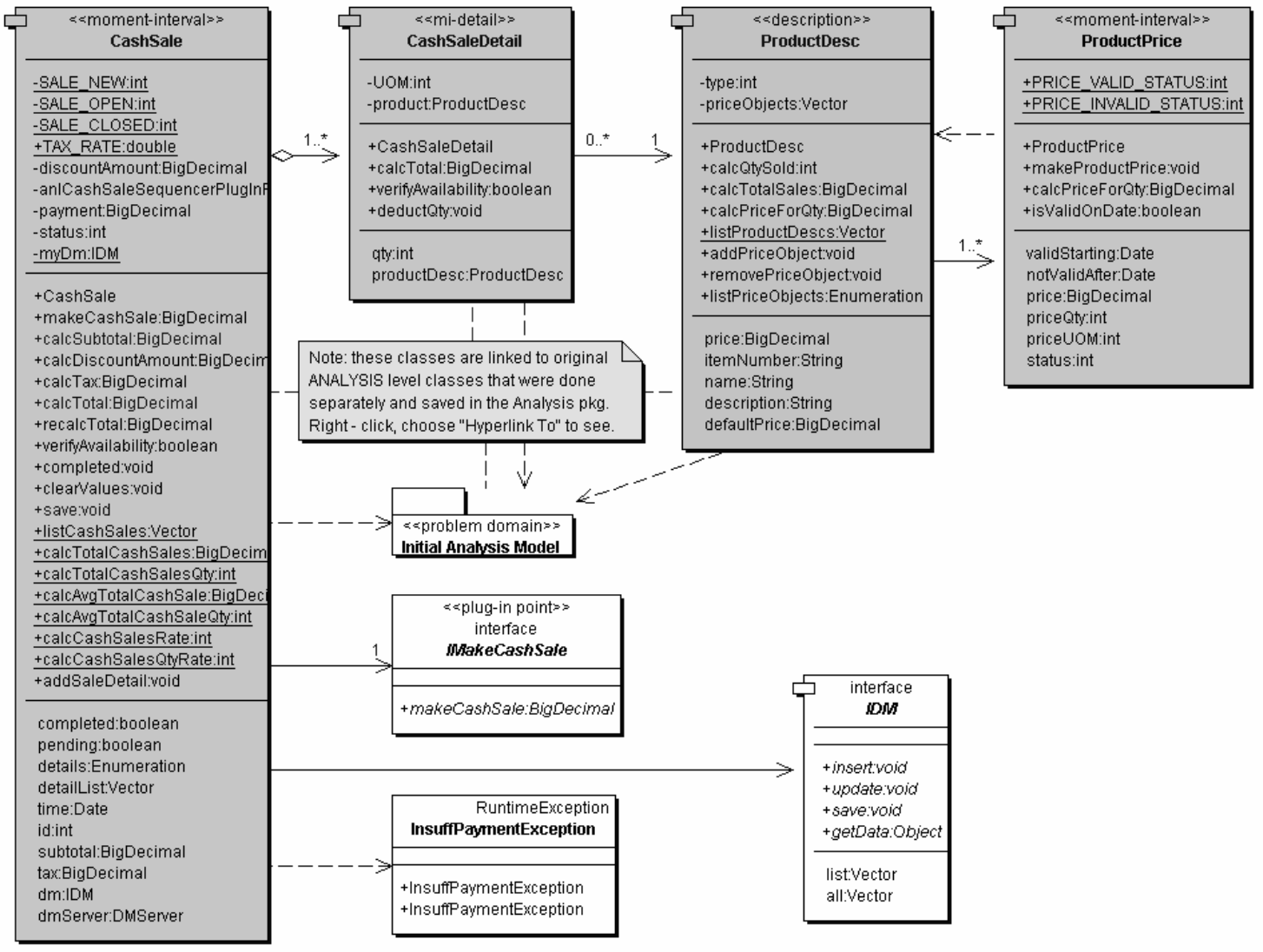
Příklad (zjednodušeno)

```
create or replace trigger C1
before insert or update of plat on Zamestnanci e1
declare
  cnt INTEGER;
begin
  select count(*) into cnt from Zamestnanci e2
    where e1.ID = e2.vedouci
    and e2.plat >= e1.plat;
  if cnt = 0 then
    e1.plat := :new.plat
  end if;
end;
```

Pokladna (analytický model)

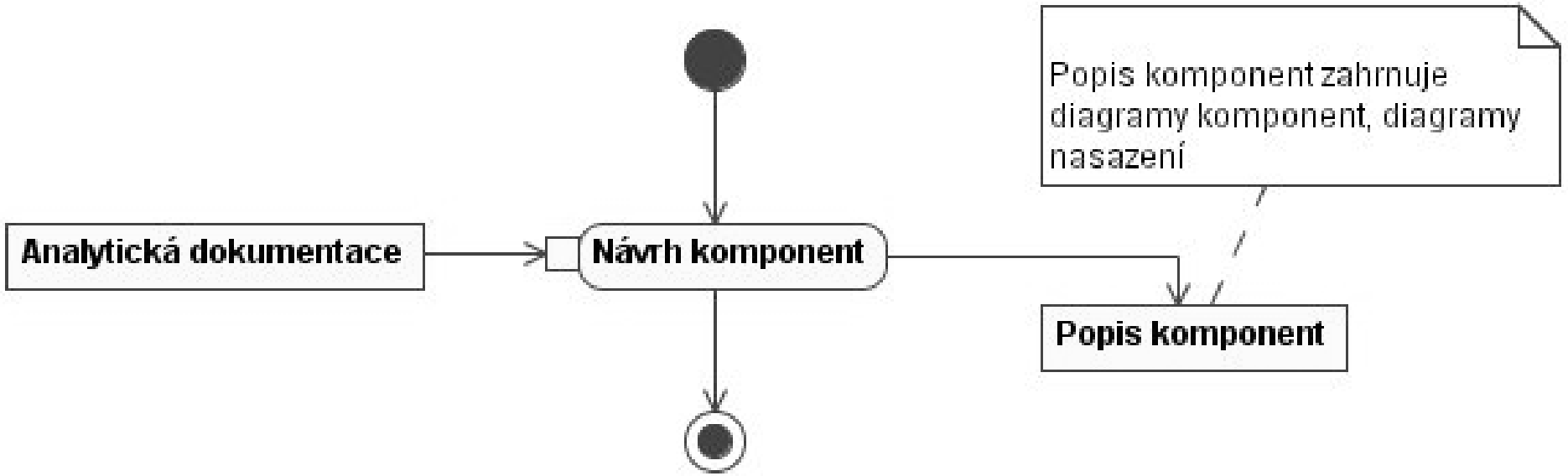


Pokladna (datový model po návrhu)



Návrh komponent

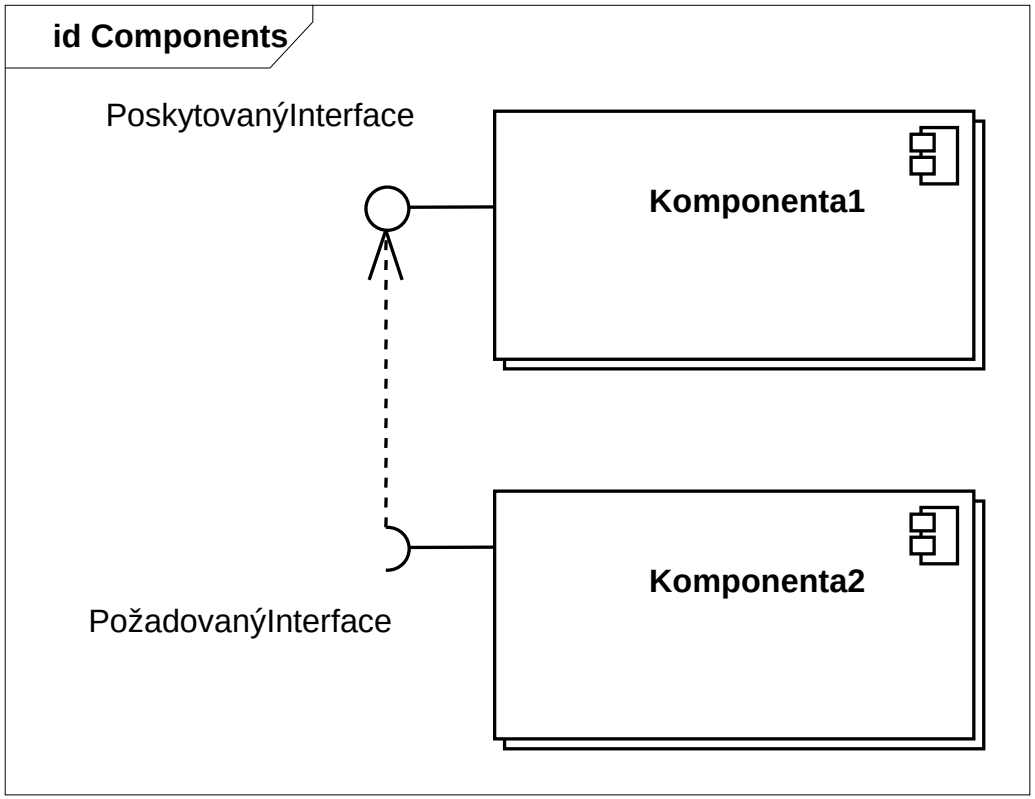
Návrh komponent



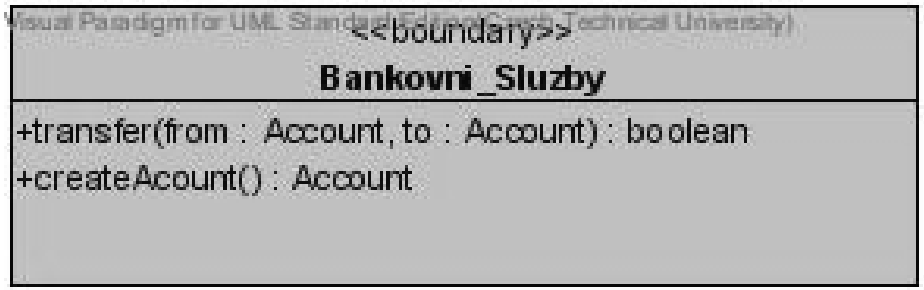
Co to je komponenta?

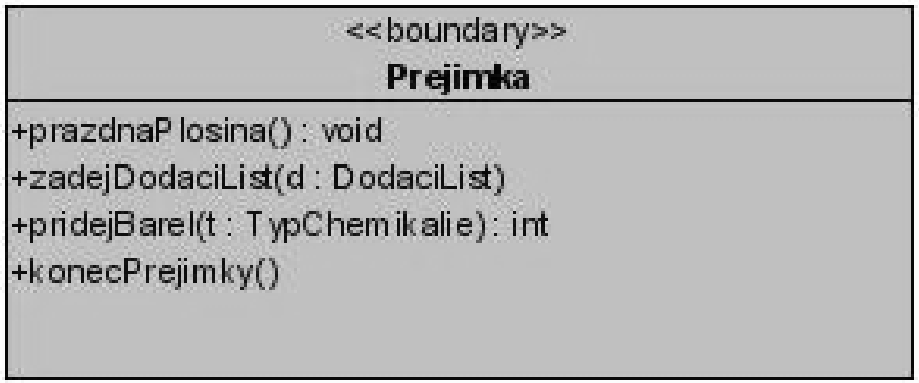
- Modulární, nasaditelná, nahraditelná a opakovaně použitelná část systému, která zapouzdřuje implementaci a zveřejňuje rozhraní.
- Komponenta je skupina objektů, které dohromady:
  - poskytují určitá rozhraní pro externí objekty
  - vyžadují určitá rozhraní od externích objektů (vyžadují součinnost)
- Na rozdíl od tříd, nemusí existovat instance komponenty (vyhýbají se nutnosti mít stav)
- Nedodává se obvykle ve zdrojovém tvaru, ale jako vykonatelný kód
- Často vyžaduje pro práci určité prostředí (např. kontejner)

Notace diagramů komponent I.



Interface vyžaduje popis





Komponenta se může skládat z dalších komponent

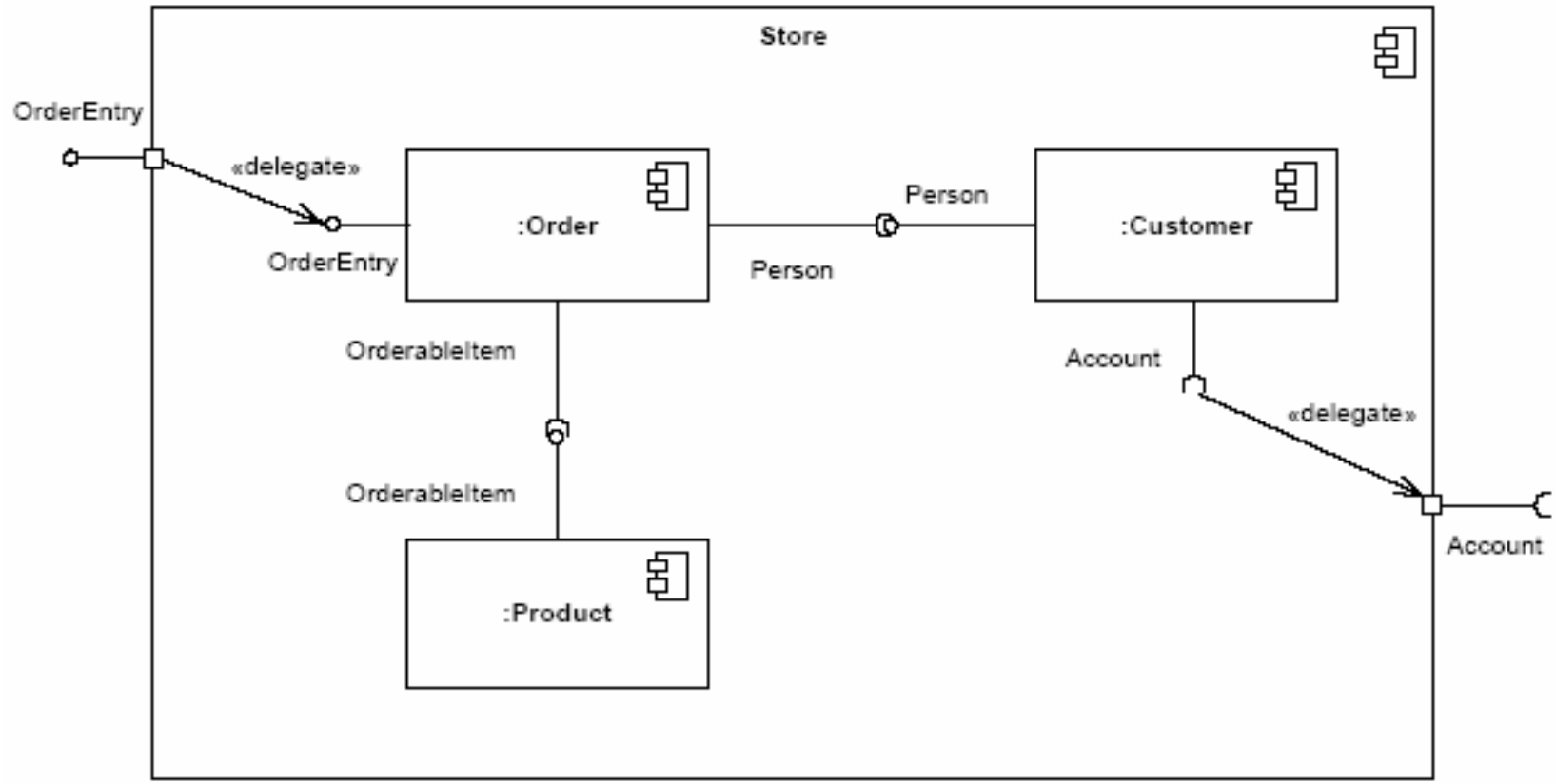
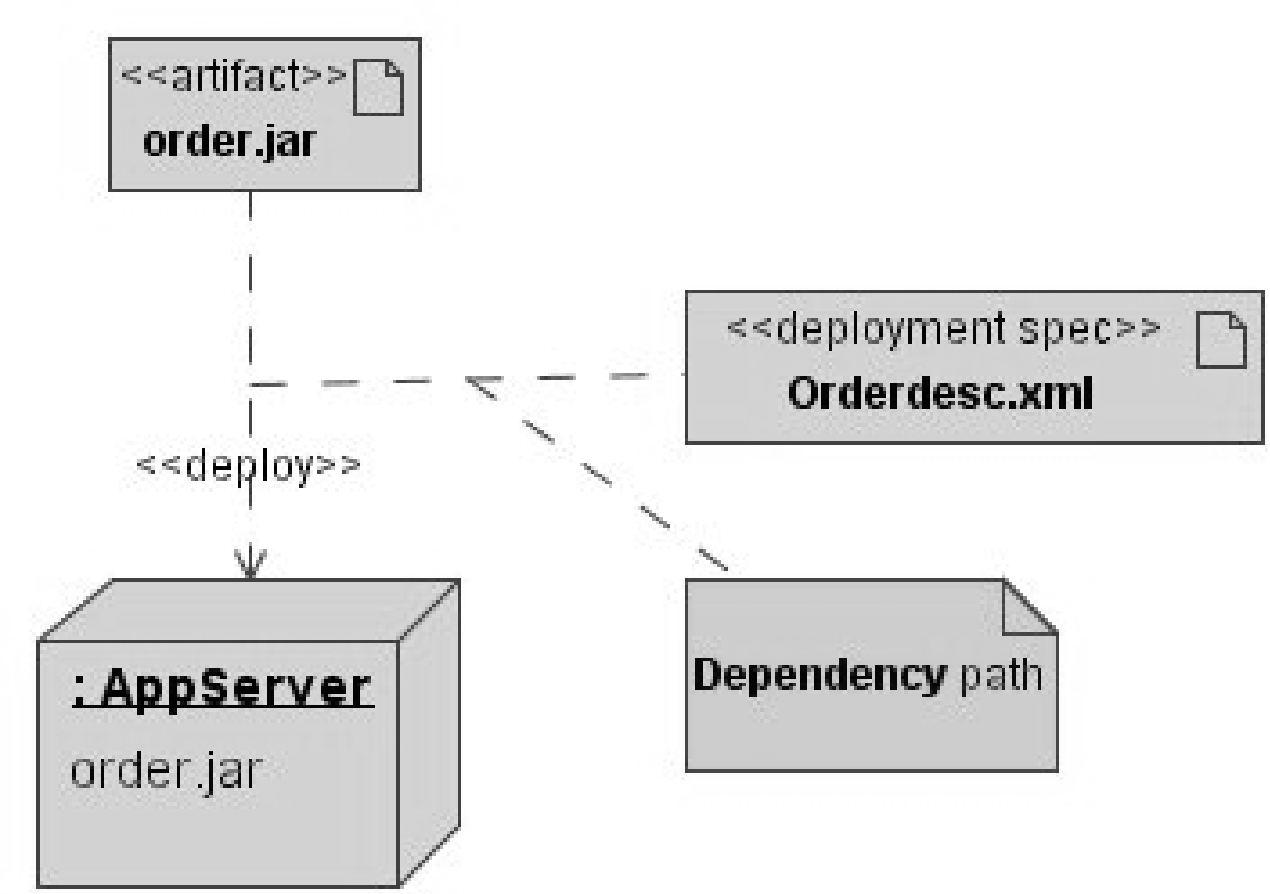
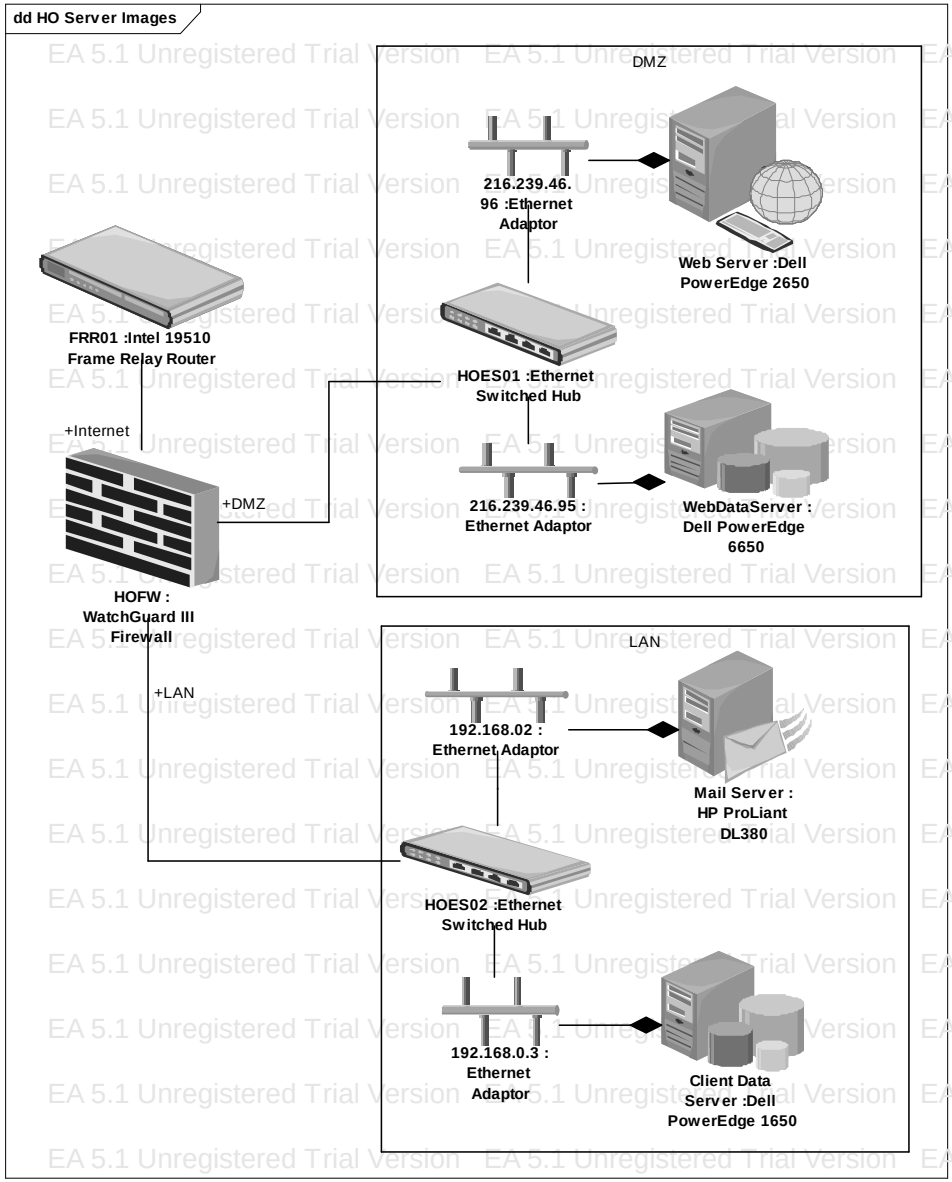


Diagram nasazení komponenty





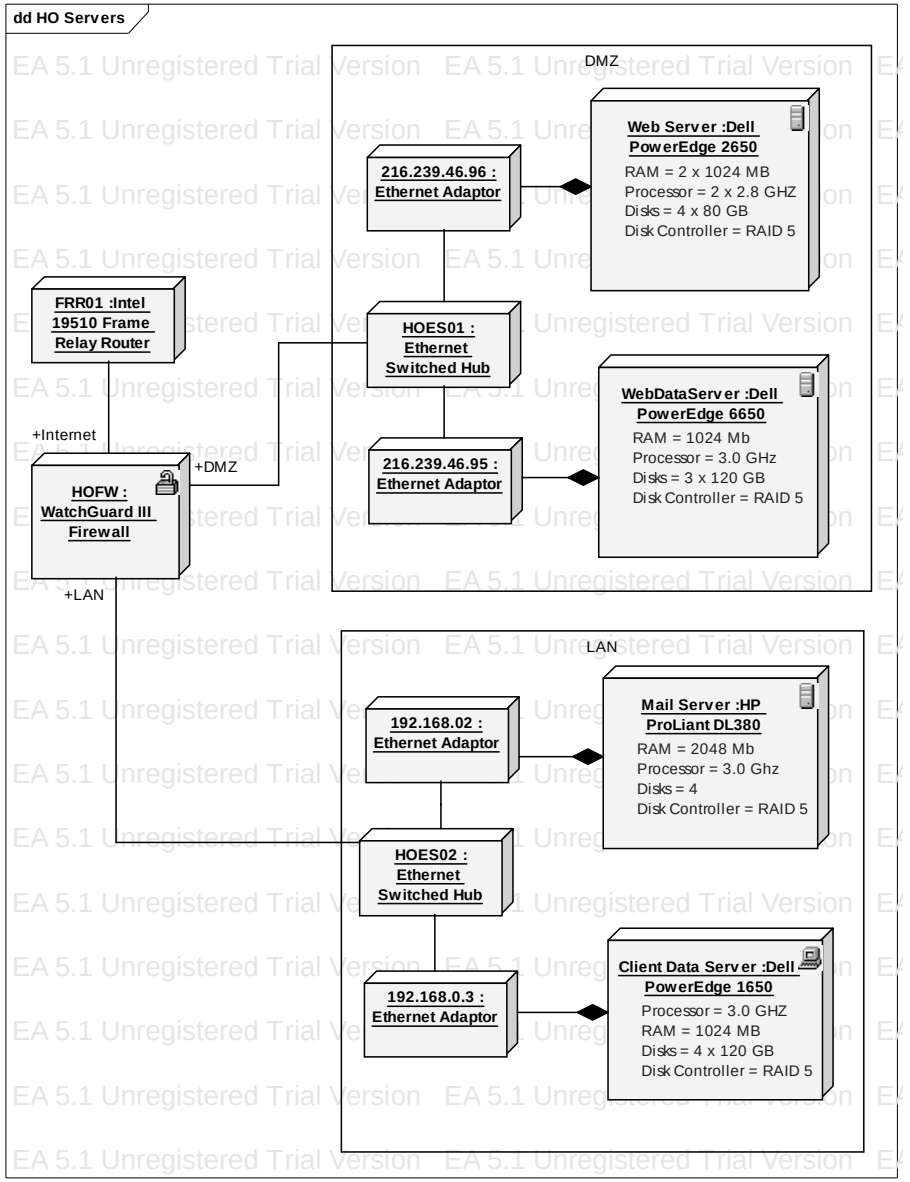
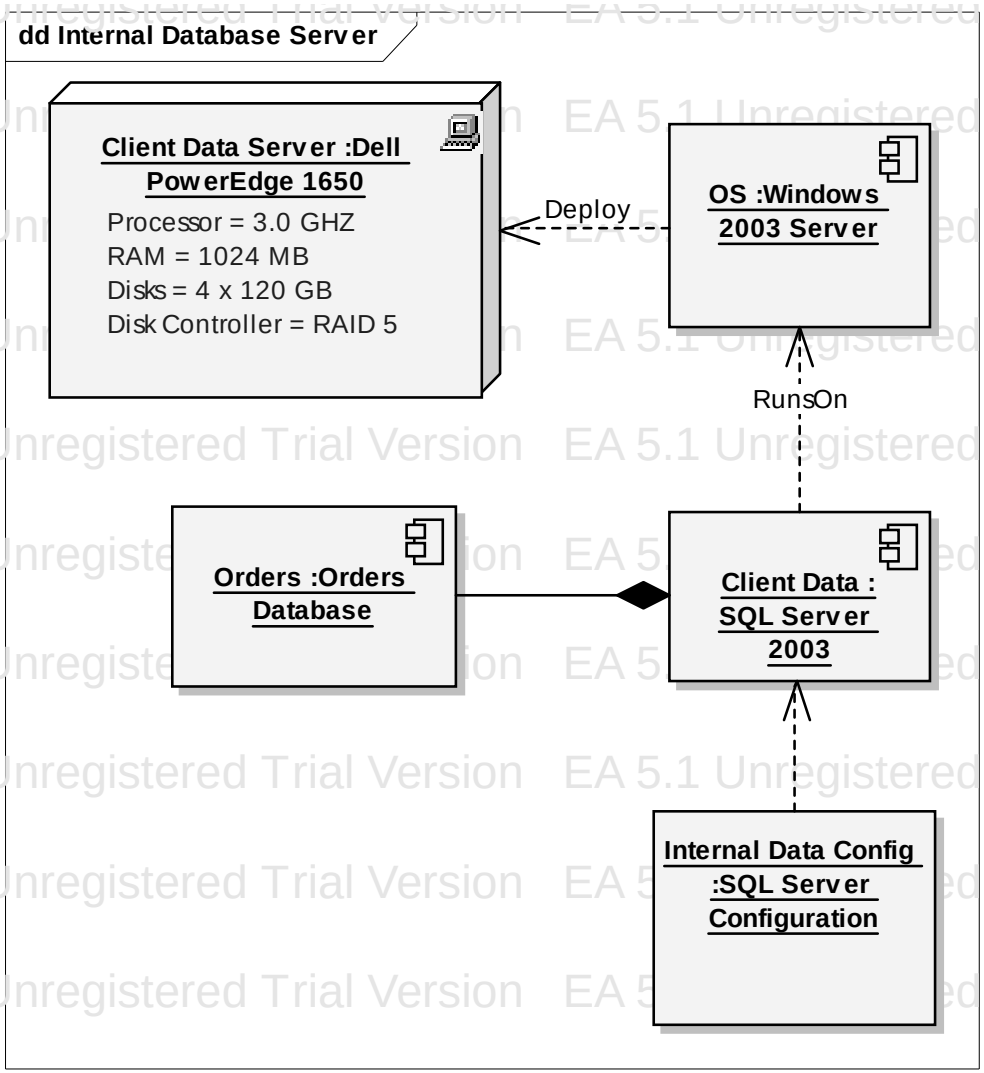
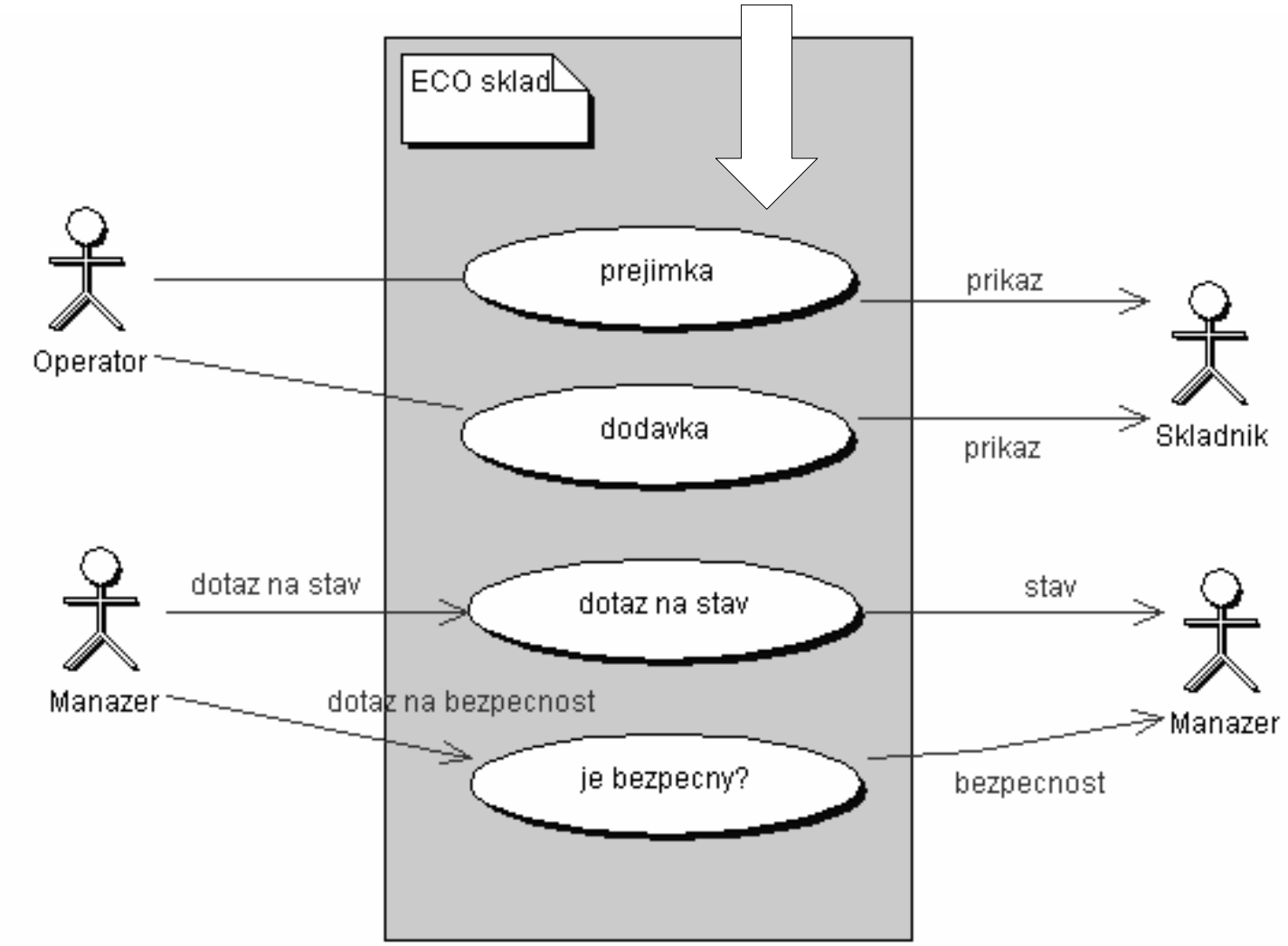


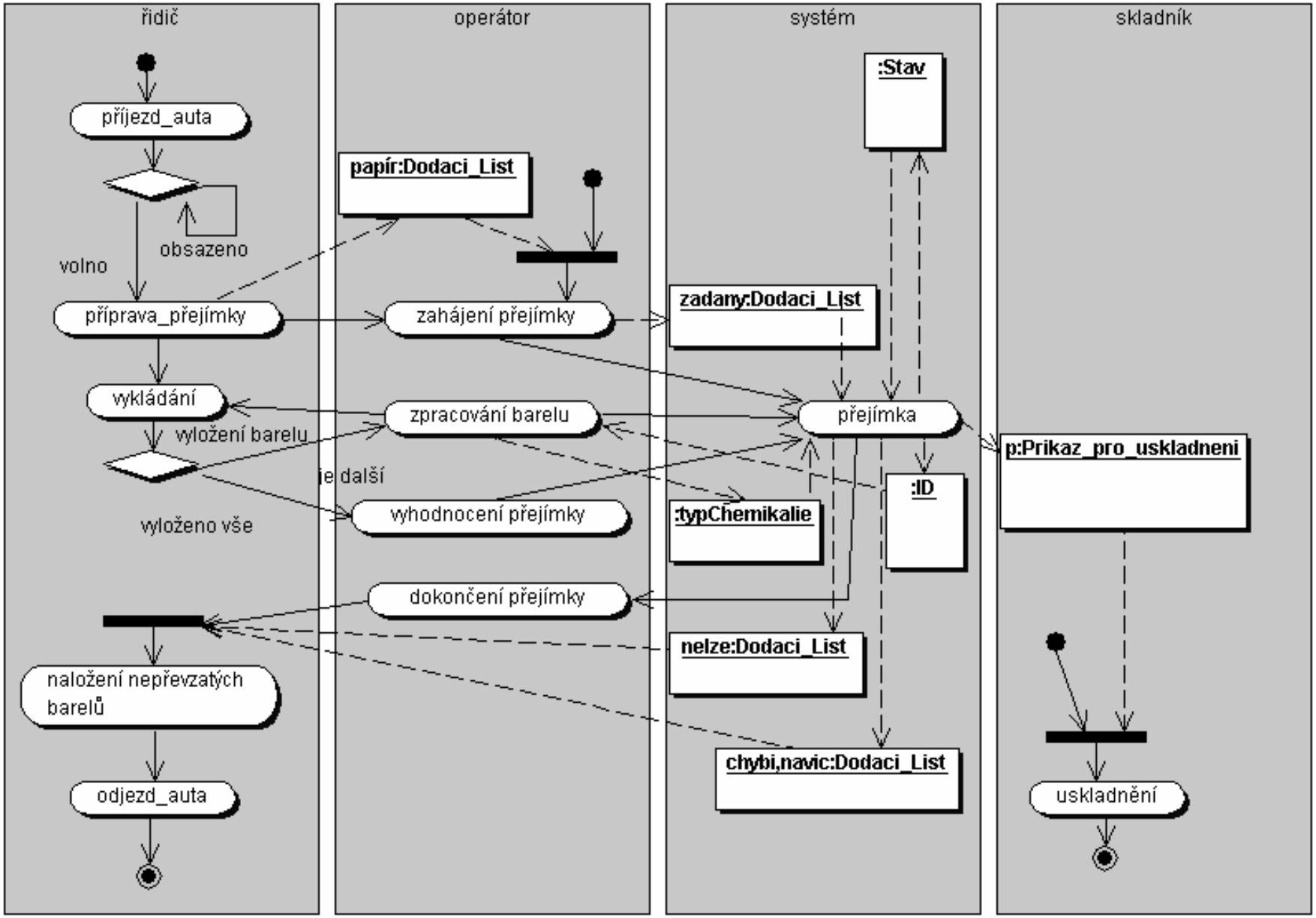
Diagram nasazení komponenty DB



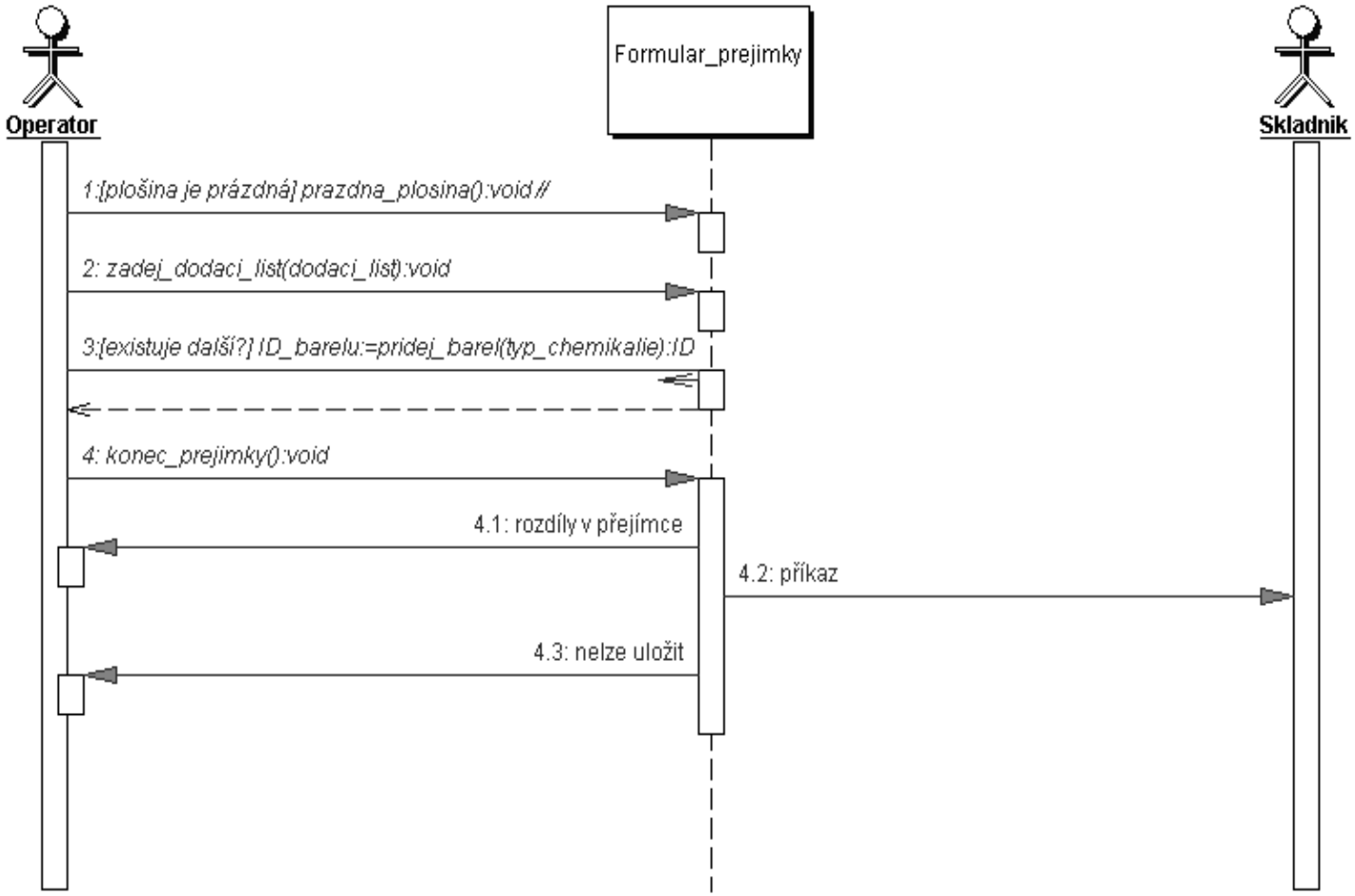
Příklad: Návrh dle Fusion

Př. ECO sklad


ECO sklad
prejimka
dodavka
dotaz na stav
dotaz na bezpecnost
je bezpecny?
Operator
Manazer
Skladnik
Manazer
prikaz
prikaz
dotaz na stav
stav
bezpecnost

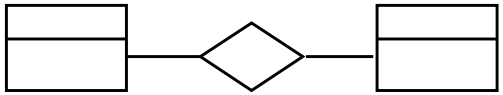

řidič
operátor
systém
skladník
příjezd_auta
obsazeno
volno
příprava_přejímky
vykládání
vyložení barelu
vyloženo vše
naložení nepřevzatých barelů
odjezd_auta
papír:Dodaci_List
zahájení přejímky
zpracování barelu
vyhodnocení přejímky
dokončení přejímky
:Stav
zadany:Dodaci_List
přejímka
:typChemikalie
:ID
nelze:Dodaci_List
chybi,navic:Dodaci_List
p:Prikaz_pro_uskladneni
uskladnění

Scénář pro “přejímku” v UML

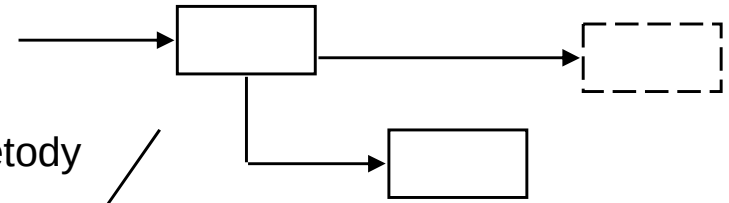


Princip návrhu ve Fusion

Datový model



Model komunikace objektů

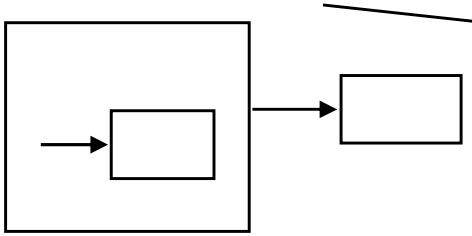


datové atributy

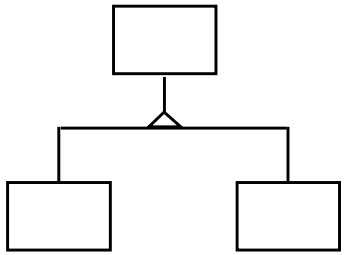
metody

Viditelnost

objektové atributy



isa



Popis tříd

class A **isa** S

  attribute a : int

  attribute b : **shared** B

  …

  method f(par)

  …

endclass

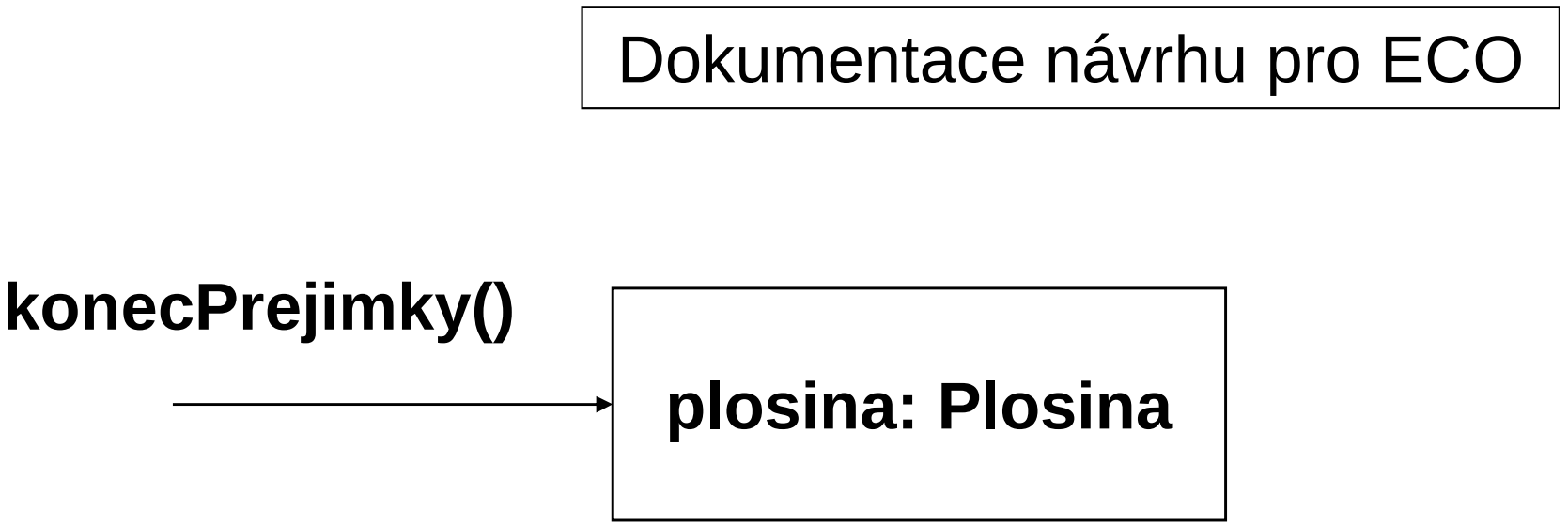
Dědičnost

Datový slovník

Postup návrhu ve Fusion

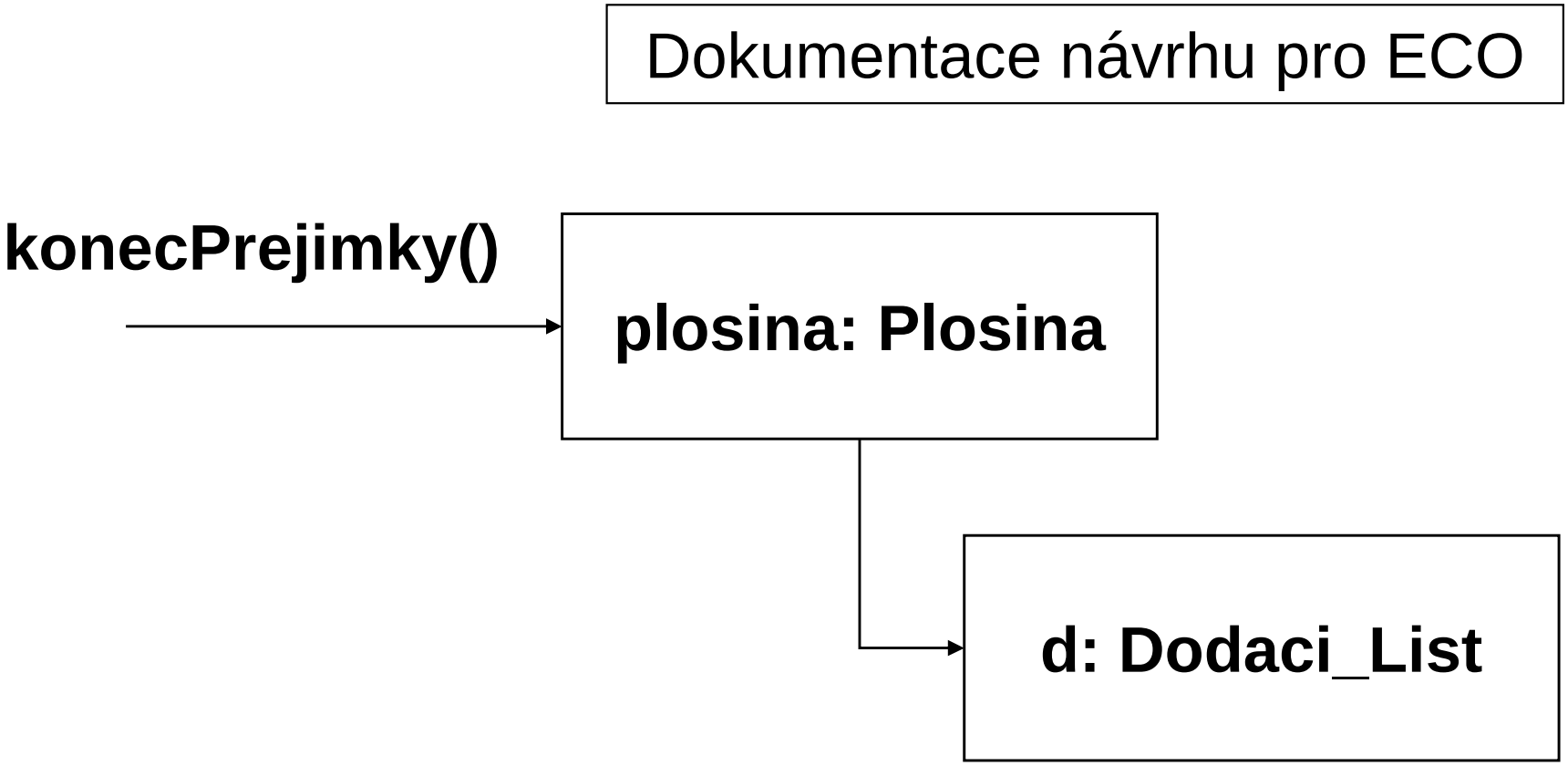
1. **Pro každou operaci funkčního modelu nakresli diagram komunikace (včetně metod vzniklých při návrhu)**
2. Pro každou třídu datového modelu zakresli potřebnou viditelnost podle diagramů komunikace
3. Pro každou třídu datového modelu vytvoř popis třídy
4. Doplň návrh o dědičnost

Diagram komunikace pro “konec přejímky“



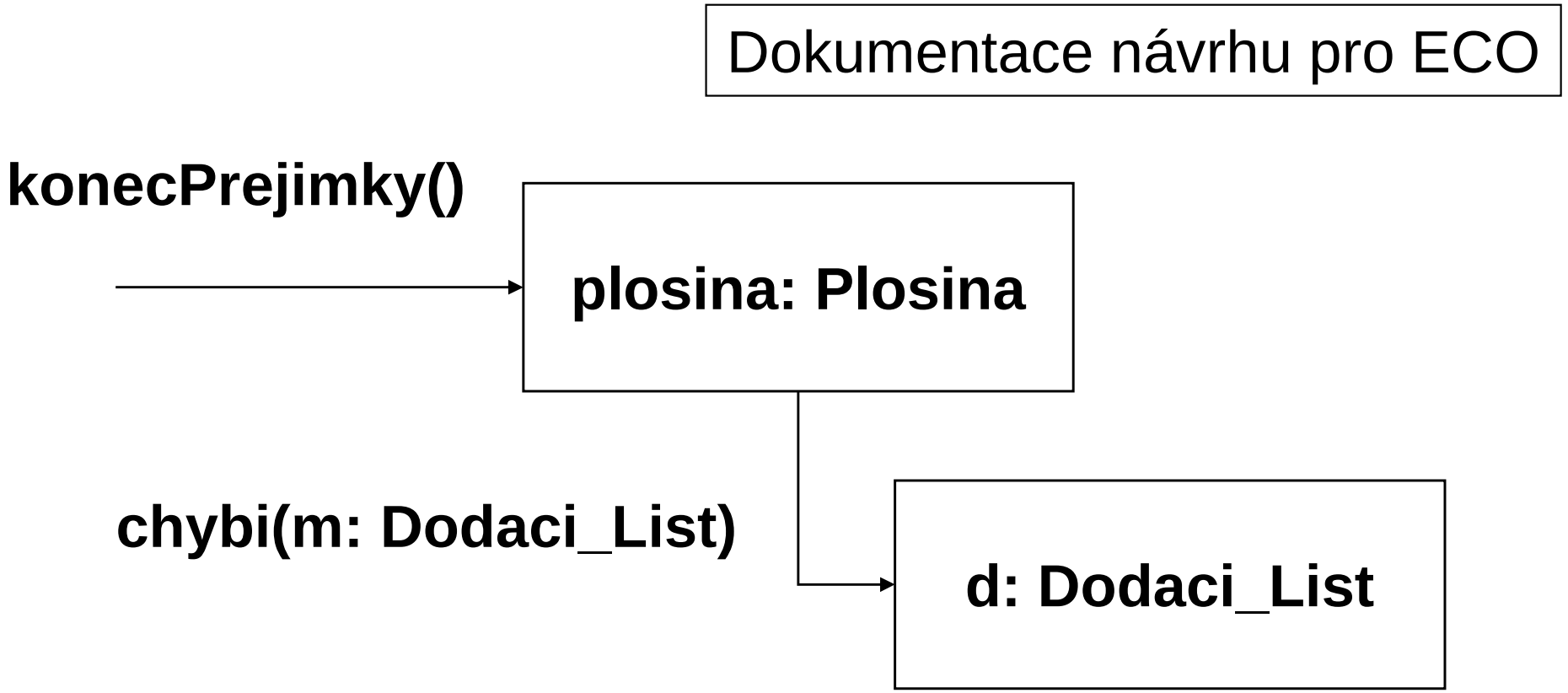
1. Výběr zodpovědného objektu

Diagram komunikace pro “konec přejímky“



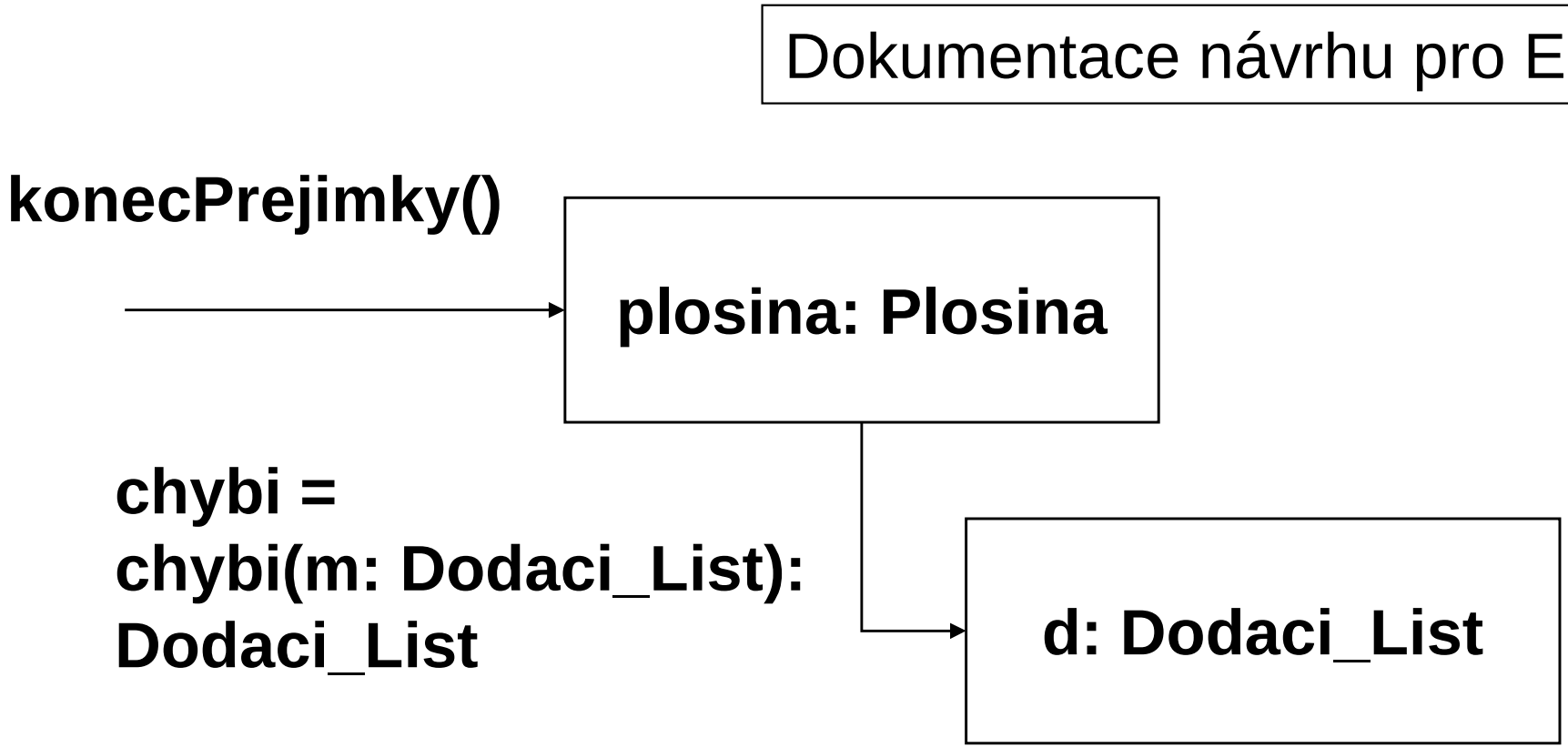
2. Výběr kooperujícího objektu

Diagram komunikace pro “konec přejímky“



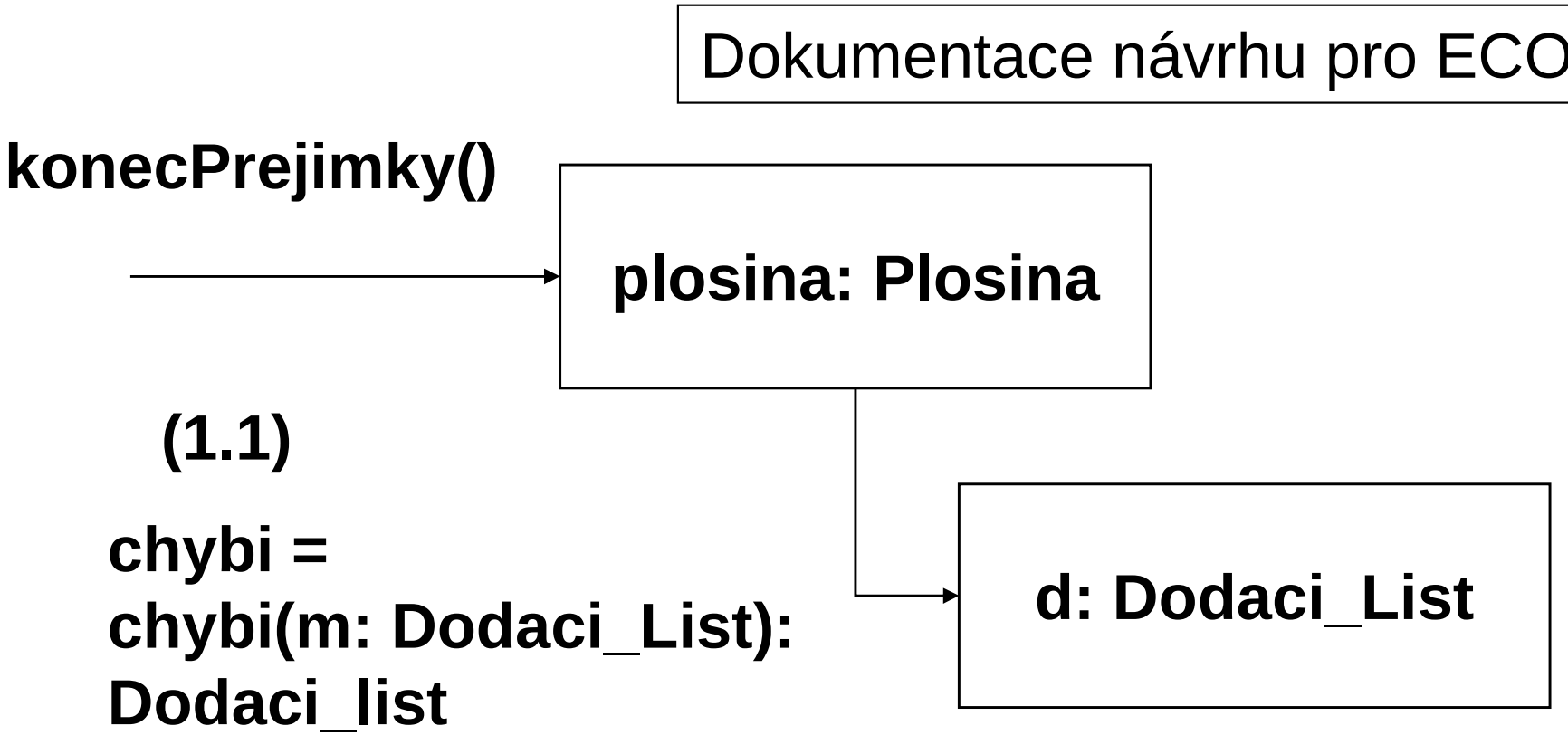
3. Výběr metody kooperujícího objektu

Diagram komunikace pro “konec přejímky“



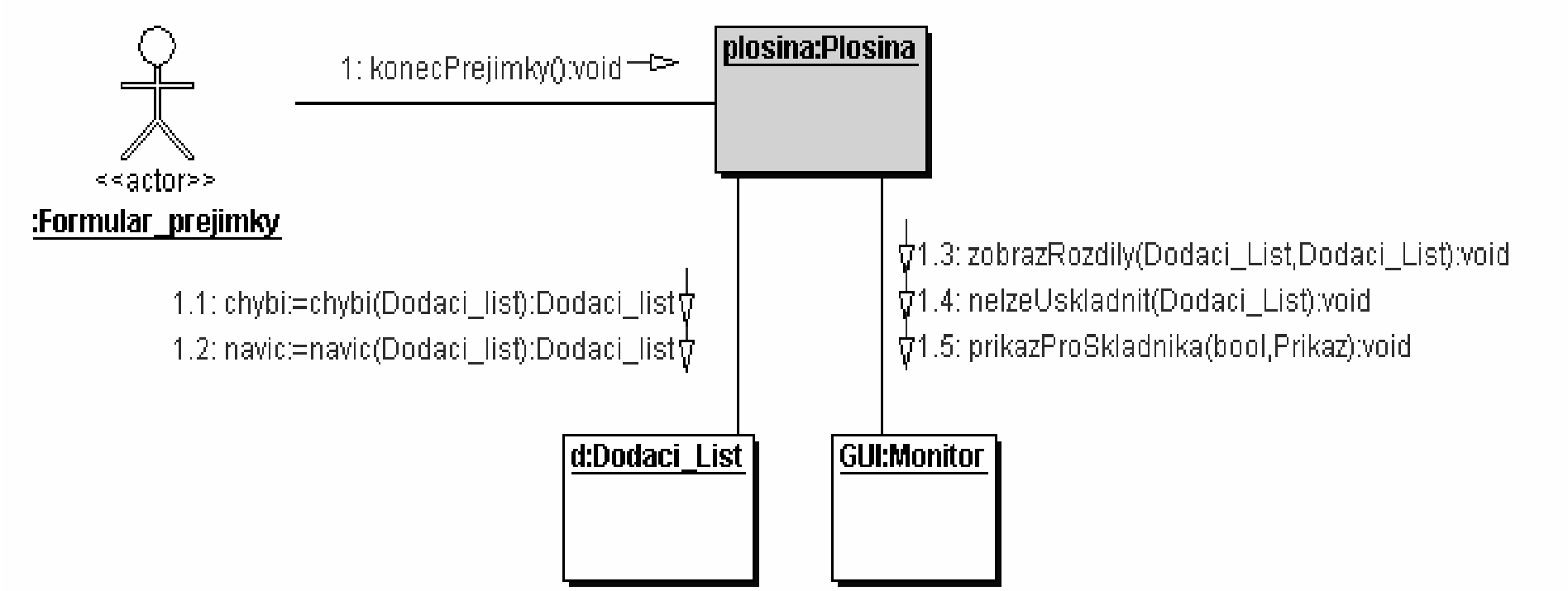
4. Bude něco vracet?

Diagram komunikace pro "konec přejímky"

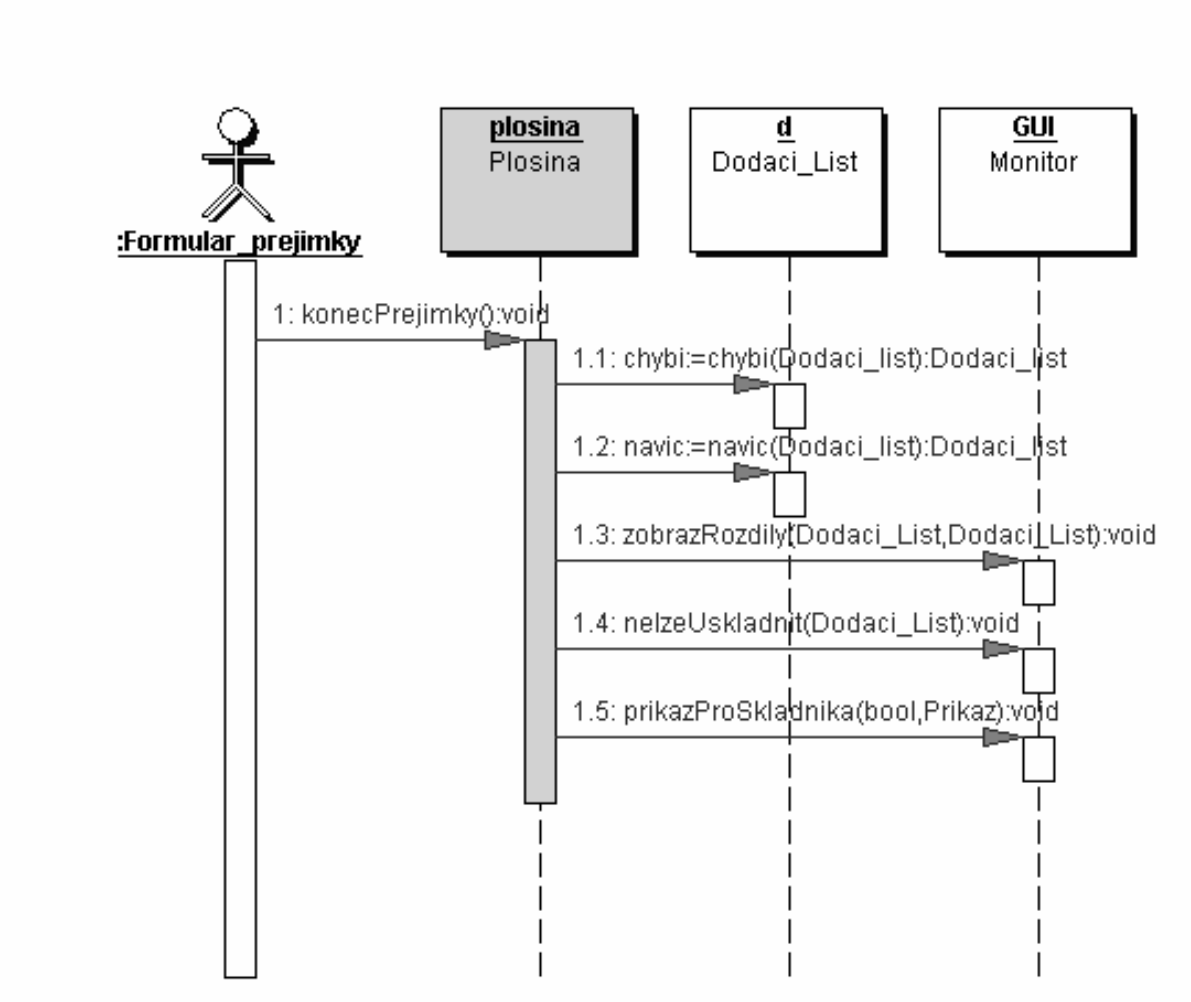


5. Stanovení pořadí volání

Výsledek


<<actor>>
:Formular_prejimky
1: konecPrejimky():void
plosina:Plosina
1.1: chybi:=chybi(Dodaci_list):Dodaci_list
1.2: navic:=navic(Dodaci_list):Dodaci_list
1.3: zobrazRozdily(Dodaci_List,Dodaci_List):void
1.4: nelzeUskladnit(Dodaci_List):void
1.5: prikazProSkladnika(bool,Prikaz):void
d:Dodaci_List
GUI:Monitor

Nebo jako scénář



Popis pro „chybí“

- Operace „chybí“ nepochází z analýzy, vznikla při návrhu, ale je nutno ji rovněž popsat (aby pak programátor věděl, co má dělat)

Popis pro “chybí”

Dokumentace návrhu pro ECO

Operation: chybi

Description: zjistí, co chybí při přejímce

Reads: supplied m: dodaci_list,
zadany_dodaci_list: dodaci_list

Changes: new chybi: dodaci_list

Sends:

Assumes:

Results:

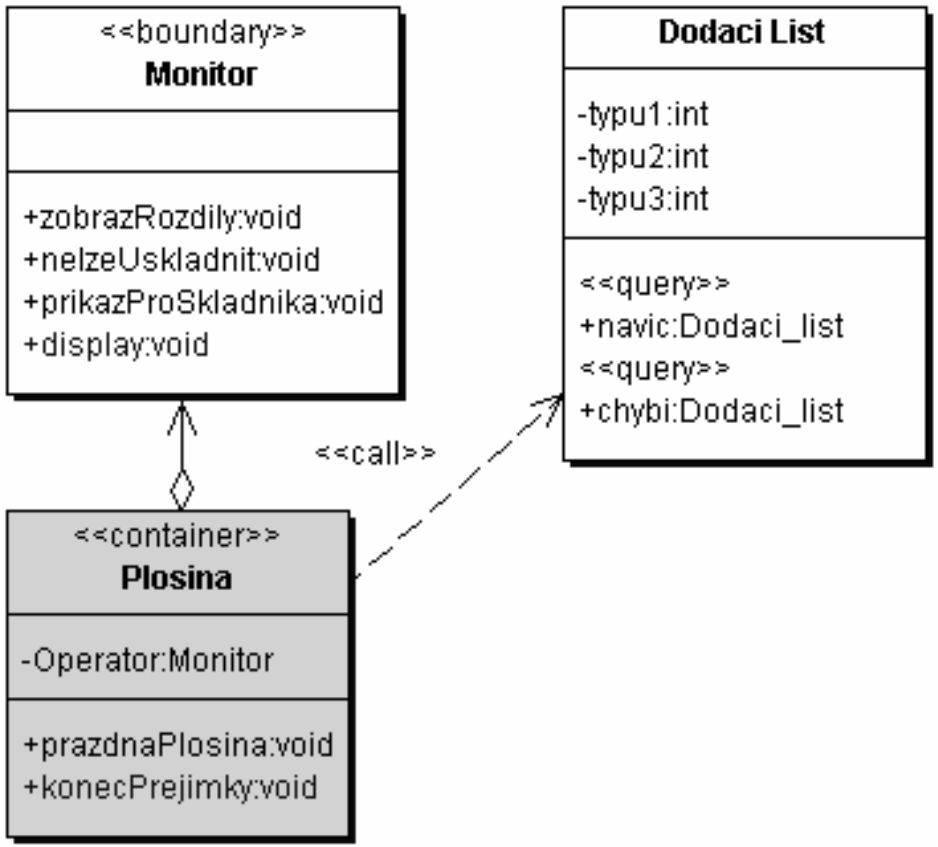
porovná zadaný dodací list m s dodacím listem zadany_dodaci_list

vytvoří objekt chybi: dodací list, obsahující barely které chybí

Postup návrhu ve Fusion

1. Pro každou operaci funkčního modelu nakresli diagram komunikace (včetně metod vzniklých při návrhu)
2. **Pro každou třídu datového modelu zakresli potřebnou viditelnost podle diagramů komunikace**
3. Pro každou třídu datového modelu vytvoř popis třídy
4. Doplň návrh o dědičnost

Viditelnost v UML



Postup návrhu ve Fusion

1. Pro každou operaci funkčního modelu nakresli diagram komunikace (včetně metod vzniklých při návrhu)
2. Pro každou třídu datového modelu zakresli potřebnou viditelnost podle diagramů komunikace
3. **Pro každou třídu datového modelu vytvoř popis třídy**
4. Doplň návrh o dědičnost

Popis tříd ve Fusion

```
class <jméno> [ isa <jméno> ]
// pro každý atribut
  [attribute][constant]<jméno>:[vazba]<typ>
  …
// pro každou metodu
  [method]<jméno>(<argumenty>)[:<typ>]
  …
endclass

vazba = [ref|(exclusive|shared)[bound]
argumenty = [<jméno>:<typ>](<jméno>:<typ>)*
typ = [col]<jméno>
```

Popis třídy “Sklad”

Dokumentace návrhu pro ECO

```
class Sklad isa Budova
/*   attribute sousedi:
          exclusive bound col Budova
          – dedi od Budovy */
     attribute kapacita:int
     attribute barely:
          exclusive bound col Barel
     method je_místo?(b: Barel): Bool
     method cti_cislo(): int
     method pridej(b: Barel)
     method kolik?(t: TypChem): int
     …
endclass
```

Popis třídy “Plosina”

Dokumentace návrhu pro ECO

```
class Plosina
      attribute sklady:
            ref shared col Sklad
      attribute Operator:
            share bound Monitor
      attribute dodaci_list:
            ref shared Dodaci_list
      method konec_prejimky()
      method prazdna_plosina()
      method cti_obsah(): col Barel
      …
endclass
```

Implementace třídy “Plošina”

Dokumentace implementace ECO

```
class Plosina {
protected:
     Set<Sklad &> sklady; // ref shared col
     Monitor *Operator; // share bound
          Dodaci_list &dodaci_list; //  ref shared
public:
     Plosina();
     ~Plosina();
     void konec_prejimky();
     void prazdna_plosina();
     List<Barel> &cti_obsah();
     …
endclass
```

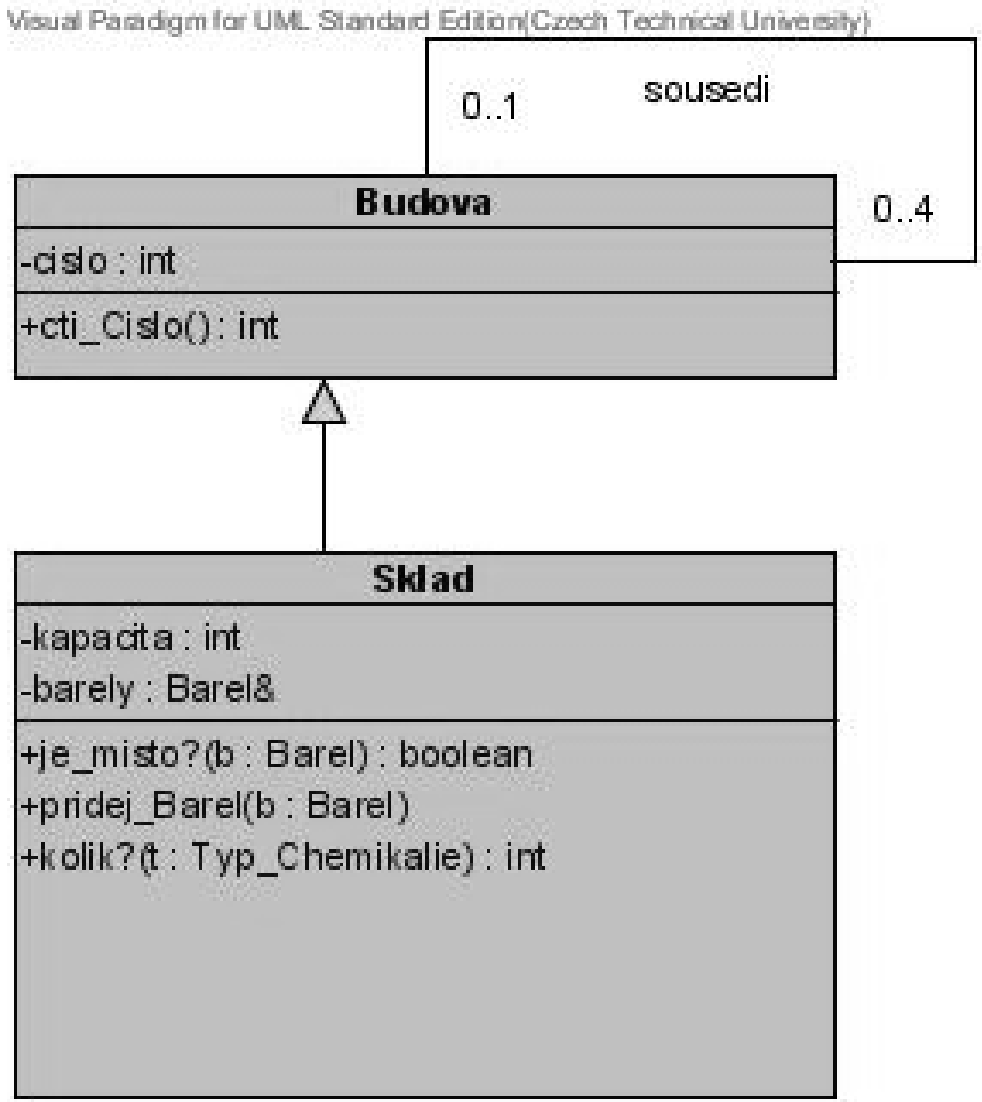
Postup návrhu ve Fusion

1. Pro každou operaci funkčního modelu nakresli diagram komunikace (včetně metod vzniklých při návrhu)
2. Pro každou třídu datového modelu zakresli potřebnou viditelnost podle diagramů komunikace
3. Pro každou třídu datového modelu vytvoř popis třídy
4. **Doplň návrh o dědičnost**

Dva zdroje dědičnosti

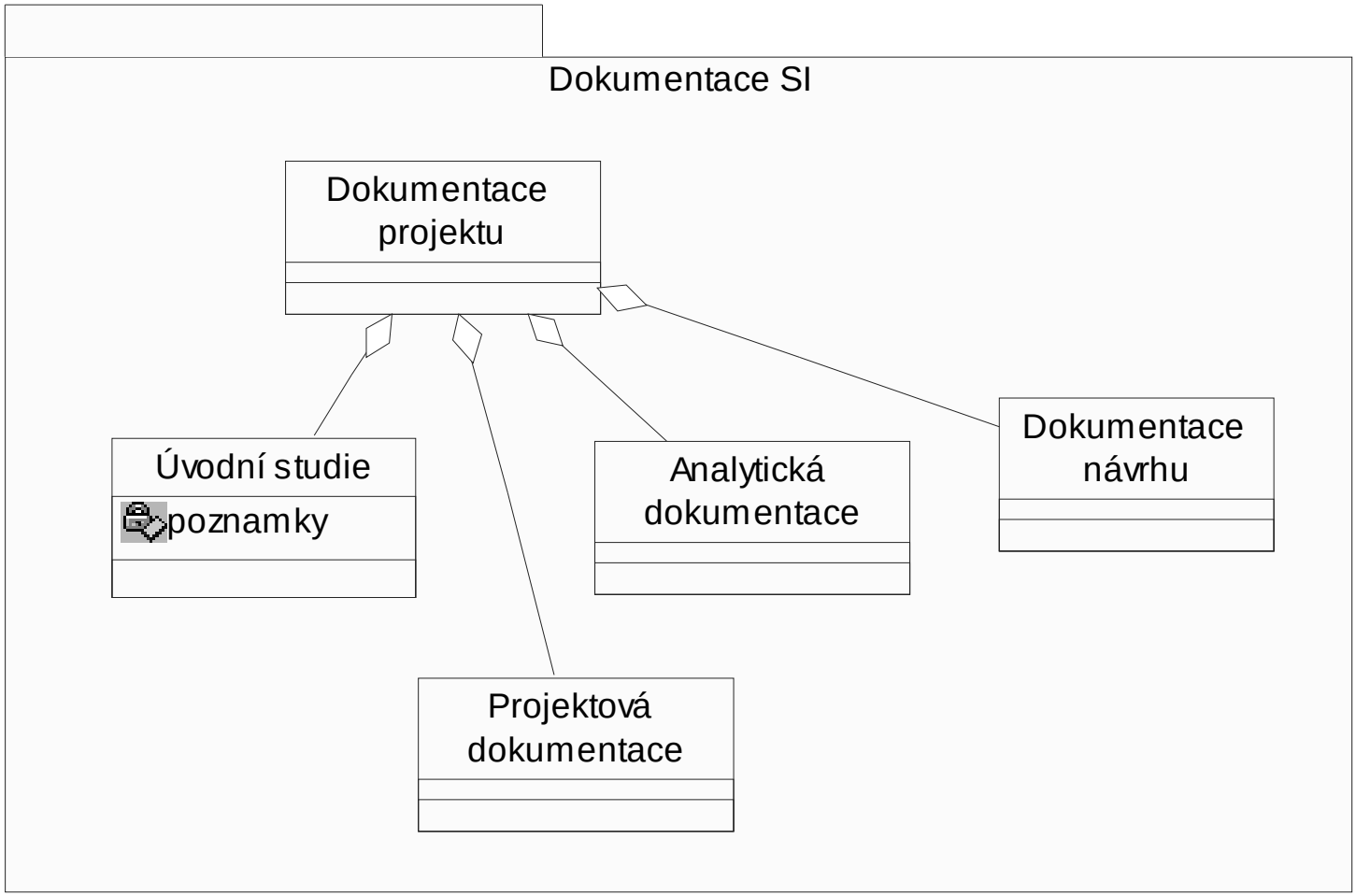
- Společné a speciální vlastnosti dat odhalené při analýze
- Vlastnosti zděděné z použitých knihoven

Popis třídy “Sklad”



Dokumentace
návrhu
pro ECO

Co bude výstupem SIN?



The End

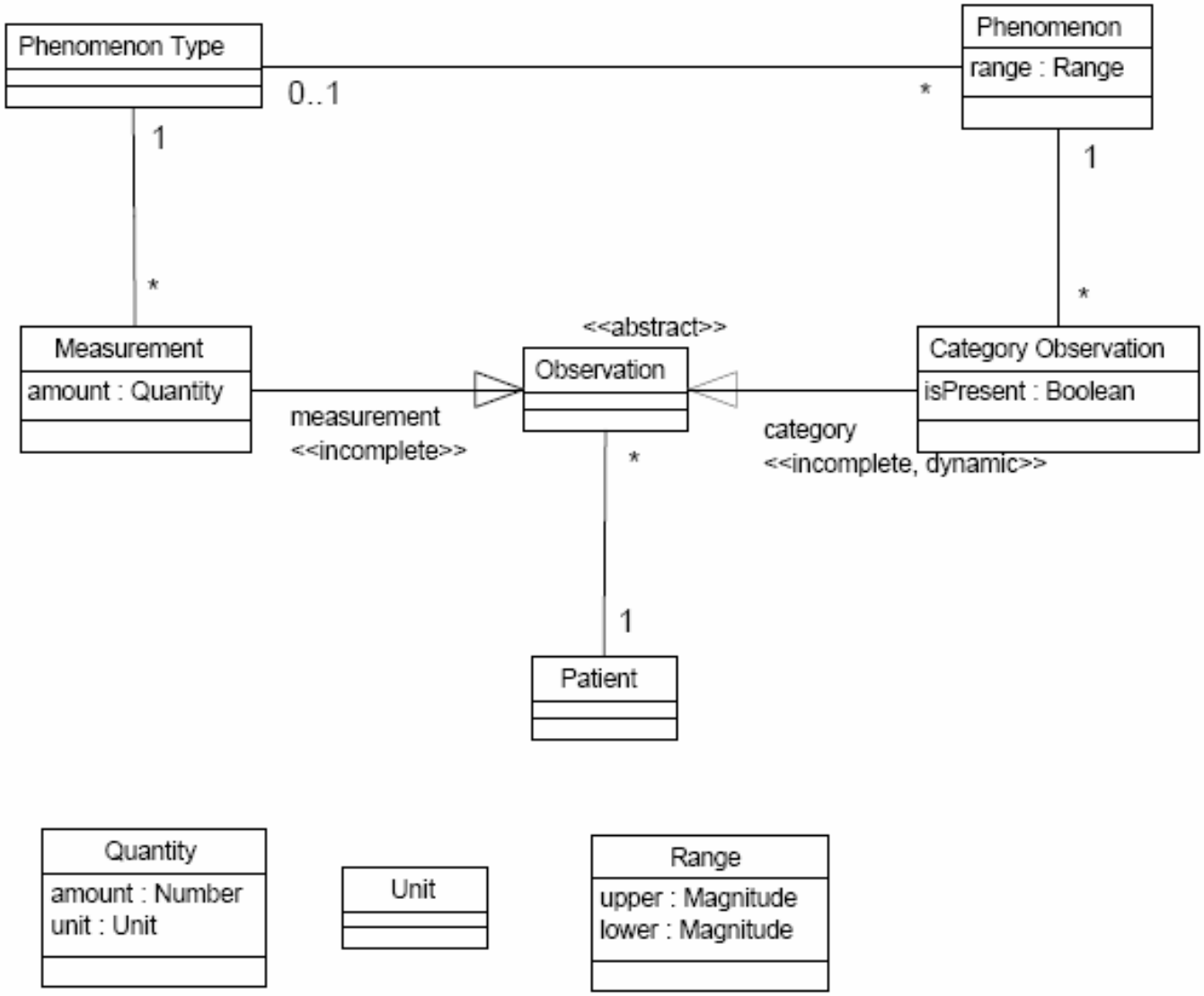
Návrhové vzory (Design Patterns)

JAK se to obvykle dělá

Typy vzorů

- Analytické vzory (konceptuální data banky)
- Architektonické vzory (vrstvená architektura, MVC)
- Návrhové vzory (sekvenční průchod kolekcí – iterátor)
- Programovací vzory (implementace seznamu pomocí pole)

Analytický vzor: Měření a pozorování

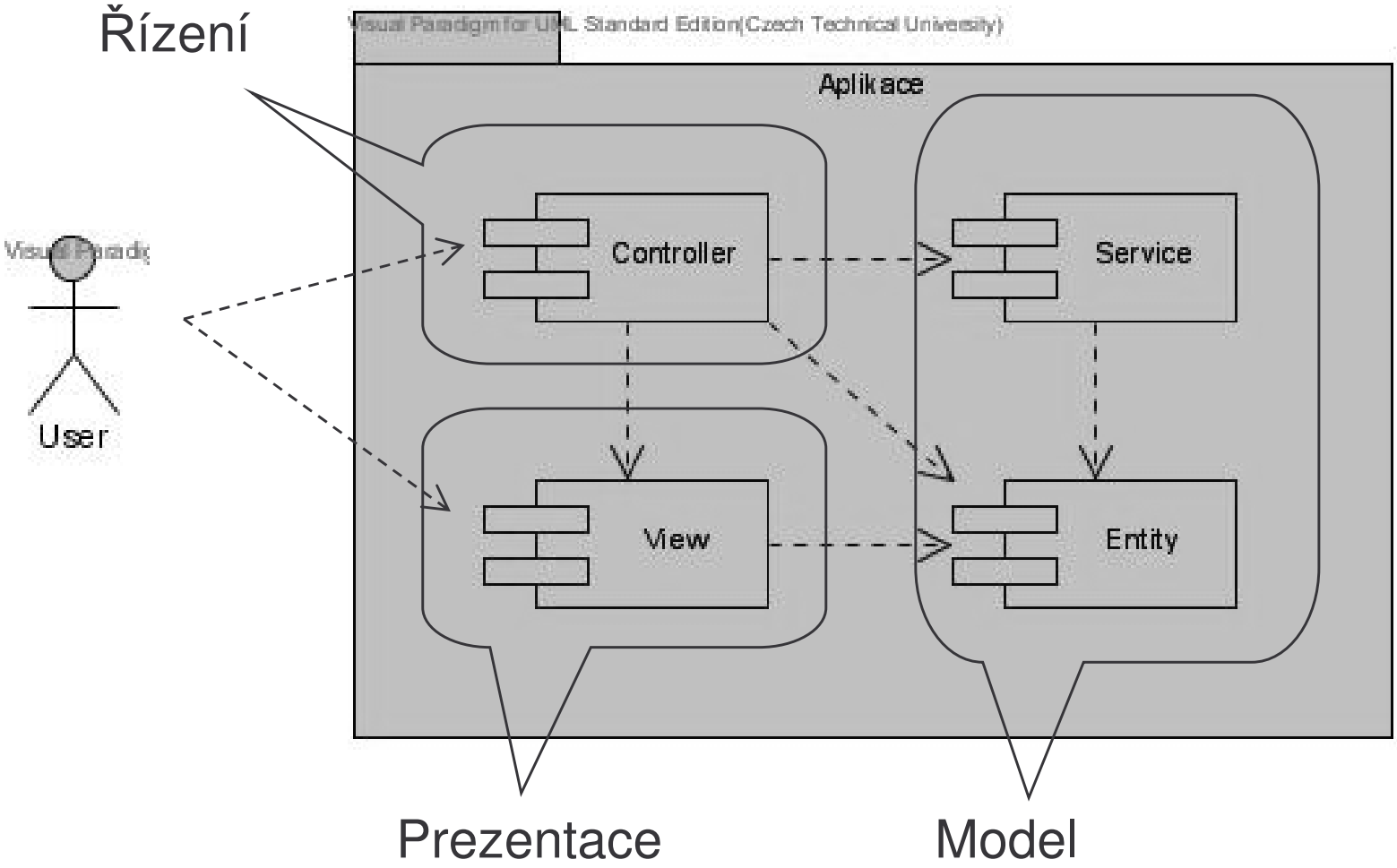


Zdroj: http://www.martinfowler.com/apsupp/appfacades.pdf

Architektonické vzory

- Vzor pro konstrukci, např. snaha o konstrukci dostatečně pružné aplikace
- Vyplatí se oddělit správu dat obhospodařovaných danou aplikací, jejich prezentaci a vlastní řízení aplikace
- Toto rozdělení pak vyjádřit přímo architekturou aplikace

Architektonický vzor MVC



Co jsou návrhové vzory?

- Standardní řešení častých problémů v návrhu softwaru
- Příklad: Sada vzorů GoF = Gang of Four
- Co nejsou návrhové vzory?
  - idiomy = ustálené fragmenty kódu – konkrétní prog. jazyk, nejde o návrh
  - konvence kódu – nejde o návrh
  - algoritmy – řeší výpočet, ne návrh

Co to je návrhový vzor?

- Christopher Alexander: *”Každý vzor popisuje často se vyskytující problém a poté popisuje jádro řešení tohoto problému tak, aby bylo možno toto řešení opakovaně využívat, bez toho, že bychom stejnou věc dělali dvakrát".*
- Přestože se tato definice týká návrhu budov, totéž platí pro návrhové vzory při tvorbě programů. Řešení je zde vyjádřeno pomocí potřebných objektů a způsobu jejich komunikace.

Zdroje příkladů:

- Použity vzory GoF (Gamma, Helm, Johnson, Vlisidess: Design Patterns - The Elements of reusable object-oriented software. 1995)
- Návrhové vzory I., FEL 2005, Tomáš Richta
- Návrhové vzory II., FEL 2005, Petr Šlégr

Obecné části návrhových vzorů

- Jméno vzoru
- Popis problému
- Popis řešení
- Důsledky použití

Jméno vzoru

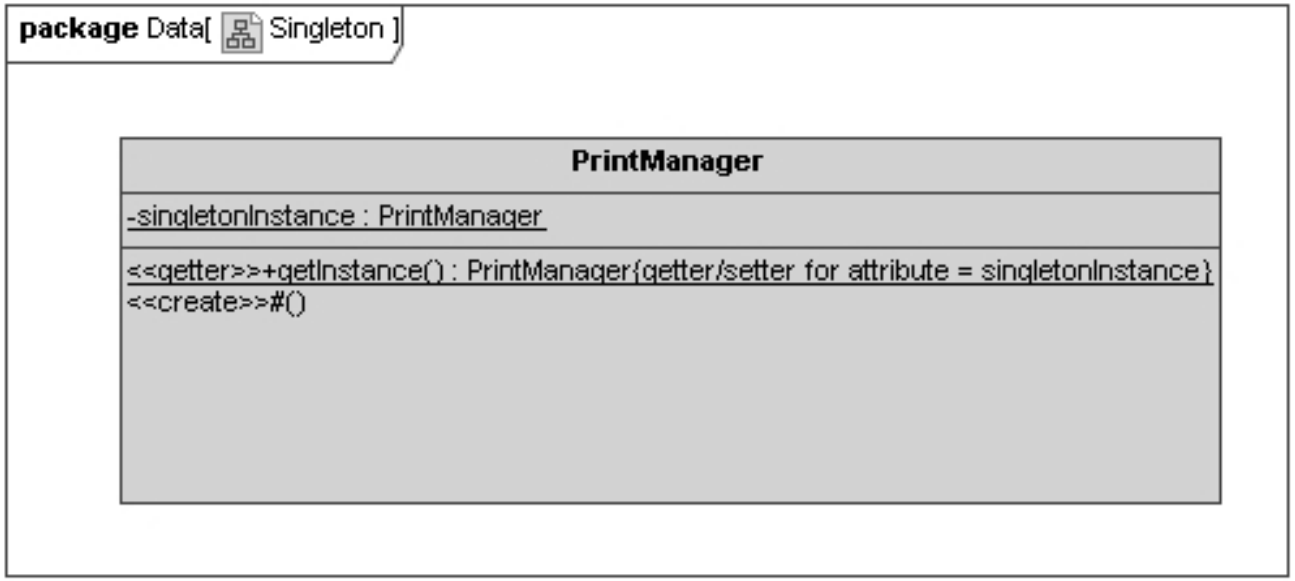
- Termín, přes který se na vzor odvoláváme.
- Aby se v katalogu vzorů dobře a intuitivně hledalo, je volba jména důležitá a obtížná.
- Př.: Proxy (zástupce), Iterator (iterátor)

Popis problému pro vzor

- Vyjadřuje situaci, kde se použití vzoru hodí.
- Může se stanovit příkladem, seznamem podmínek, které musí platit, apod.

Problém

- Je třeba, aby v aplikaci existovala pouze jedna instance nějaké třídy. Jak to zajistit?



Singleton

```
public class PrintManager {
    private static PrintManager instance;

    private PrintManager () { /* ... */  }

    // jina jmena: getDefault(), getPrintManager,...
    public static synchronized PrintManager getInstance () {
        if (instance == null) { // "lazy" inicializace
            instance = new PrintManager ();
        }
        return instance;
    }

    // vlastni instancni metody...
}
```

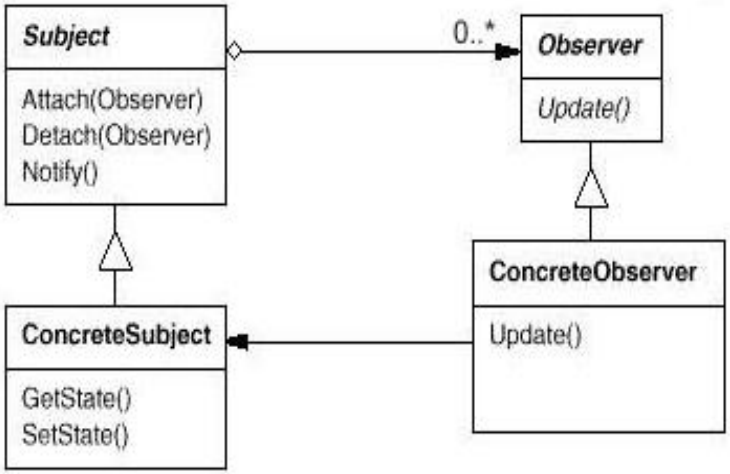
Singleton

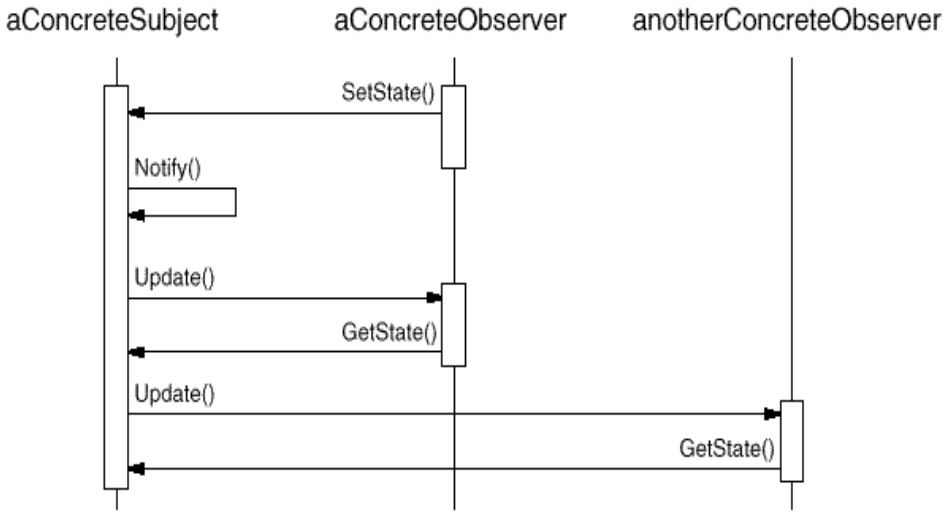
- Vlastnosti:
  - konkrétní typ poskytnuté instance lze určit za běhu
  - poskytovanou instanci lze měnit bez změn klientského kódu
  - snadná změna logiky tvorby - např. povolení více instancí
- Příklad v J2SE: `java.lang.Runtime`

Problém

- Na stavu jednoho objektu závisí řada jiných objektů.
- Při změně stavu objektu potřebují být závislé objekty automaticky vyrozuměny.
- Jak toho docílit?

Observer/Pozorovatel



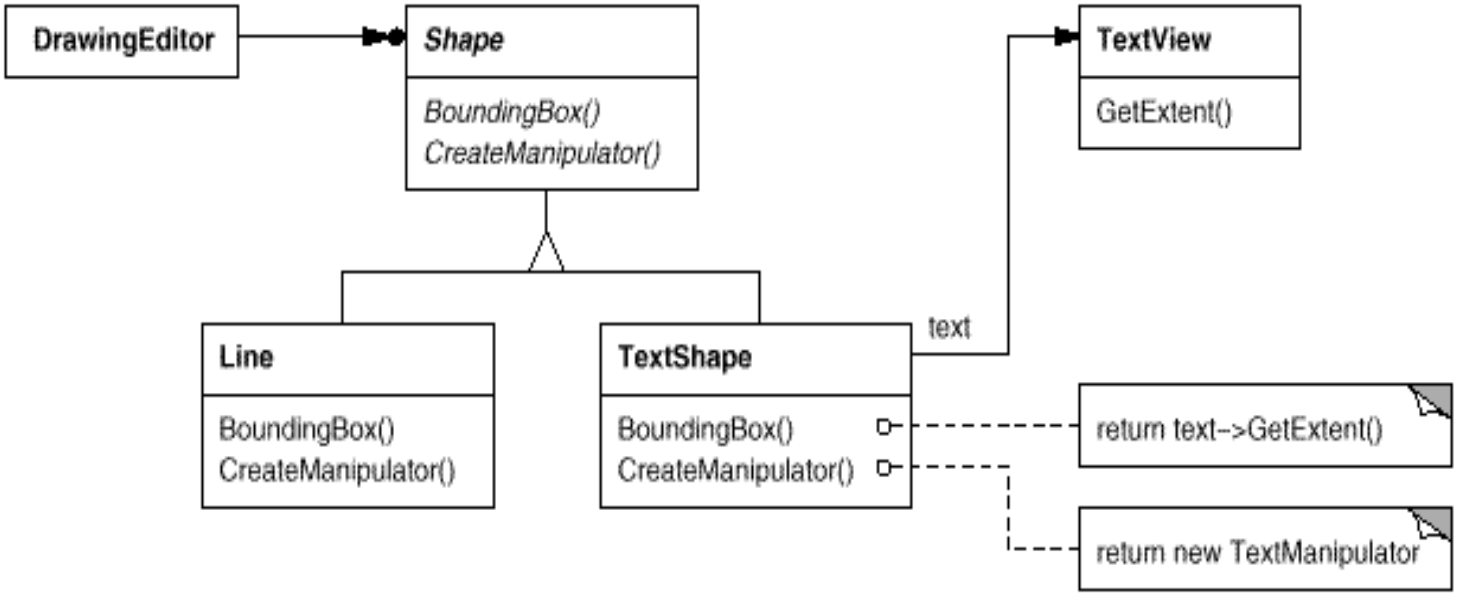


Problém

- Chceme použít nějakou užitečnou třídu, která ale má jiné rozhraní, než očekáváme. Co s tím?

Adapter/Adaptér

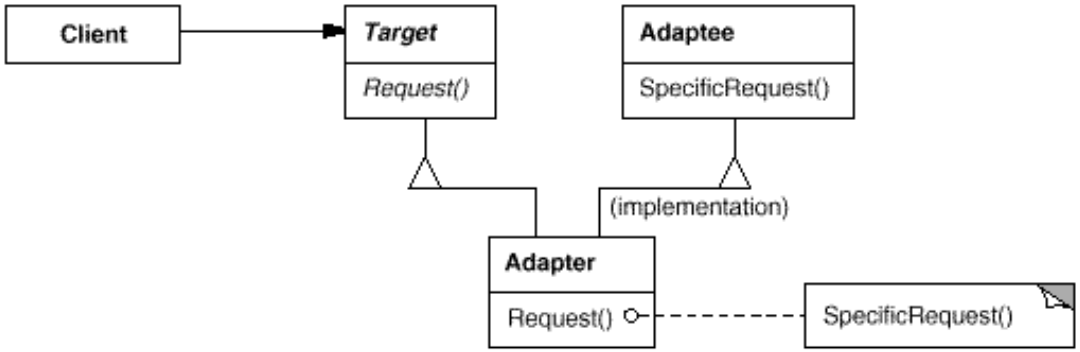
- Také wrapper/obal



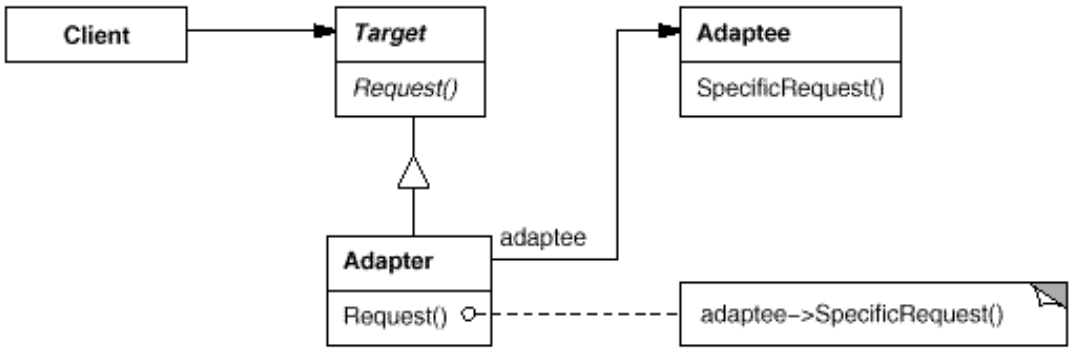
Obrázek převzat z [GoF]

Adapter/Adaptér

- Třídní implementace: vícenásobná dědičnost



- Instanční implementace: kompozice

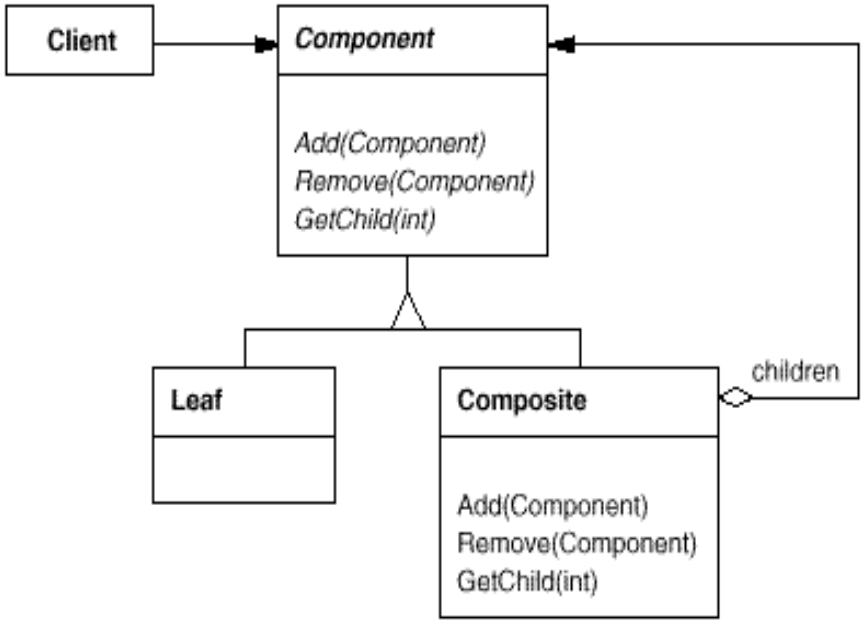


Obrázky převzaty z [GoF]

Problém

- Máme hierarchicky organizovaný celek s jehož částmi chceme manipulovat jednotným způsobem. Jak na to?

Composite/Kompozice



Obrázek převzat z [GoF]

- Použití:
  - grafická uživatelská rozhraní
  - strukturované dokumenty (XML, HTML,...)
  - stromově organizovaná data...

Návrhové vzory: kategorie

- Vzory pro **vytváření** a **manipulaci** s reprezentací informace (např. Abstraktní továrna, Proxy)
- Vzory **strukturální** vyjadřující strukturu implementace (např. Adaptér, Kompozice)
- Vzory pro **chování** (např. Iterátor)

Taxonomie návrhových vzorů

	Tvorba Vznik nových objektů	**Struktura** Skládání tříd/objektů	**Chování** Interakce tříd/objektů
Třída Statické vztahy	Tovární metoda	Adaptér (třídní)	Šablonová metoda
Instance Dynamické vztahy	Jedináček	Adaptér (objektový), Kompozice	Strategie, Pozorovatel, Iterátor

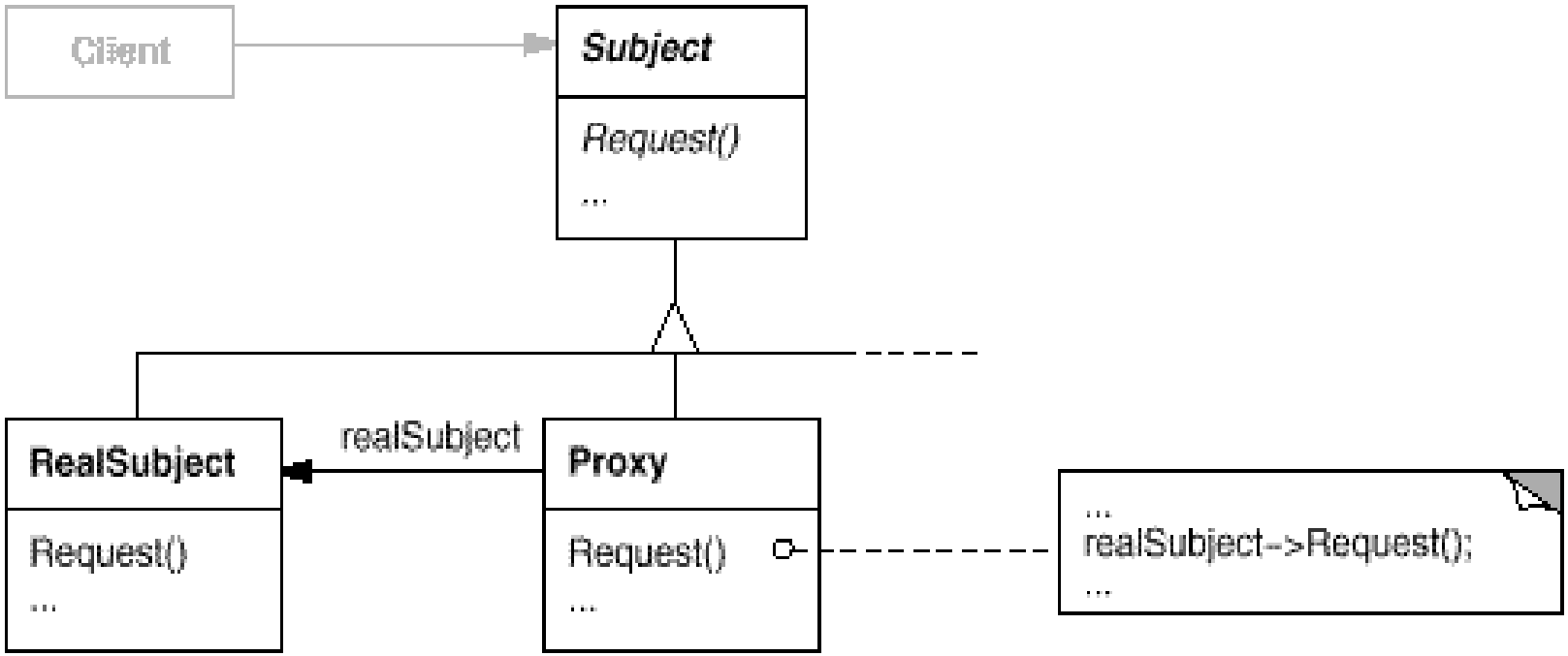
Př: Návrhový vzor "Proxy"

- Deklarace záměru:
  - pokud potřebujeme zástupce objektu, neboť vlastní objekt je někde jinde, je těžko přístupný, má být chráněn, ...
- Motivační příklad:
  - editor dokumentu by měl umět pracovat s obrázky, ale zobrazení obrázku je náročné - často postačí náhradník

Popis řešení pro vzor

- Popisuje elementy použité při řešení a jejich vztahy.
- Nejedná se o konkrétní implementaci, neboť vzor je pouze šablonou pro řešení.

Struktura vzoru Proxy

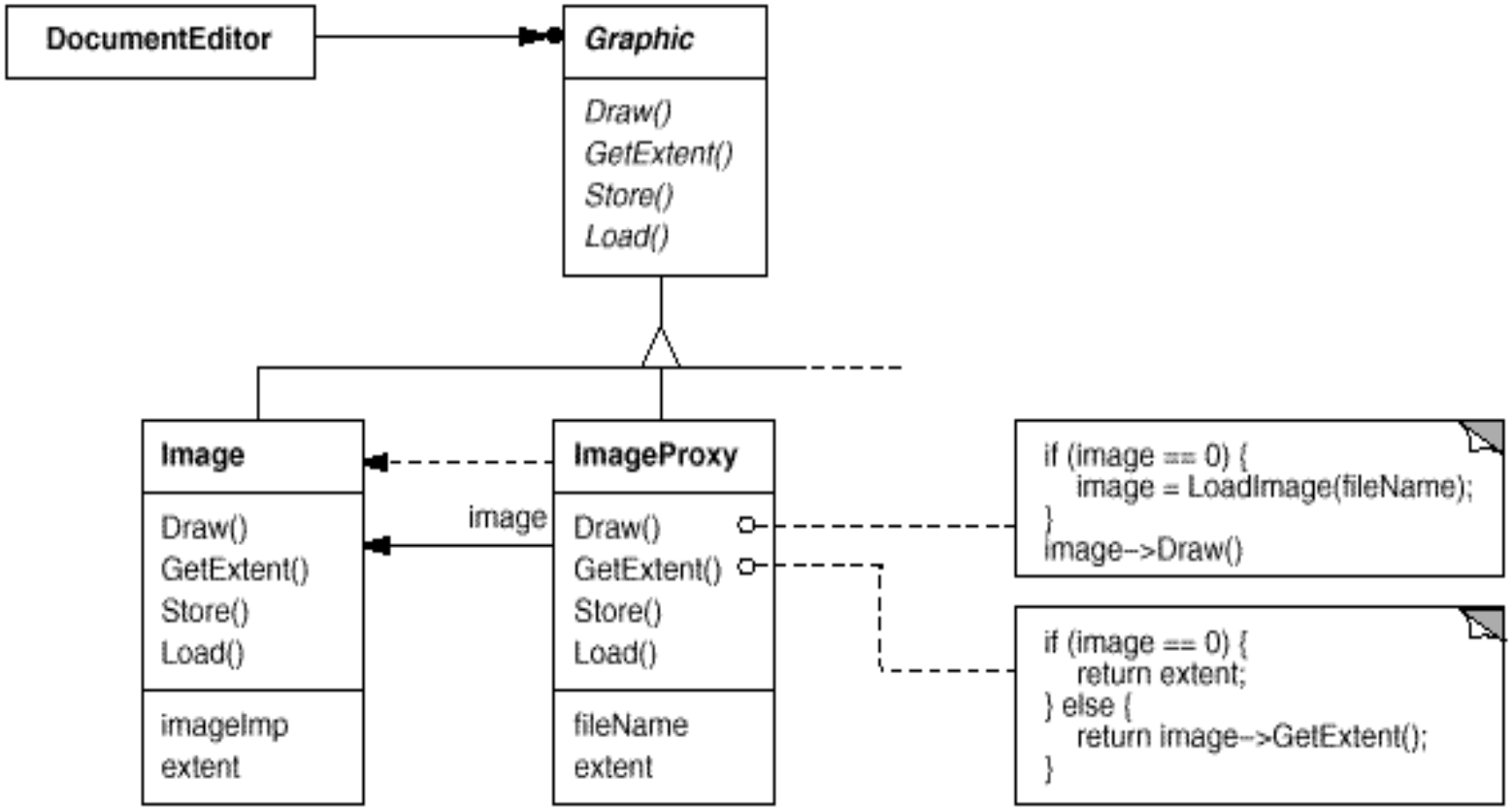


Obrázek převzat z [GoF]

Důsledky použití vzoru

- Rozmanité důsledky, které s sebou použití vzoru přináší - např. časové a prostorové nároky.
- Poslouží dobře i při výběru alternativ.

Příklad použití vzoru Proxy



Obrázek převzat z [GoF]

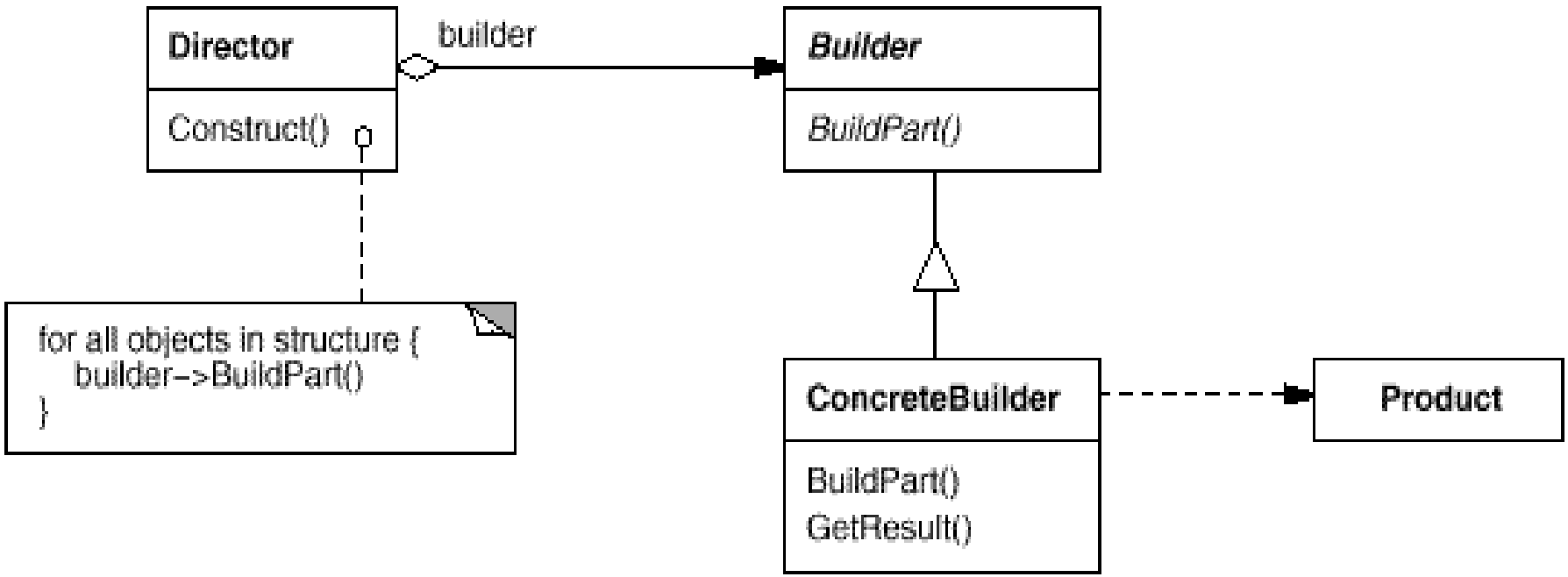
Příbuzné strukturální vzory

- **Proxy** zastupuje objekt a poskytuje stejné rozhranní jako on
- **Adapter** mění rozhranní, které objekt poskytuje
- **Dekorátor** něco přidává

Vzor: Builder

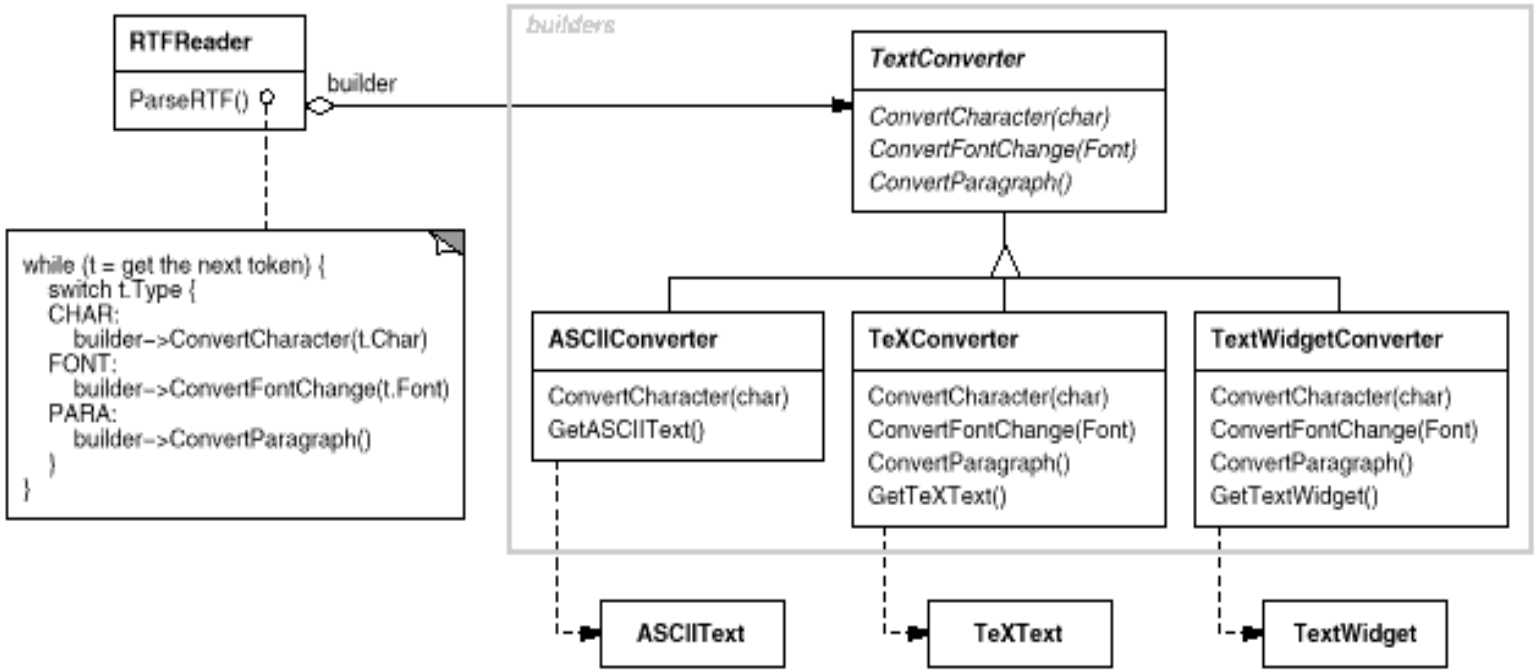
- **Deklarace záměru:**
  - oddělení konstrukce složitého objektu od jeho reprezentace
  - pokud má být algoritmus pro vytváření složitého objektu nezávislý na vytváření částí
  - pokud lze objekt reprezentovat různými způsoby
- **Motivační příklad:**
  - editor dokumentu v RTF by měl umět pracovat s různými reprezentacemi textu (ASCII, TeX, …)

Struktura vzoru Builder



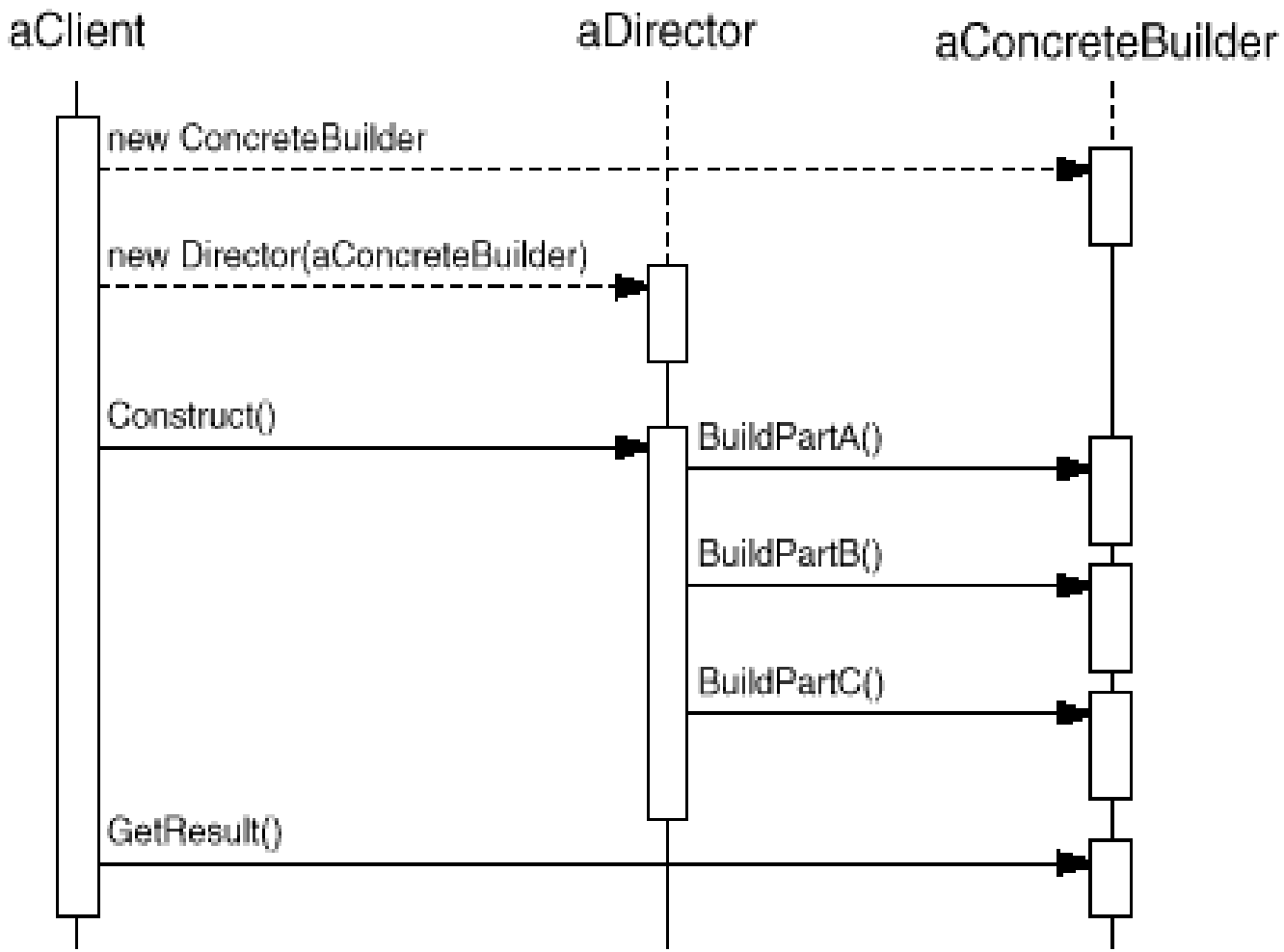
Obrázek převzat z [GoF]

Příklad použití vzoru Builder



Obrázek převzat z [GoF]

Kolaborace při použití Builder

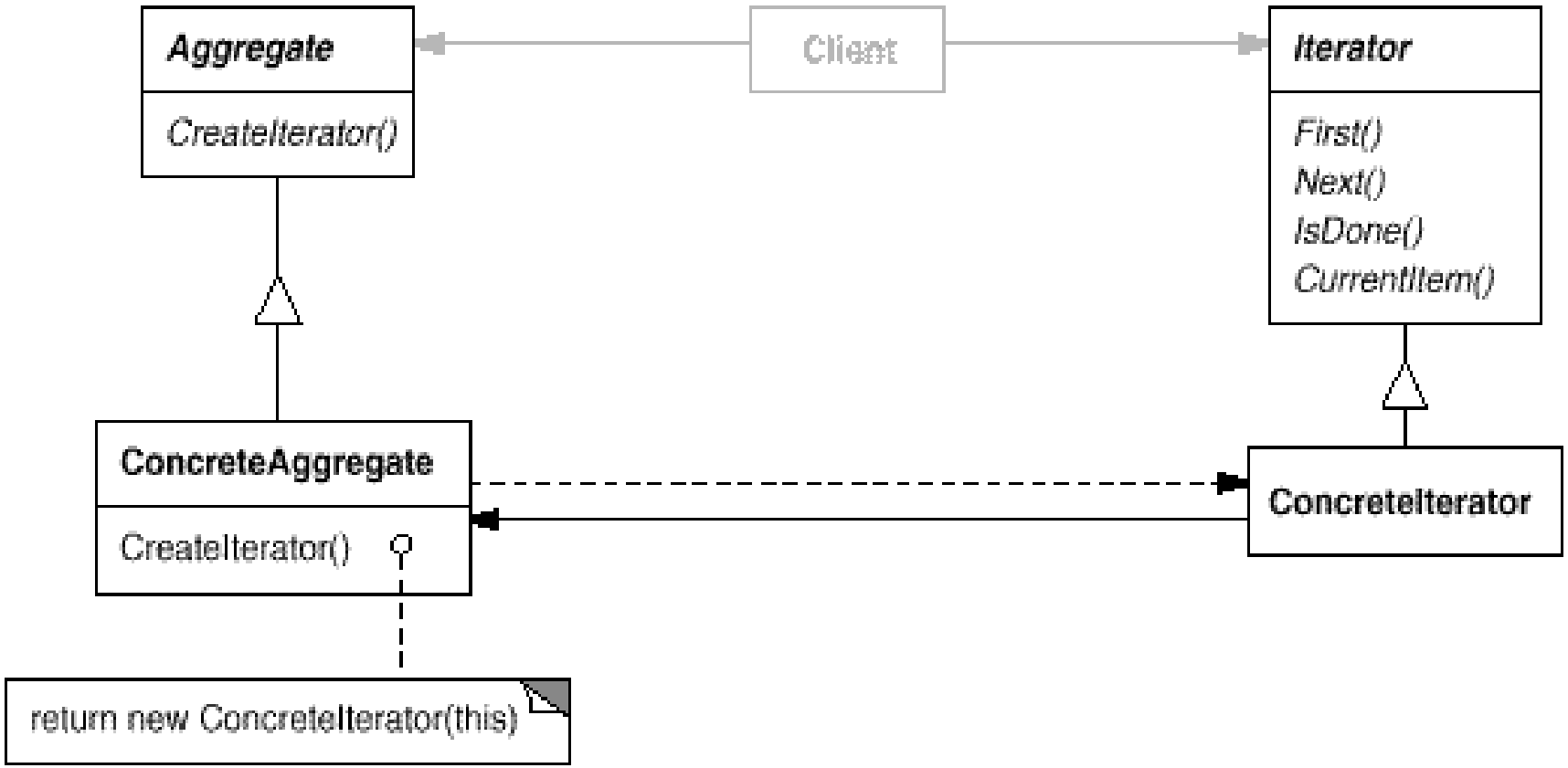


Obrázek převzat z [GoF]

Vzor: Iterator (Cursor)

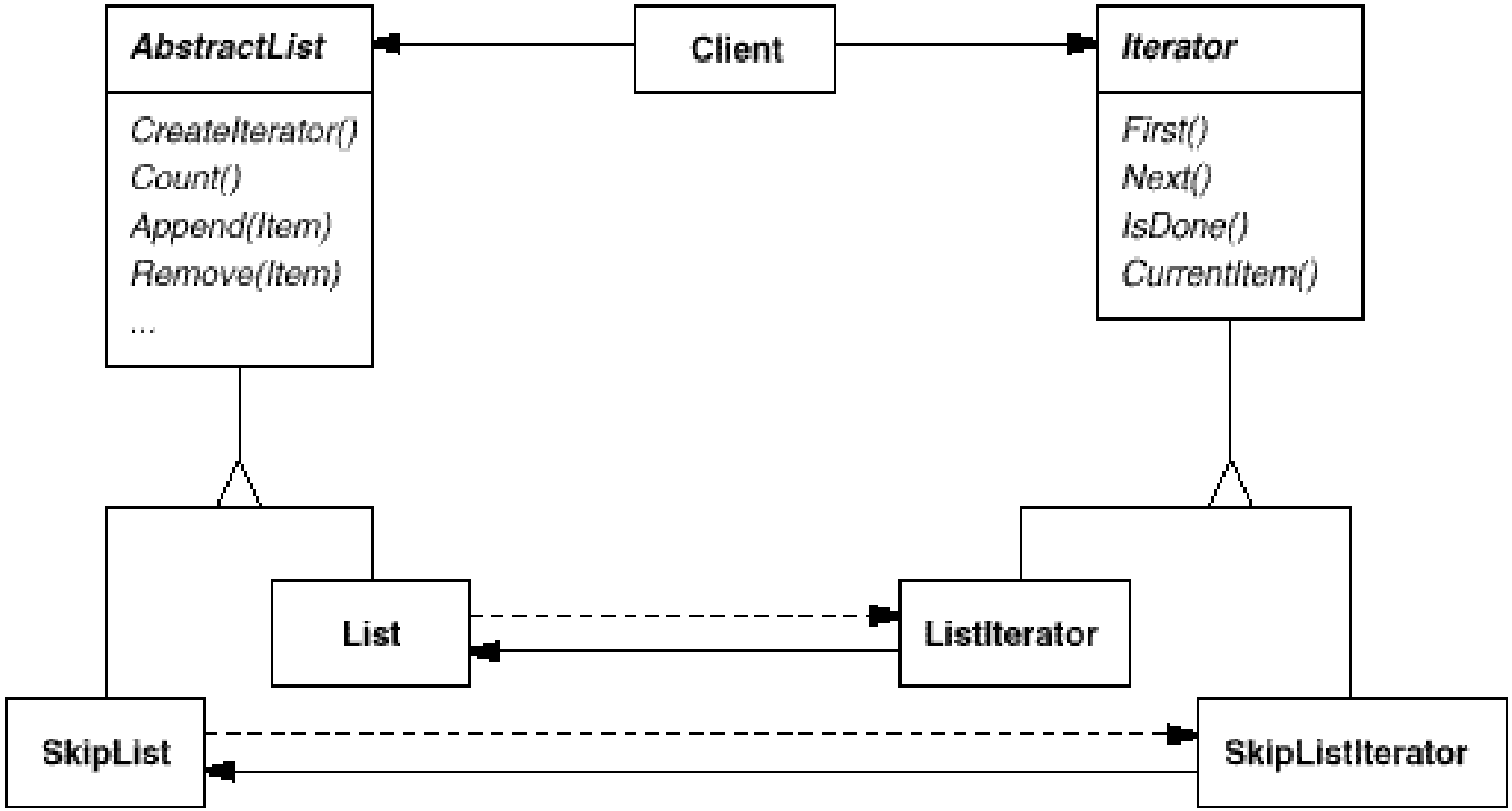
- **Deklarace záměru:**
  - pro sekvenční přístup ke složkám složeného objektu bez ohledu na reprezentaci - pokud má být procházen seznam aniž se zabýváme jeho reprezentací (uniformní přístup pro traverzování agregovaných struktur)
- **Motivační příklad:**
  - sekvenční průchod seznamy

Struktura pro Iterator



Obrázek převzat z [GoF]

Příklad použití Iterator

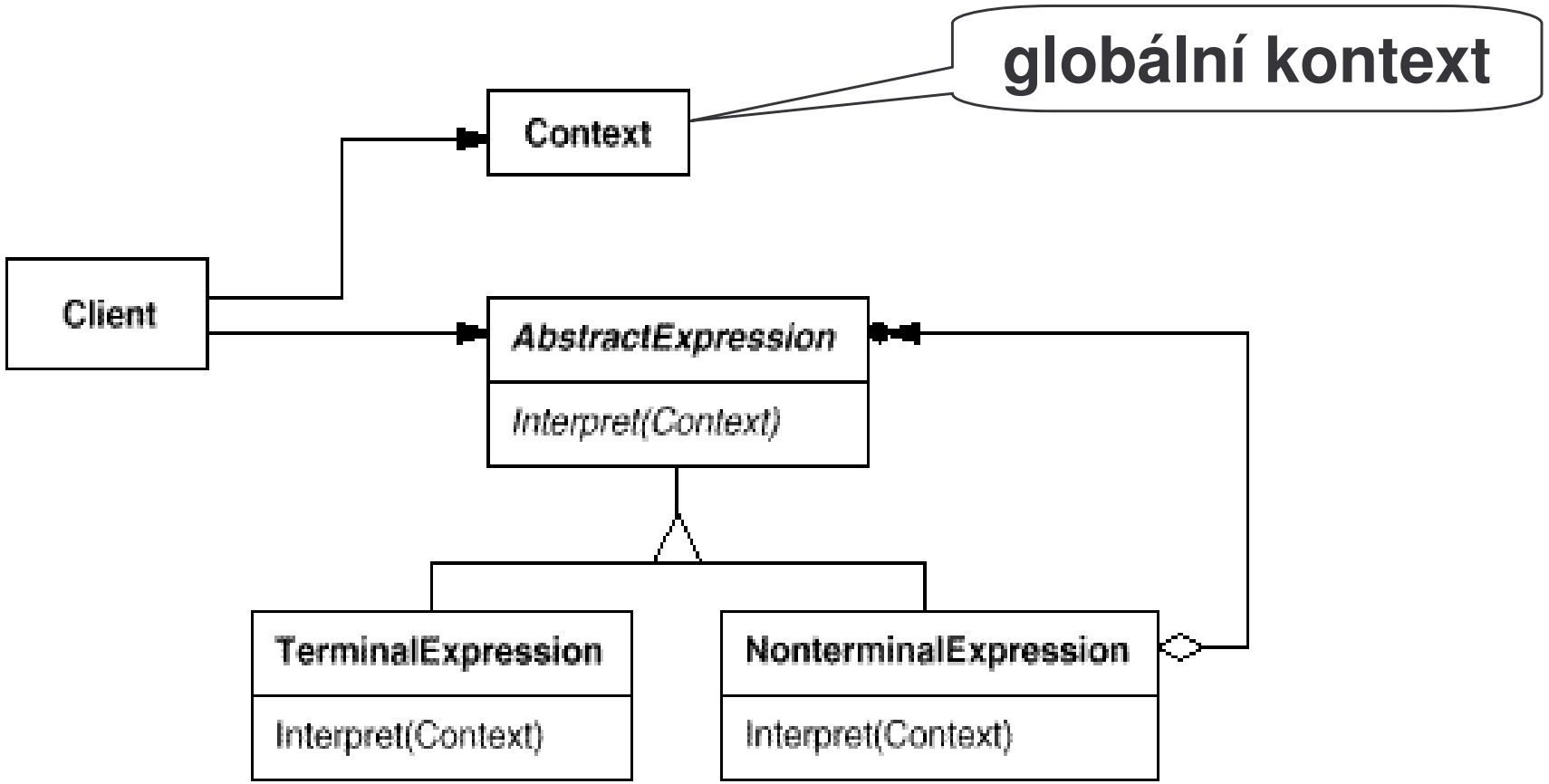


Obrázek převzat z [GoF]

Vzor: Interpret

- Deklarace záměru:
  - máme zadánu gramatiku jazyka, jehož věty chceme interpretovat
- Motivační příklad:
  - hledání vzorku zadaného regulárním výrazem v řetězci (známe gramatiku regulárního výrazu)

Struktura vzoru “Interpret”



Obrázek převzat z [GoF]

Gramatika pro regulární výraz

expression ::= literal | alternation | sequence

| repetition | '(' expression ')'

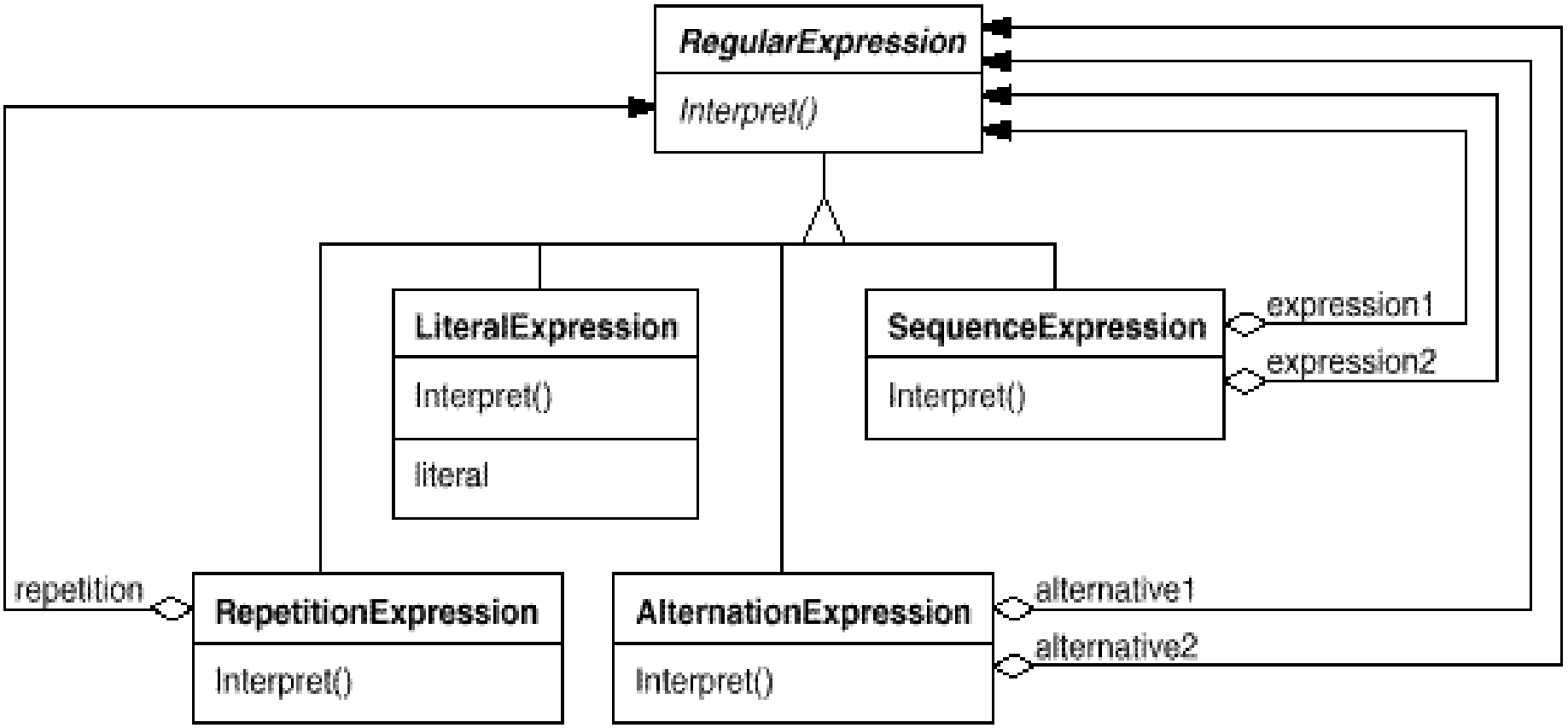
alternation ::= expression '|' expression

sequence ::= expression '&' expression

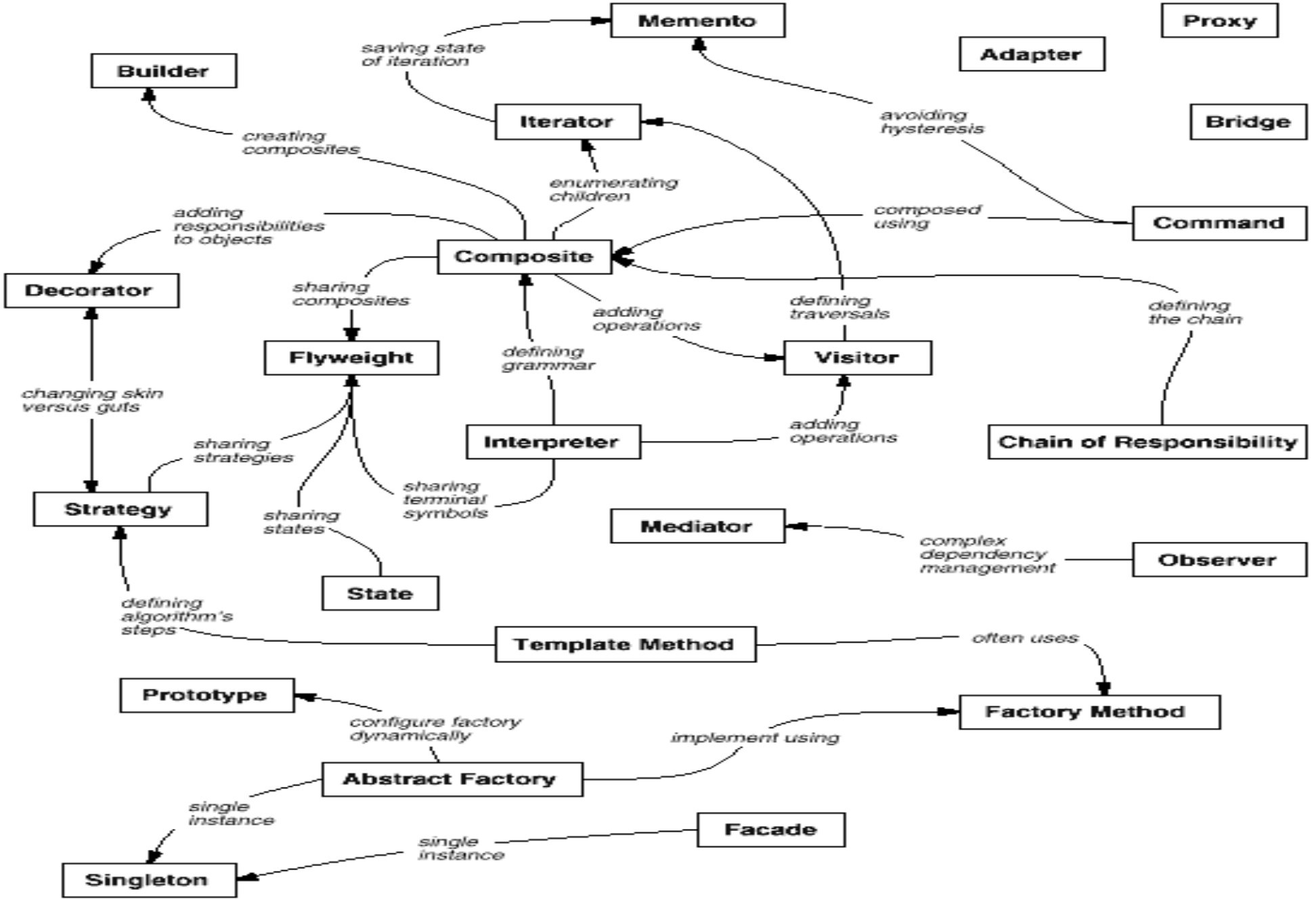
repetition ::= expression '*'

literal ::= 'a' | 'b' | 'c' | ... { 'a' | 'b' | 'c' | ... }*

“Interpret” pro reg. výrazy



Obrázek převzat z [GoF]


Memento
Proxy
saving state of iteration
Builder
Adapter
Iterator
avoiding hysteresis
Bridge
creating composites
enumerating children
adding responsibilities to objects
composed using
Command
Composite
Decorator
sharing composites
adding operations
defining traversals
defining the chain
Flyweight
defining grammar
Visitor
changing skin versus guts
sharing strategies
Interpreter
adding operations
Chain of Responsibility
Strategy
sharing terminal symbols
sharing states
Mediator
complex dependency management
Observer
State
defining algorithm's steps
Template Method
often uses
Prototype
Factory Method
configure factory dynamically
implement using
Abstract Factory
single instance
Facade
single instance
Singleton

The End

Vojtěch Merunka

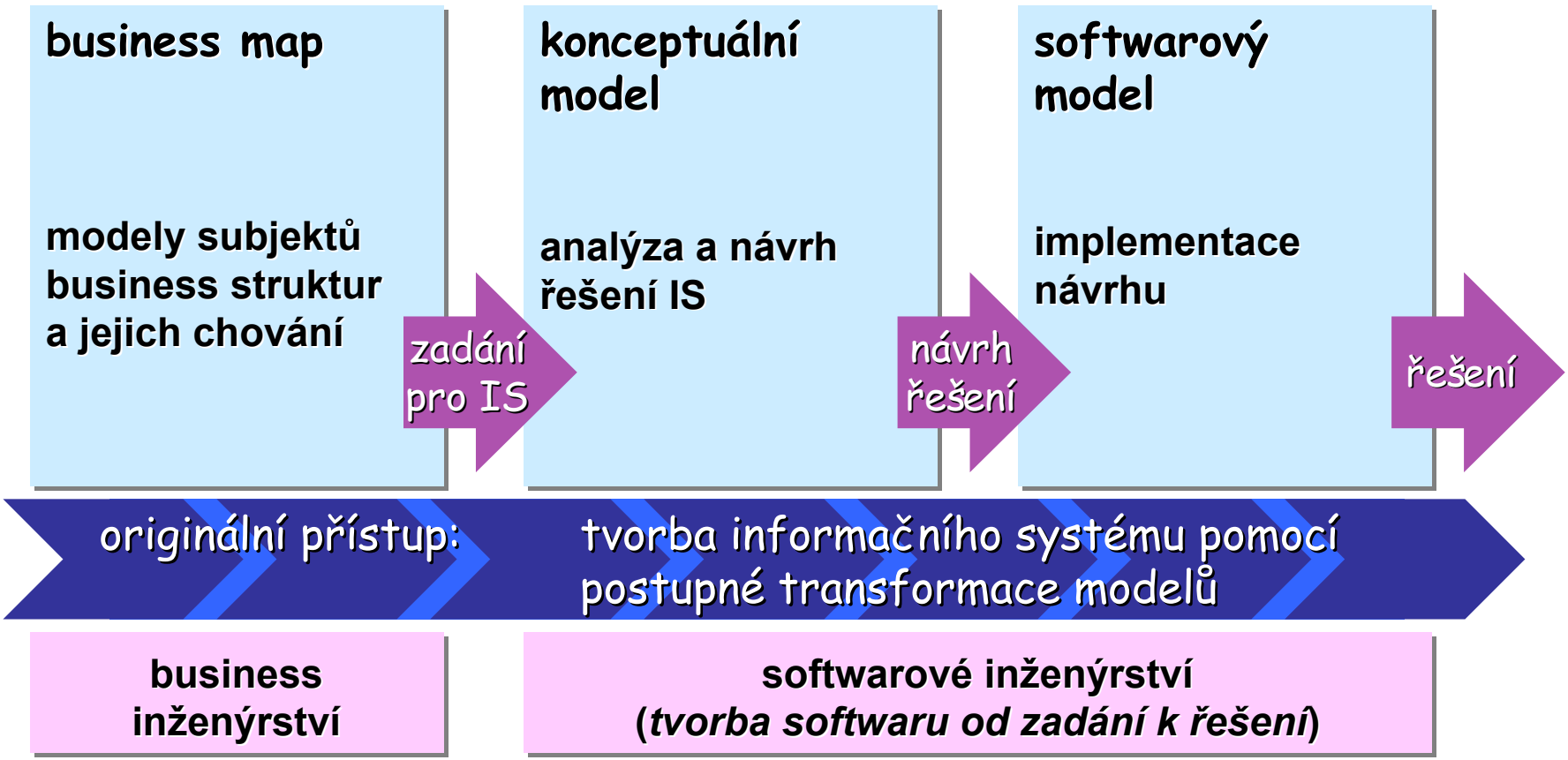
Metoda BORM

BORM – Business and Object Relation Modeling

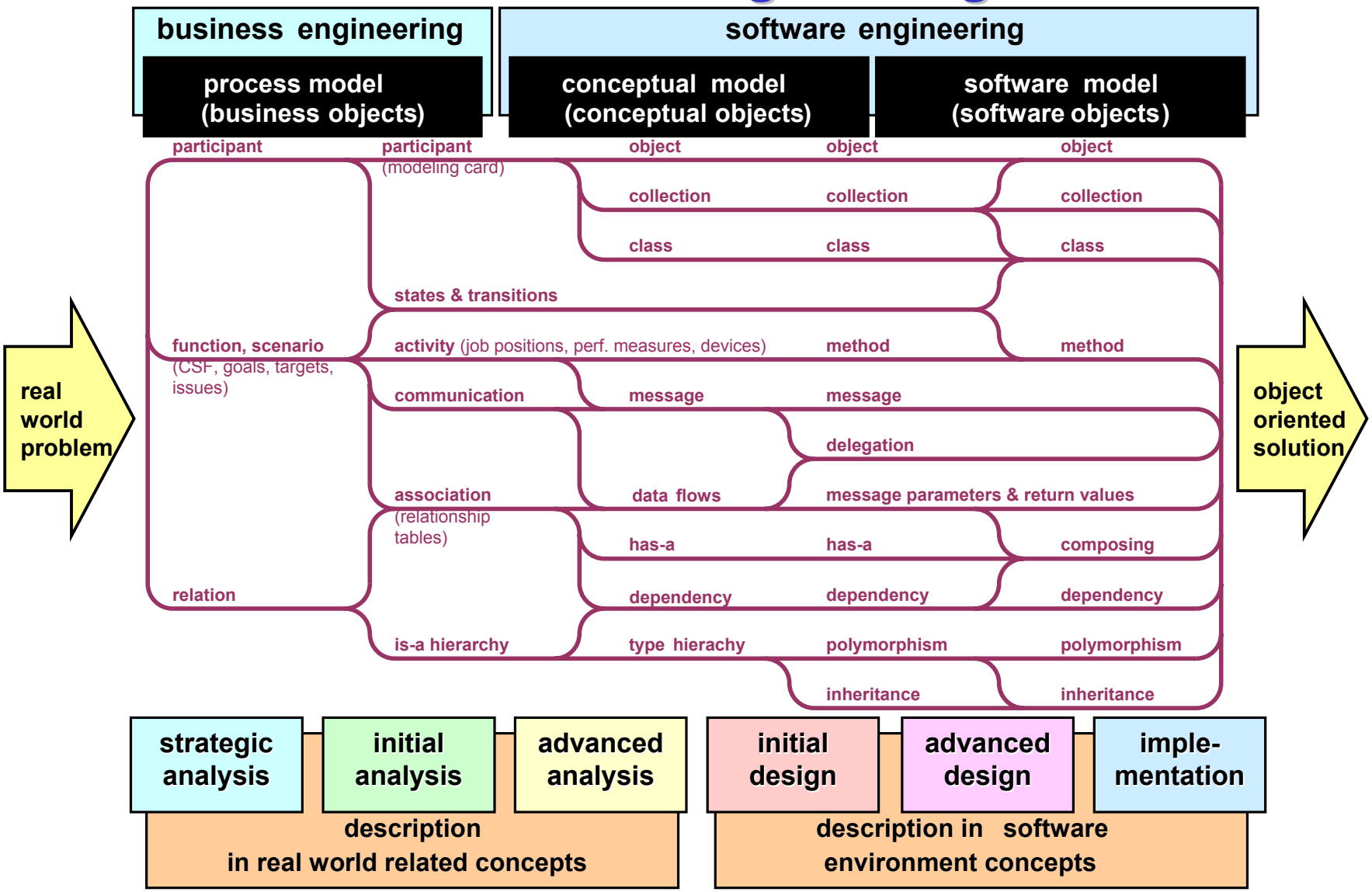
- ✓ Práce na BORMu začaly na počátku 90. let ve výzkumném projektu VAPPIENS Britské rady (Know-How Fund of the British Council).
- ✓ Metoda je od roku 1996 vyvíjena s podporou firmy Deloitte, kde se také používá.
- ✓ Podrobný popis BORMu lze nalézt v knize Carda, Merunka, Polák: Umění systémového návrhu - objektově orientovaná tvorba informačních systémů pomocí původní metody BORM, Grada 2003.

Pro BORM se doposud používal CASE nástroj Metaedit® finské firmy Metacase Ltd nebo MS Visio. Craft.CASE se používá od roku 2005.

BORM - kontext celé metody

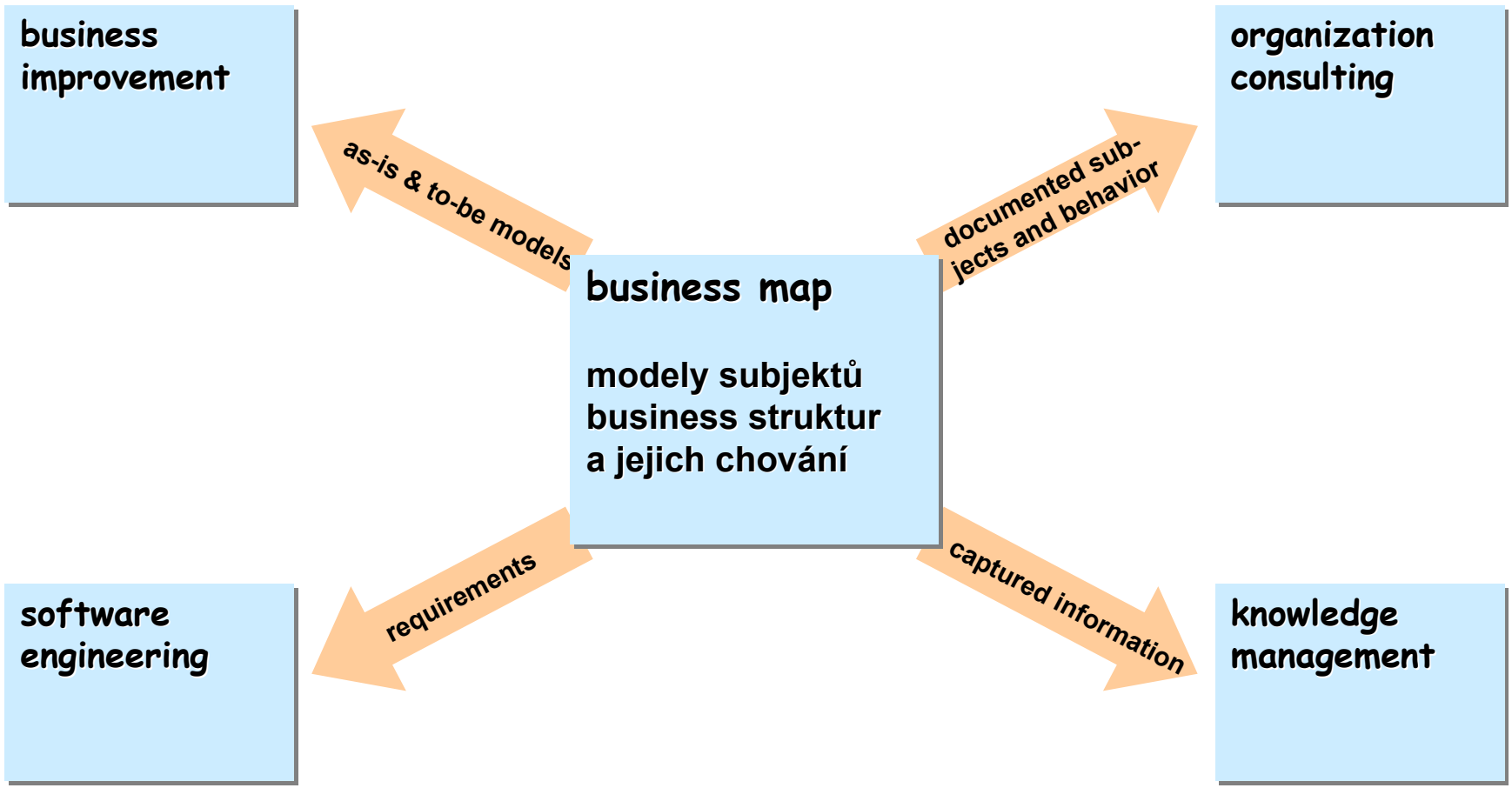


BORM Information Engineering Process

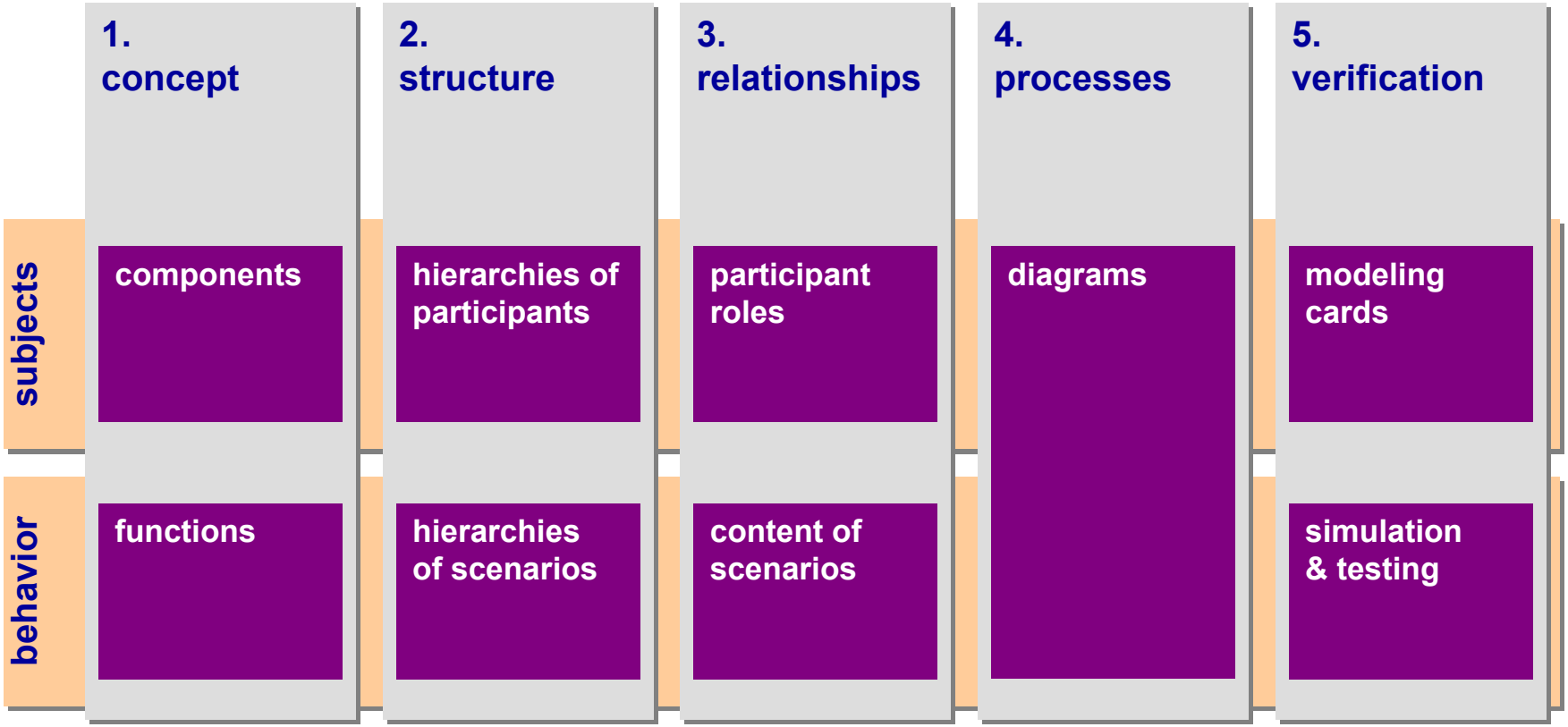


Business Map

„business analysis and design method based on combination of object-oriented approach and process modeling“

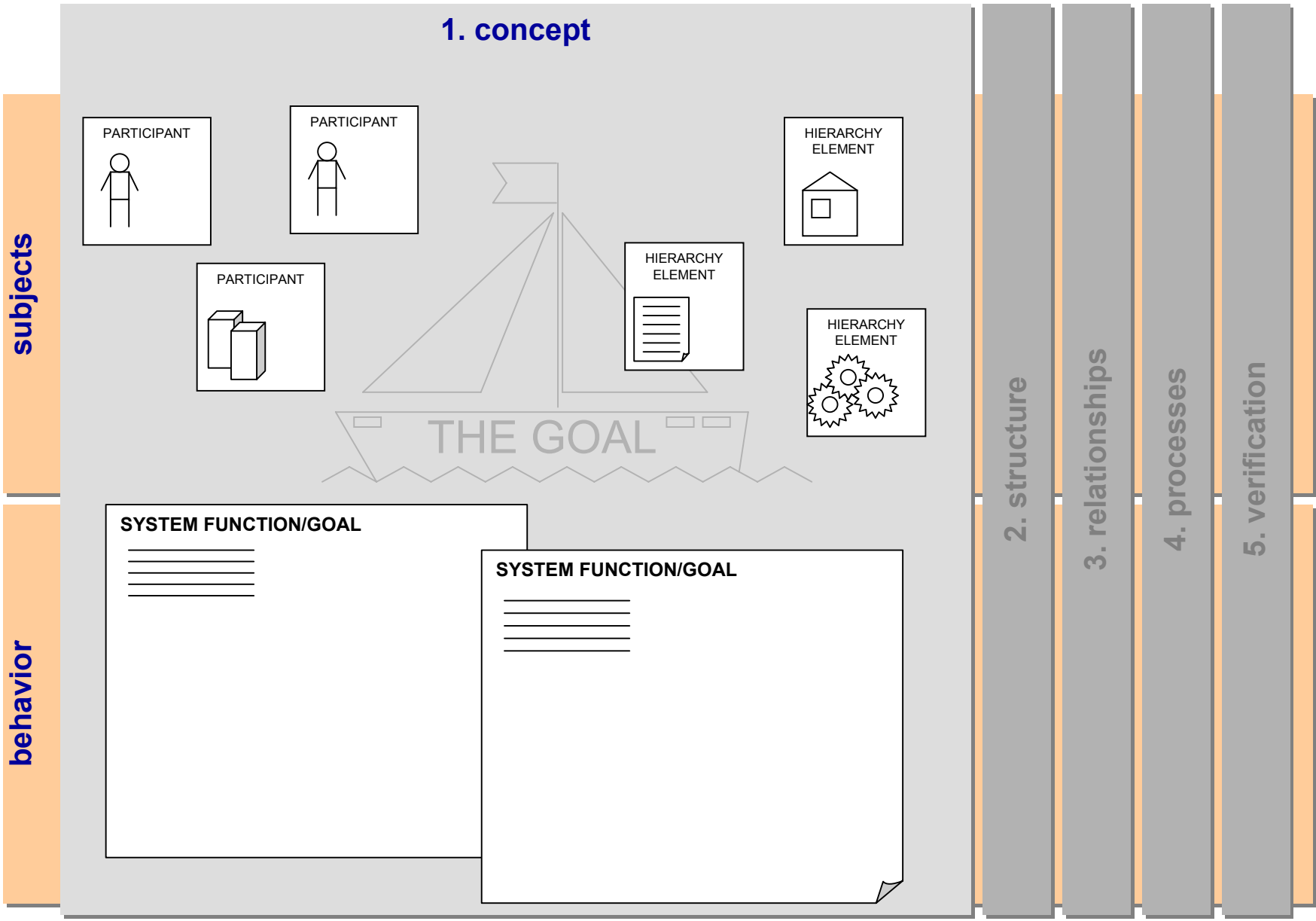


BORM - System Map - framework

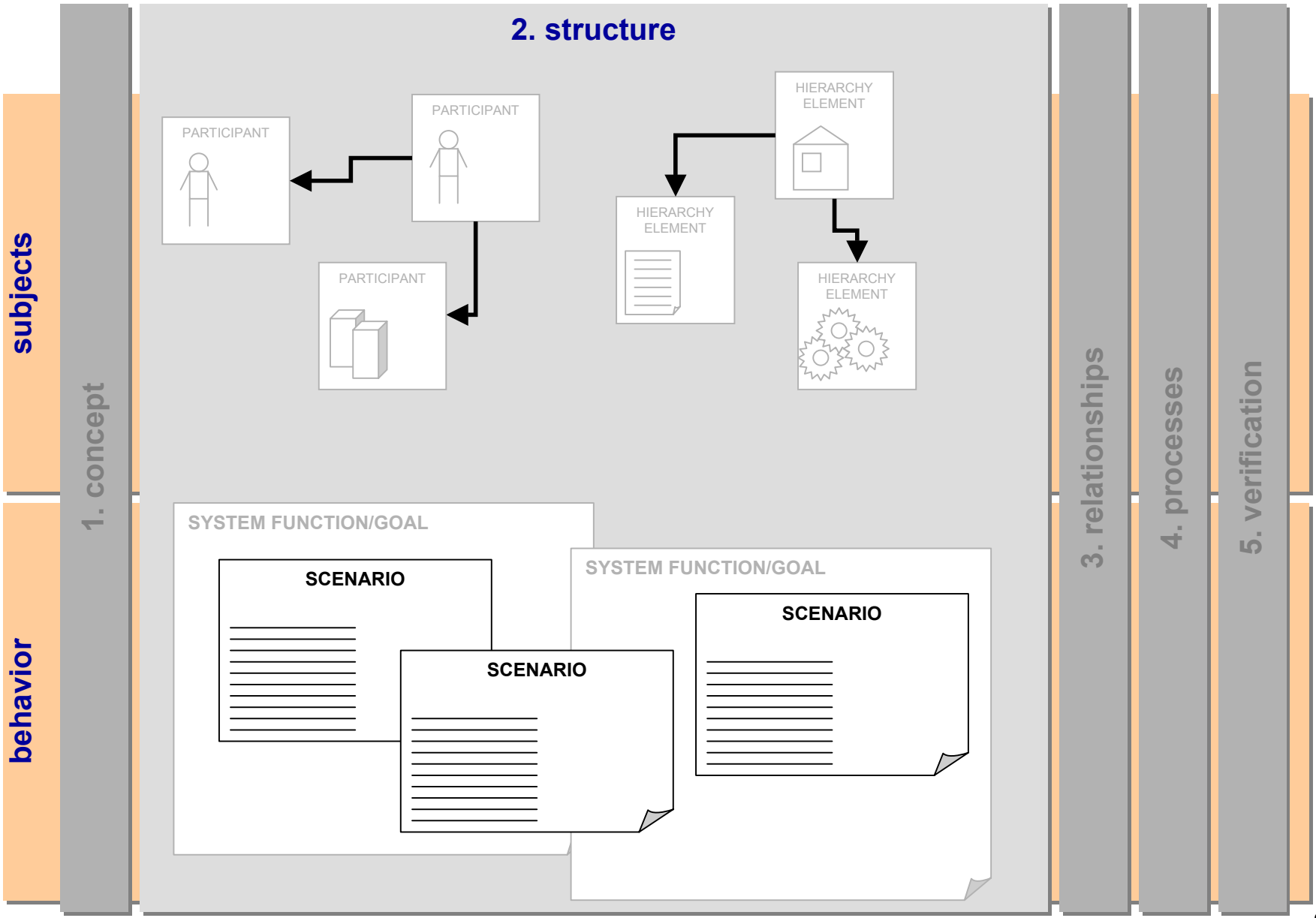


explanation: phase | thread (independent layer) | method

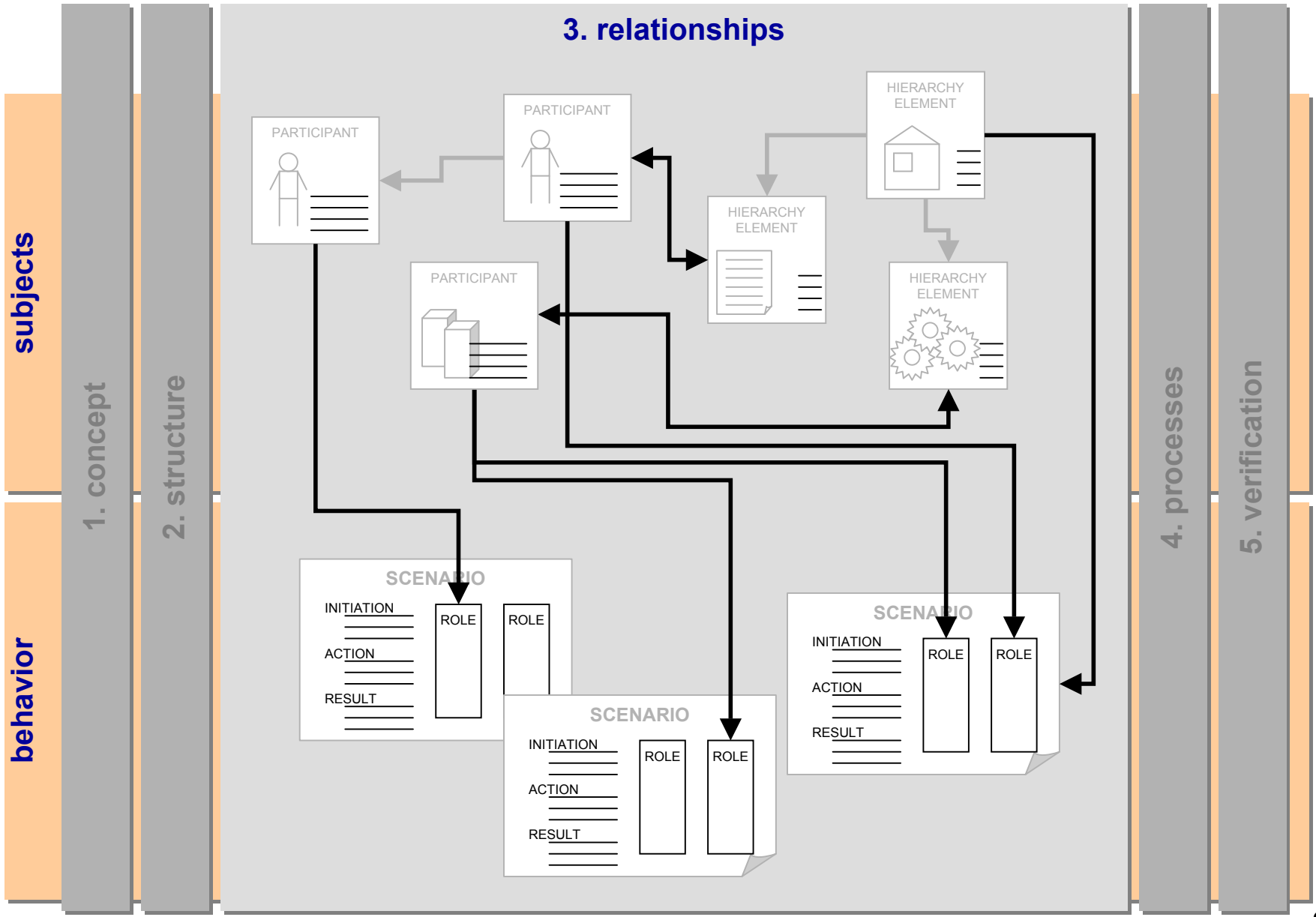
Business Map - What to do/not to do: Concept



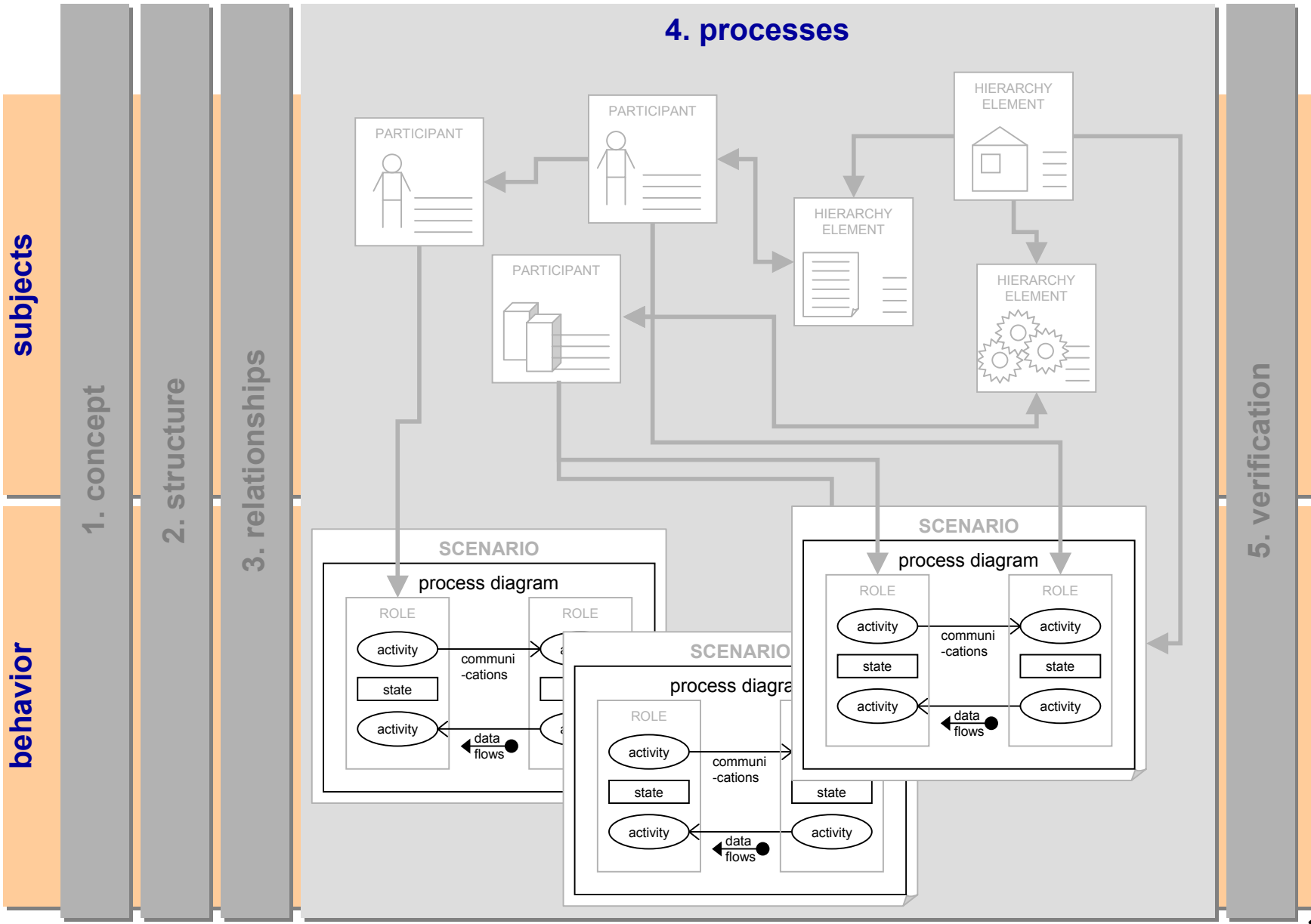
Business Map - How to structure: Structure



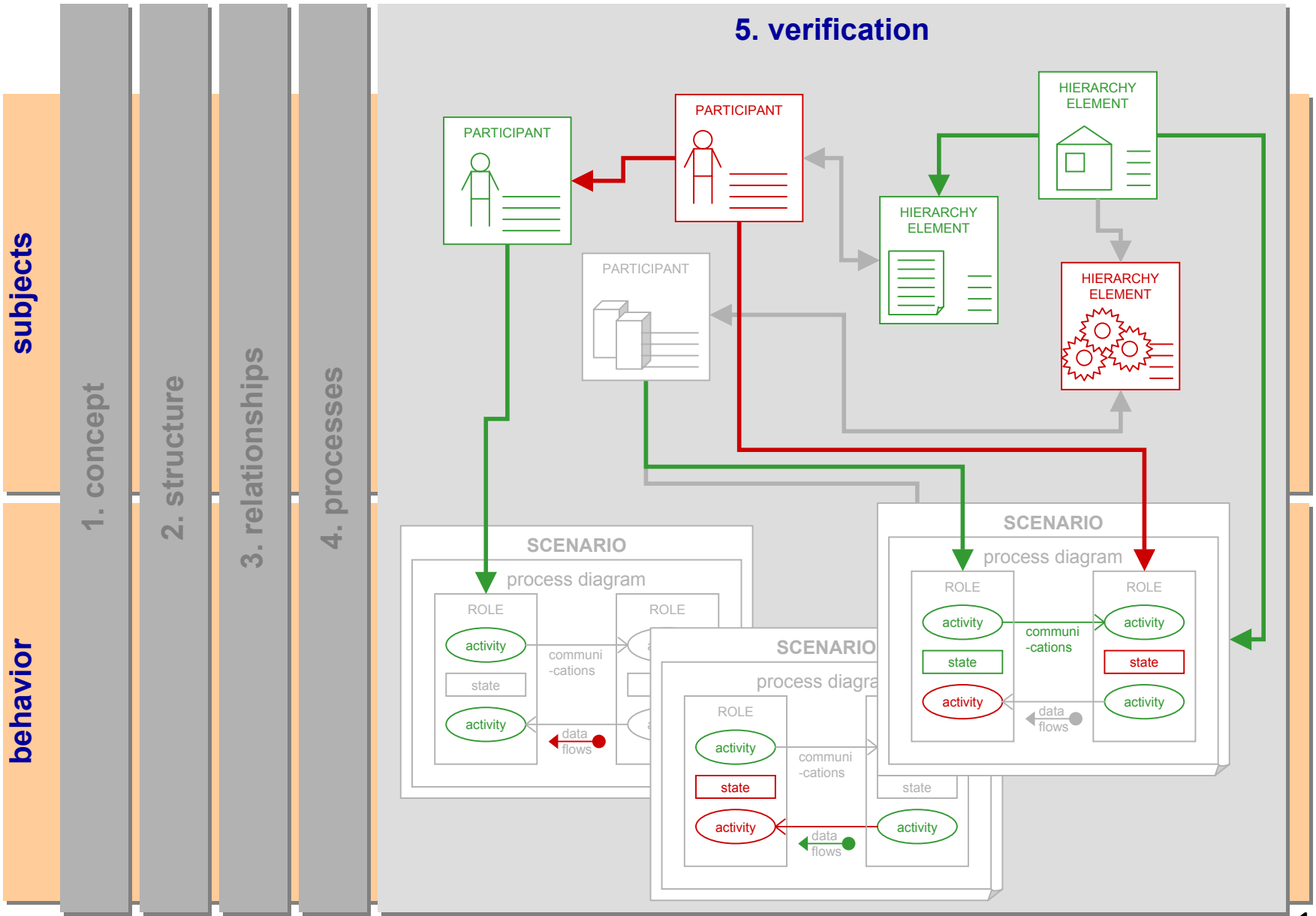
Business Map - Describe detail: Relationships



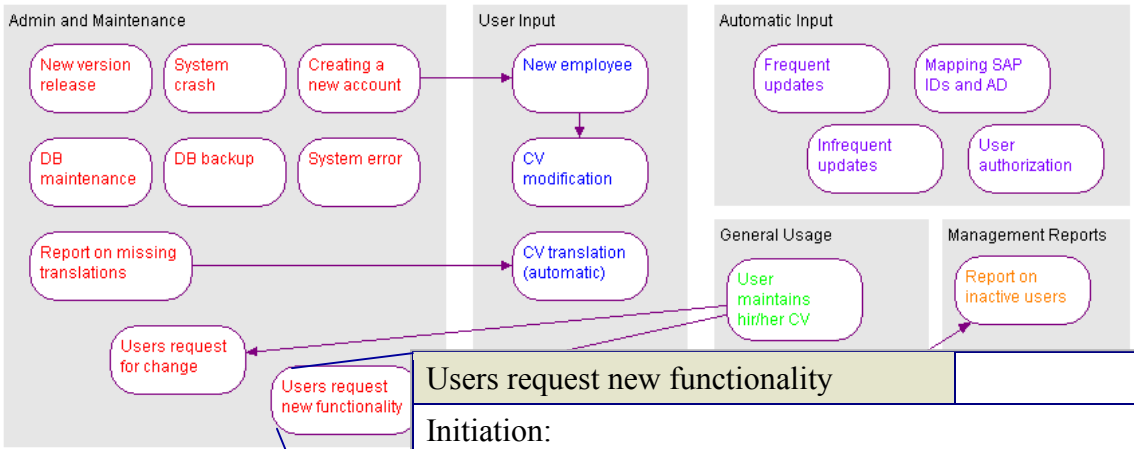
Business Map - visualize the model: Processes



Business Map - simulate and test the model: Verification



Funkce, Scénáře, Participanti a Modelové karty



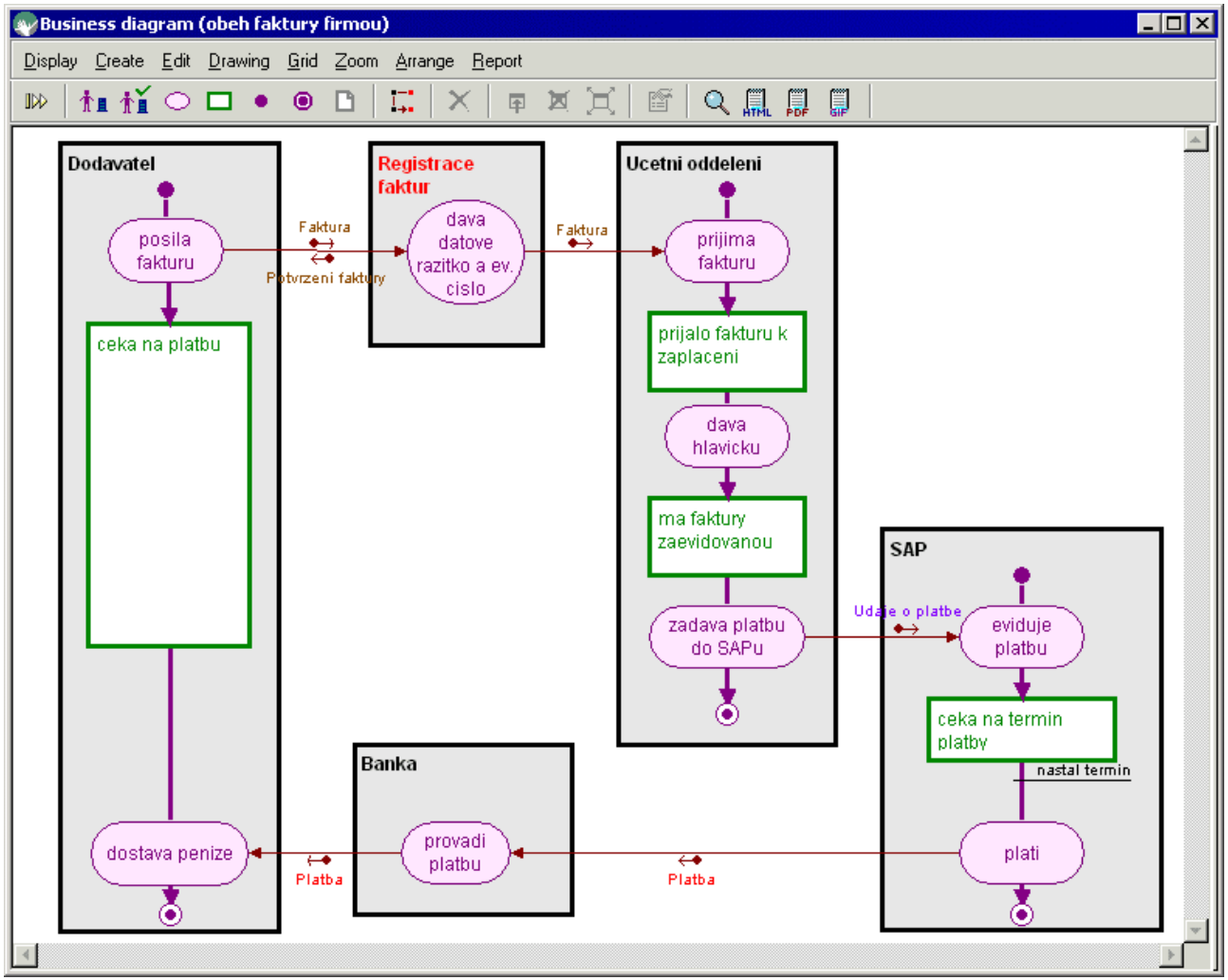
Users request new functionality	Derived from: Admin and Maintenance
Initiation: User requests new functionality.	
Action: IT Support evaluates the requests and accumulates relevant information for future development.	
Result: Developer periodically receives requests for upgrades (accumulated, not one-by-one).	
Developer	
IT Support	

Developer	**CVDB**	**IT Support**	**User**
New version release	x	x	
System error	x	x	
Users request for change		x	x
Users request new functionality		x	x

Procesní diagram

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

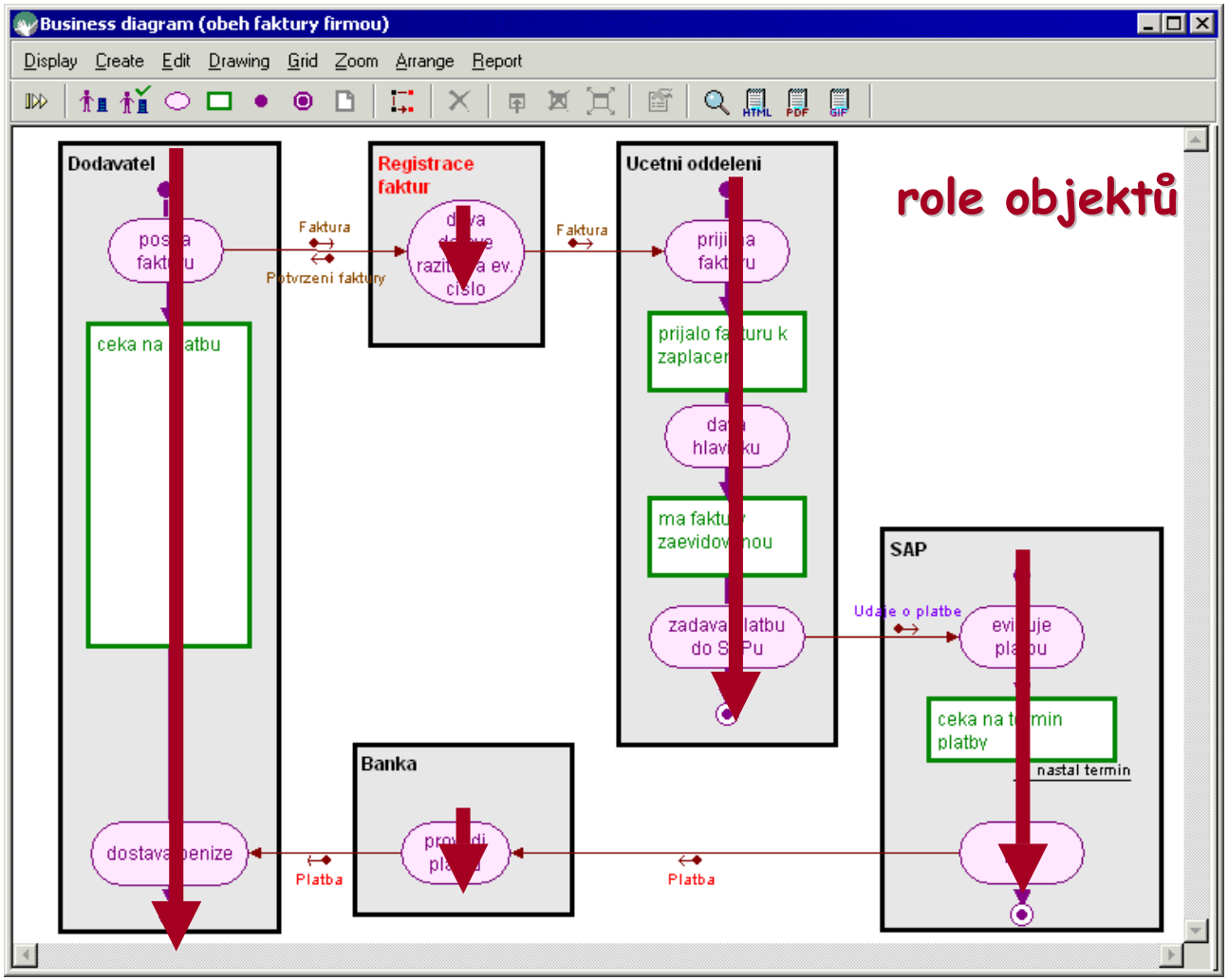
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Procesní diagram

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

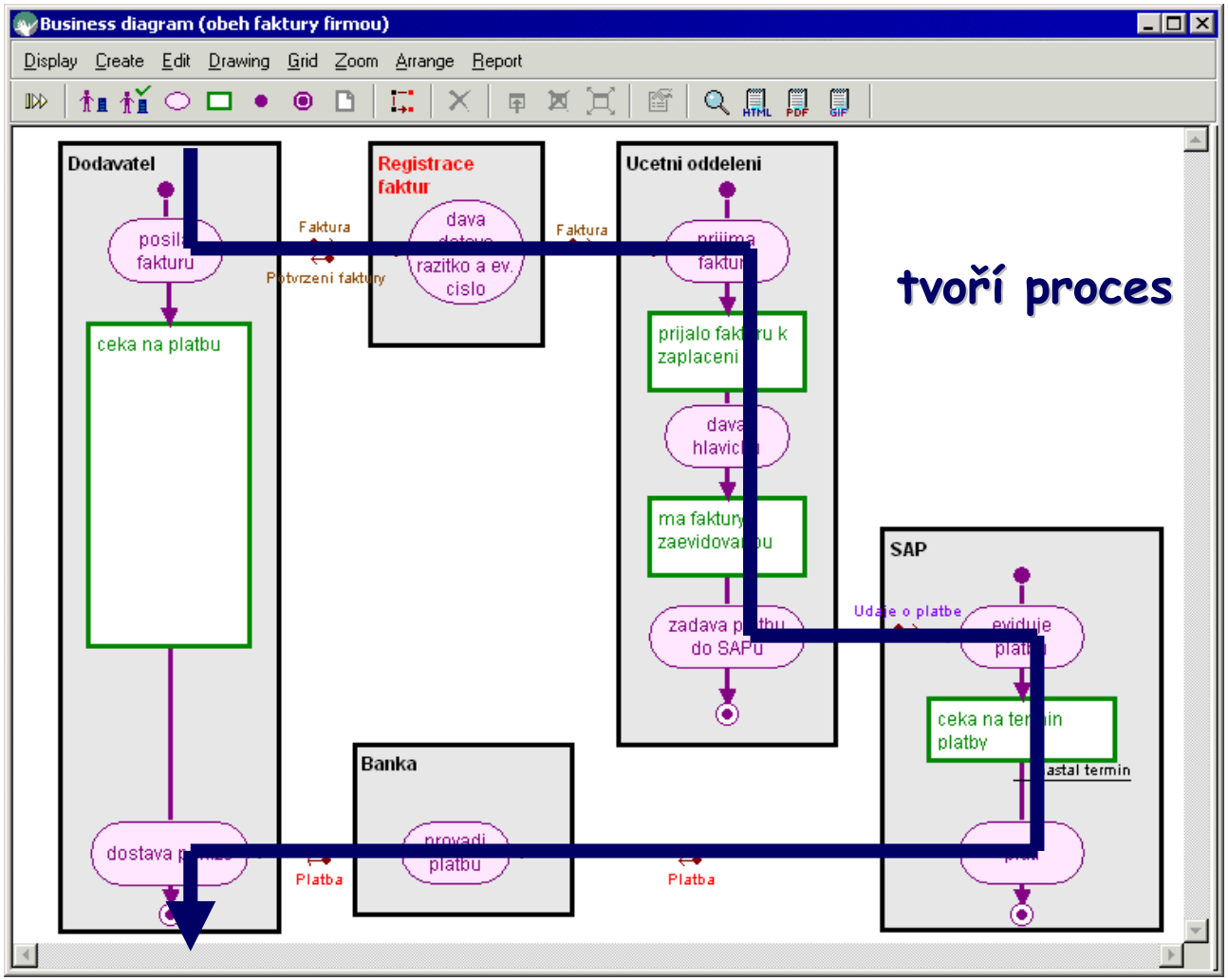
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Procesní diagram

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

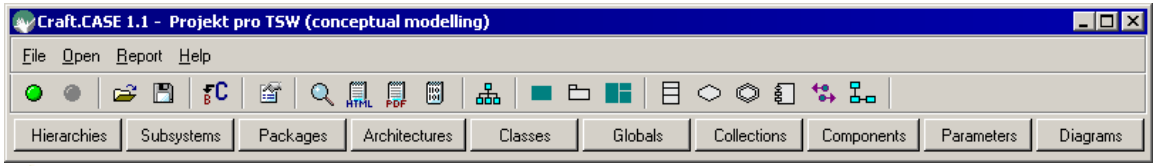
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Antonín Carda,
Vojtěch Merunka,
Vladimír Vlachovský

http://www.craftcase.com

Projekt Craft.CASE



Craft.CASE je původní český modelovací a analytický CASE nástroj podporující metodu BORM®, která je založena na kombinaci objektově orientovaného přístupu a procesního modelování.

Nástroj vzniká ve firmě e-Fractal s.r.o.

Zadání vychází ze dvou potřeb:

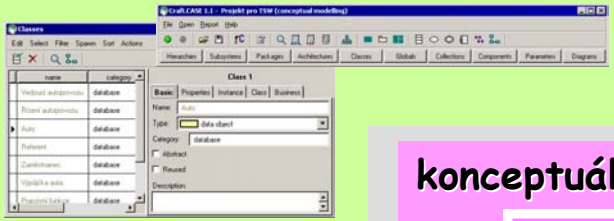
1) Jednoduše ovladatelný a na prostředky počítače nenáročný.
2) Modelovací nástroj přesně šitý na míru metodě BORM, který je částečně konfigurovatelný, dokáže procesy simulovat a generuje výstupní dokumentaci.

Program je vyvíjen v prostředí VisualWorks/Smalltalk a je určen pro použití ve Windows 2000 a XP.

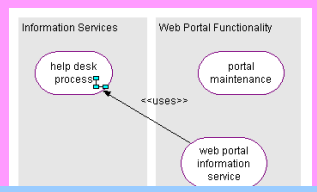
http://www.craftcase.com

Shrnutí - projektování pomocí Craft.CASE

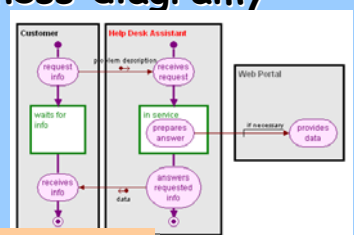
společná databáze



business model



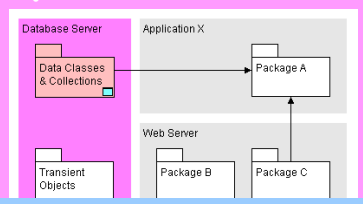
business diagramy



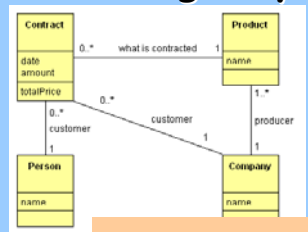
simulátor

transformace

konceptuální model



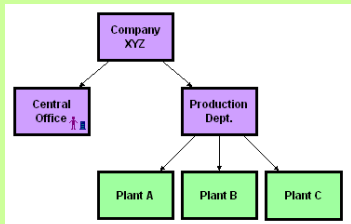
konceptuální diagramy



generátor kódu

vazby

pomocné hierarchie

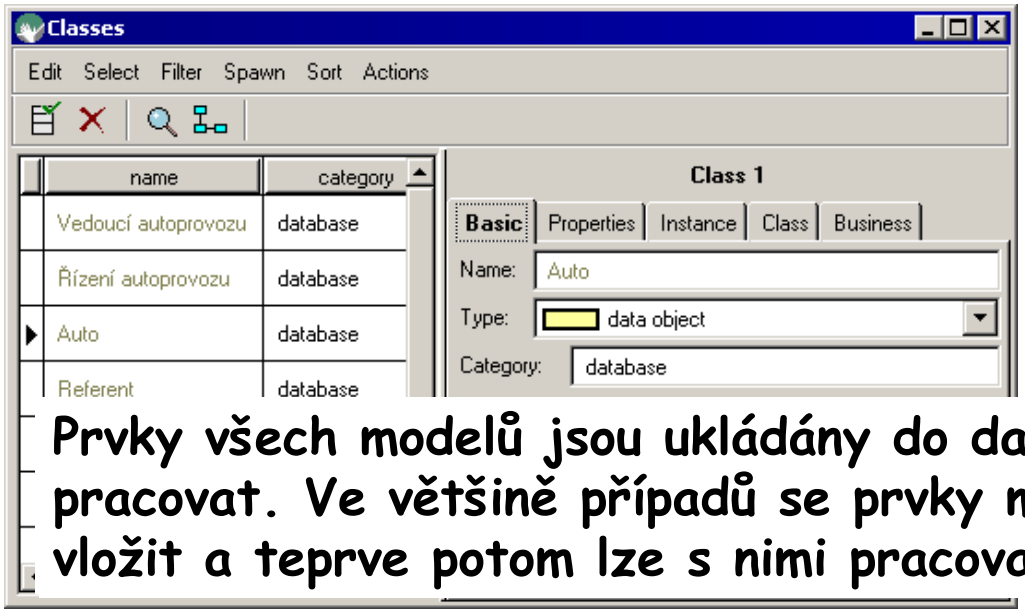


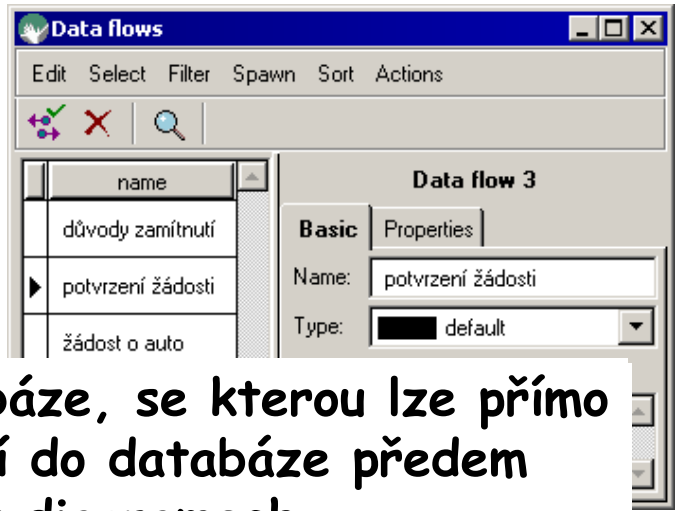
vazby

modelování zadání pro IS a jeho prostředí

analýza a návrh IS

Craft.CASE





Prvky všech modelů jsou ukládány do databáze, se kterou lze přímo pracovat. Ve většině případů se prvky musí do databáze předem vložit a teprve potom lze s nimi pracovat v diagramech.

Výstupní dokumentace je tvořena hypertexty ve formátu HTML a také PDF a obsahuje:

1. seznamy prvků z databáze
2. diagramy
3. simulační záznamy
4. modelové karty
   (= tabulky s křížovými referencemi)

5.2.5. SAP

Collaborators in diagram wit firmou':
start: eviduje platbu
ceka na termin platby: plati

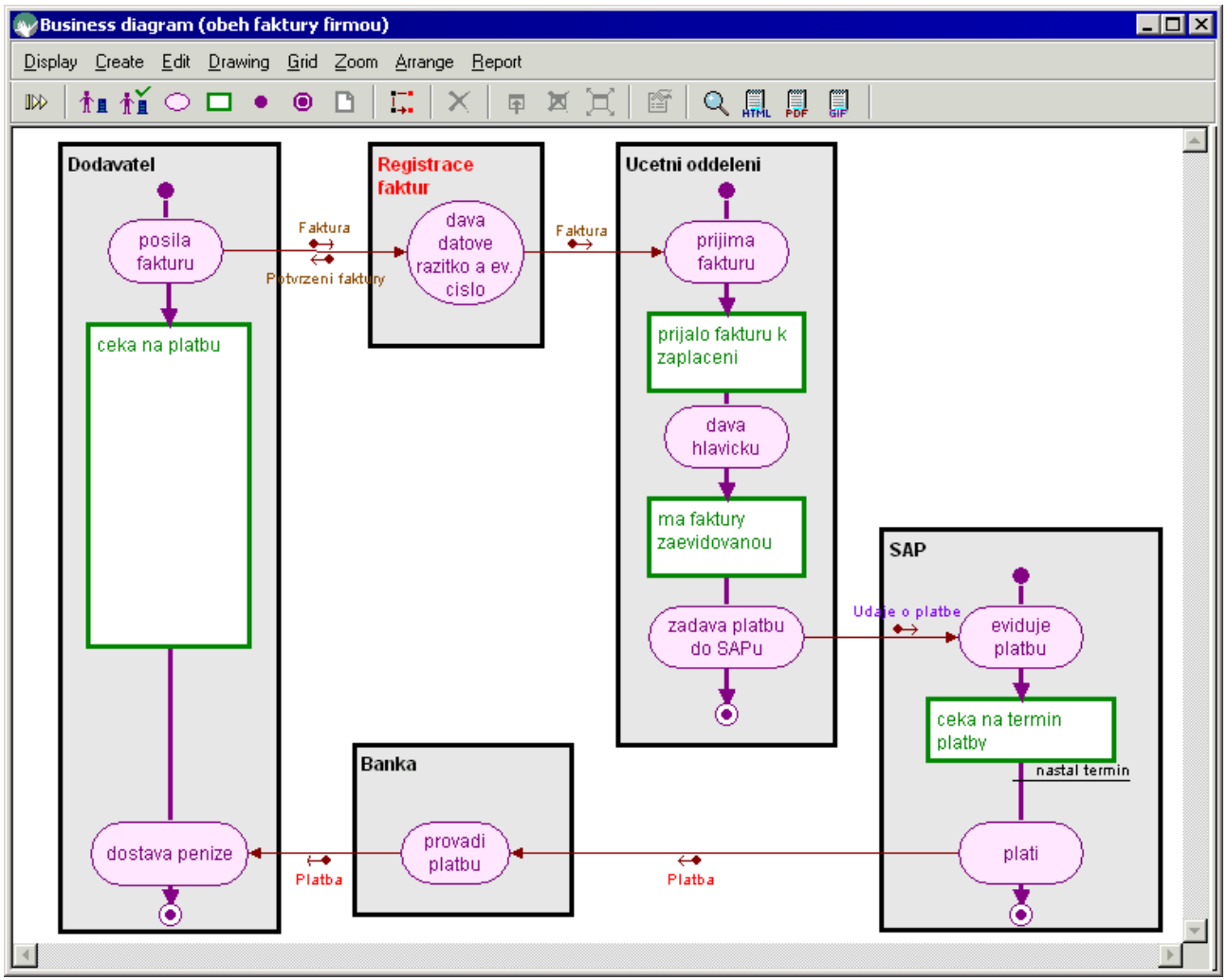
5.2.6. Vedoucí autoprovozu

Collaborators in diagram with id 'výpůjčka auta':	Auto	Referent	Vedoucí
čeká na vrácení auta: přebírá vrácené auto	<<		>>
má žádost o auto: přiděluje auto	>>	>>	
start: přijímá žádost o vydání auta		<<	
má žádost o auto: ruší výběr auta			<<
má žádost o auto: vybírá auto			

Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

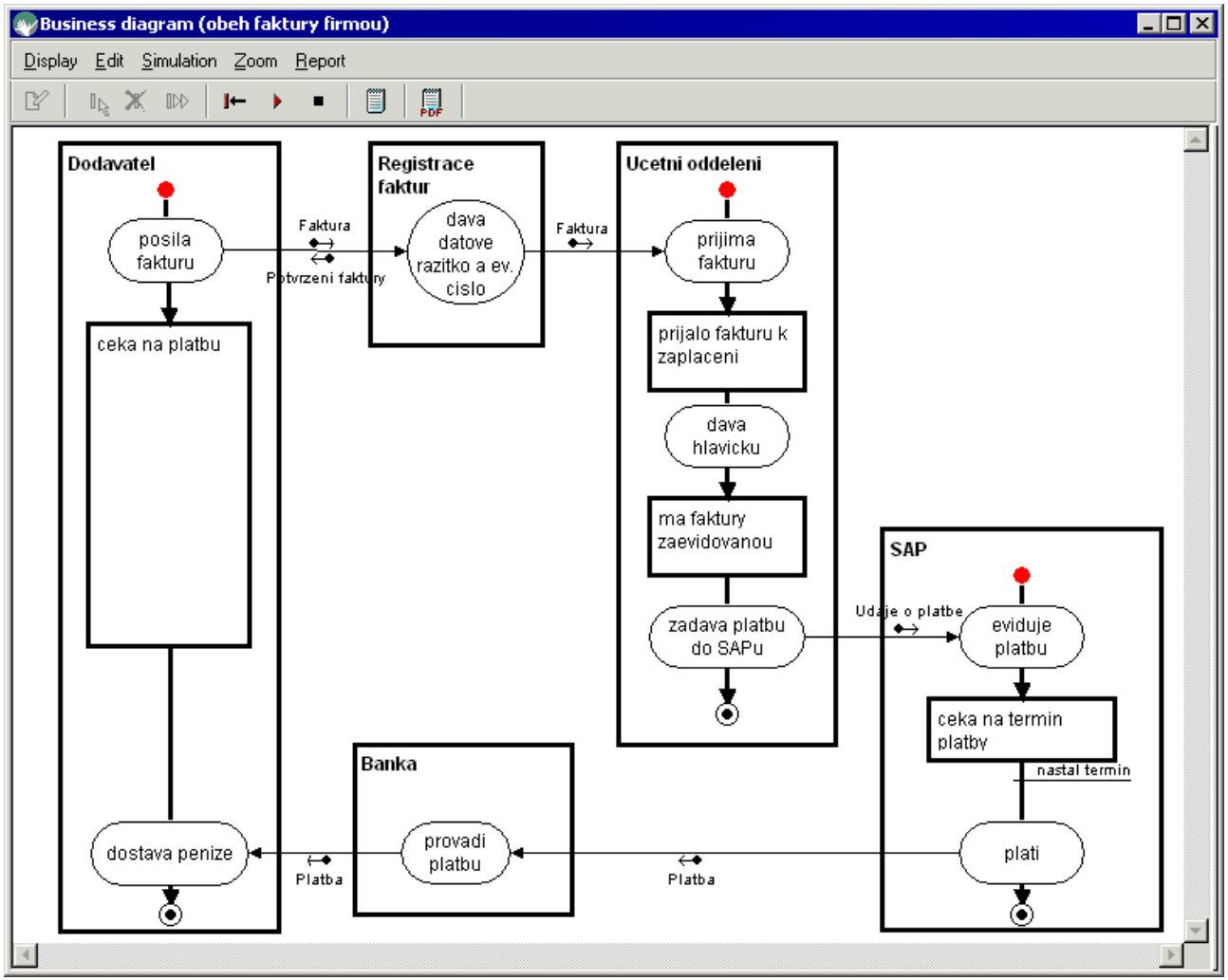
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

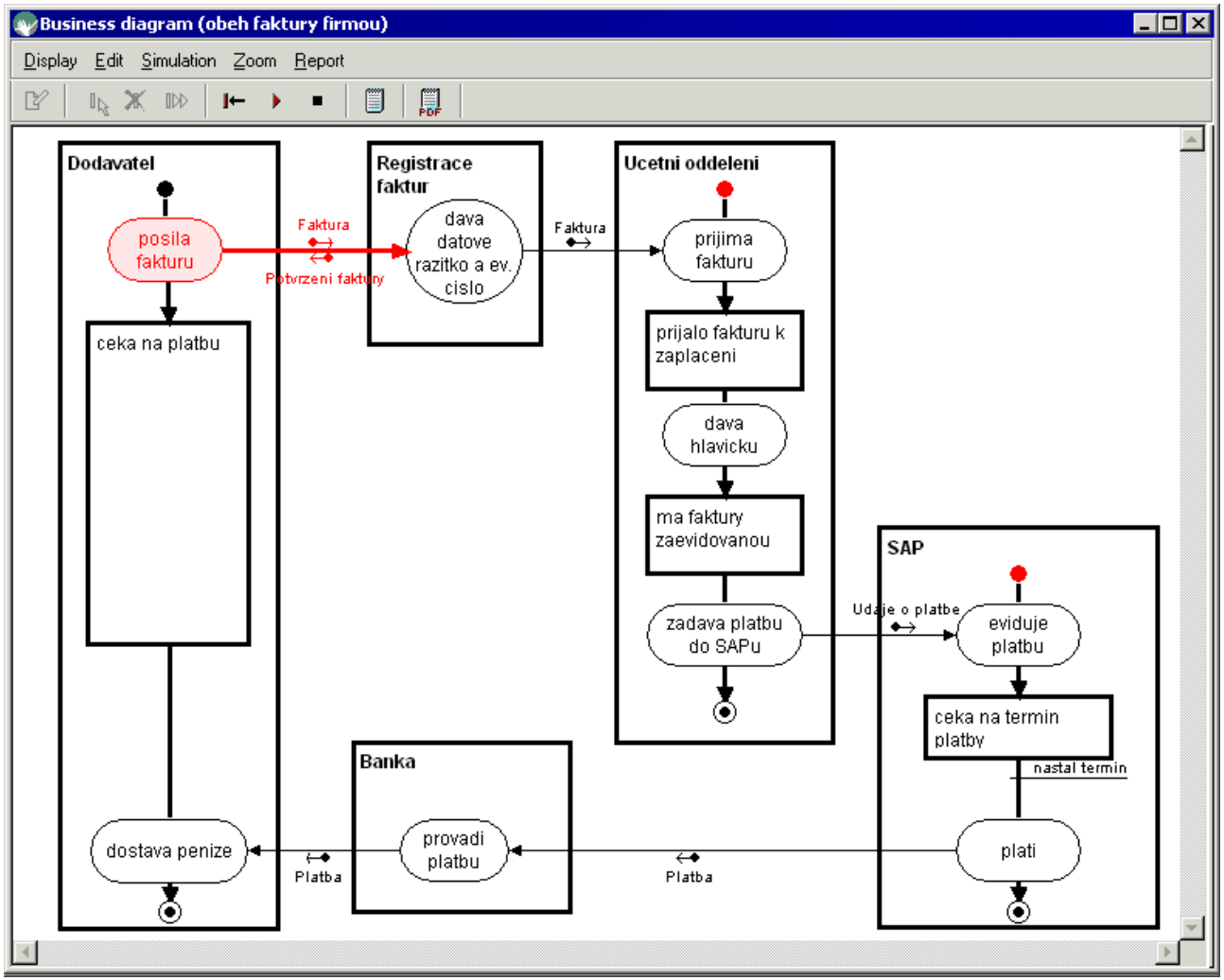
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

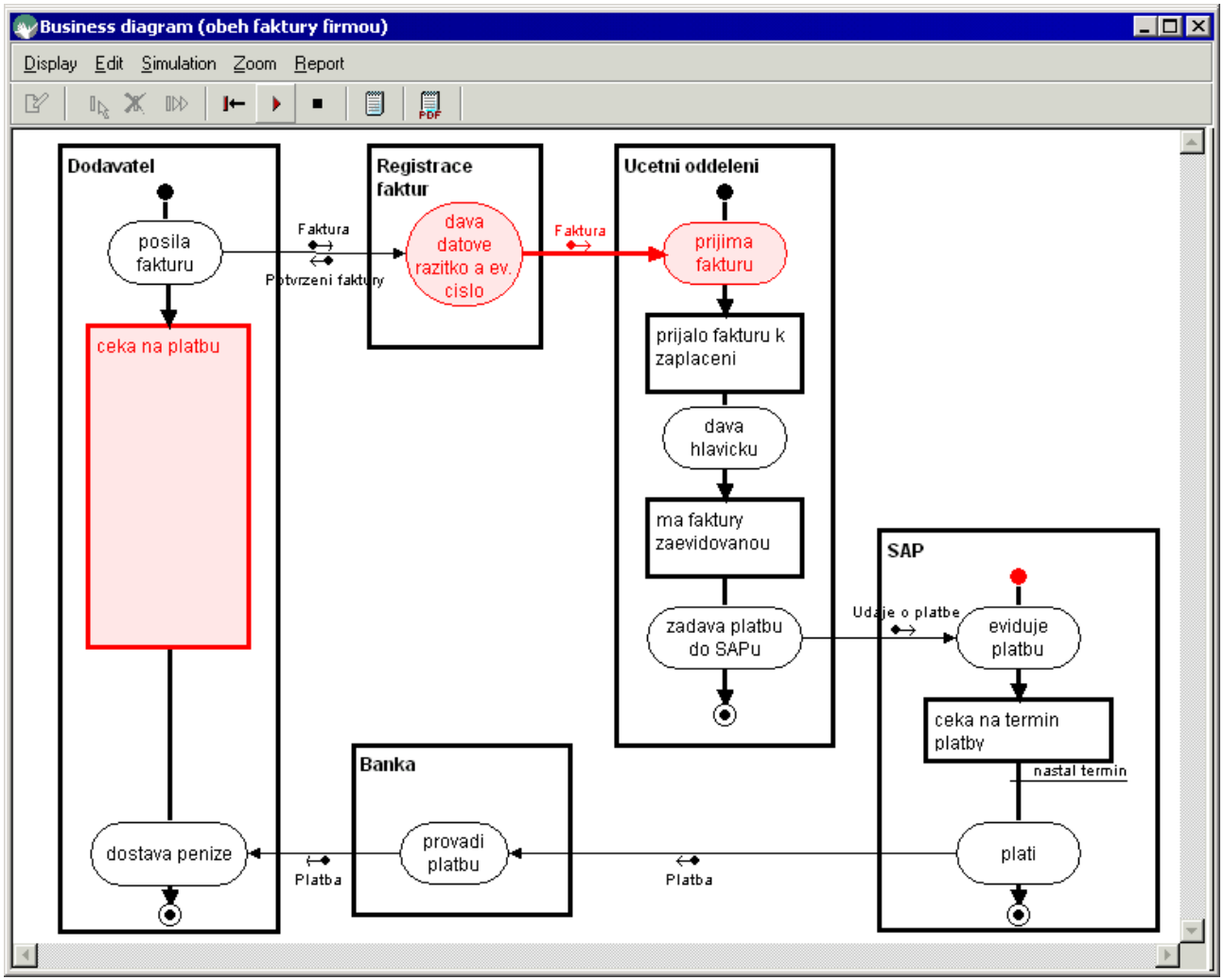
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

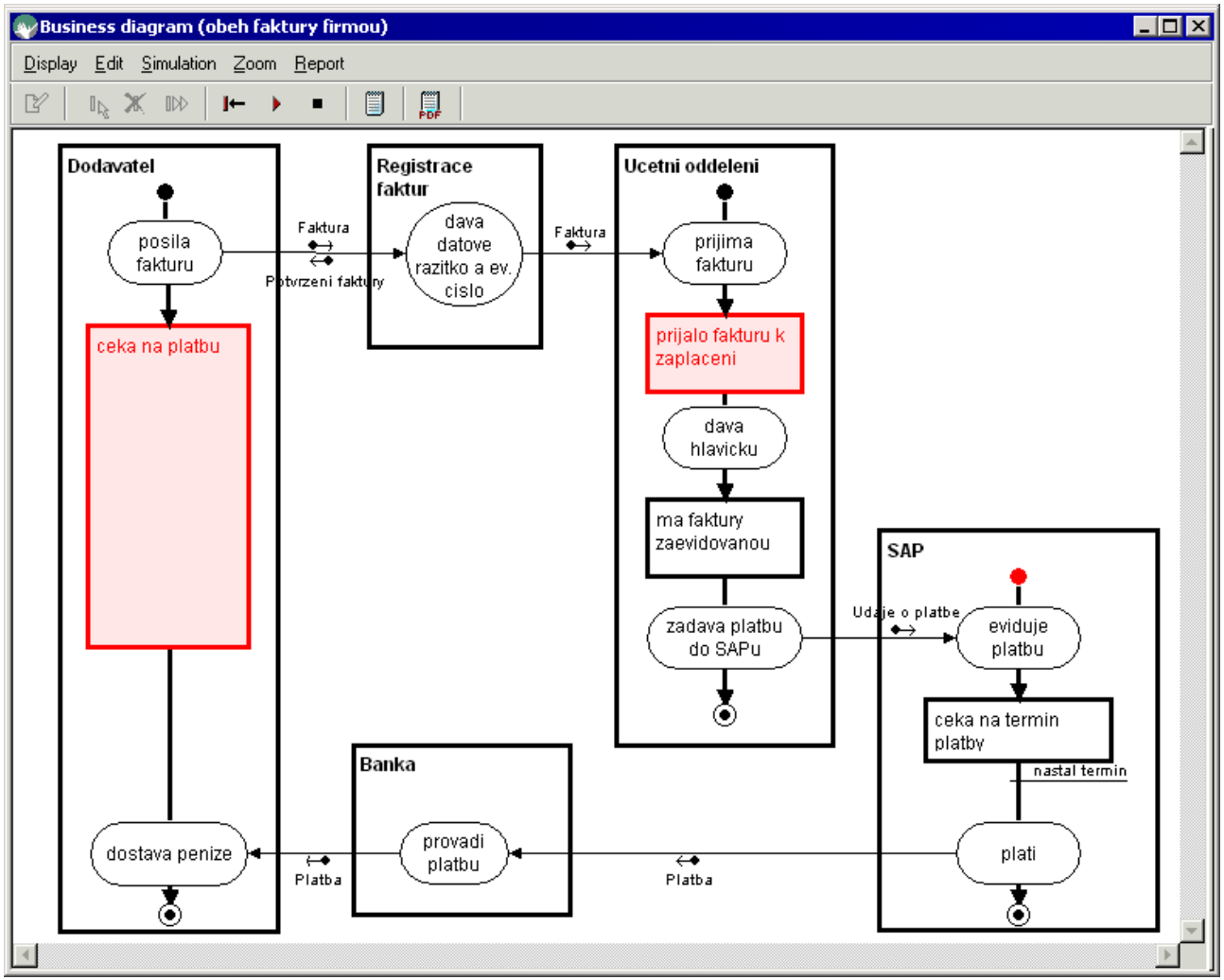
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

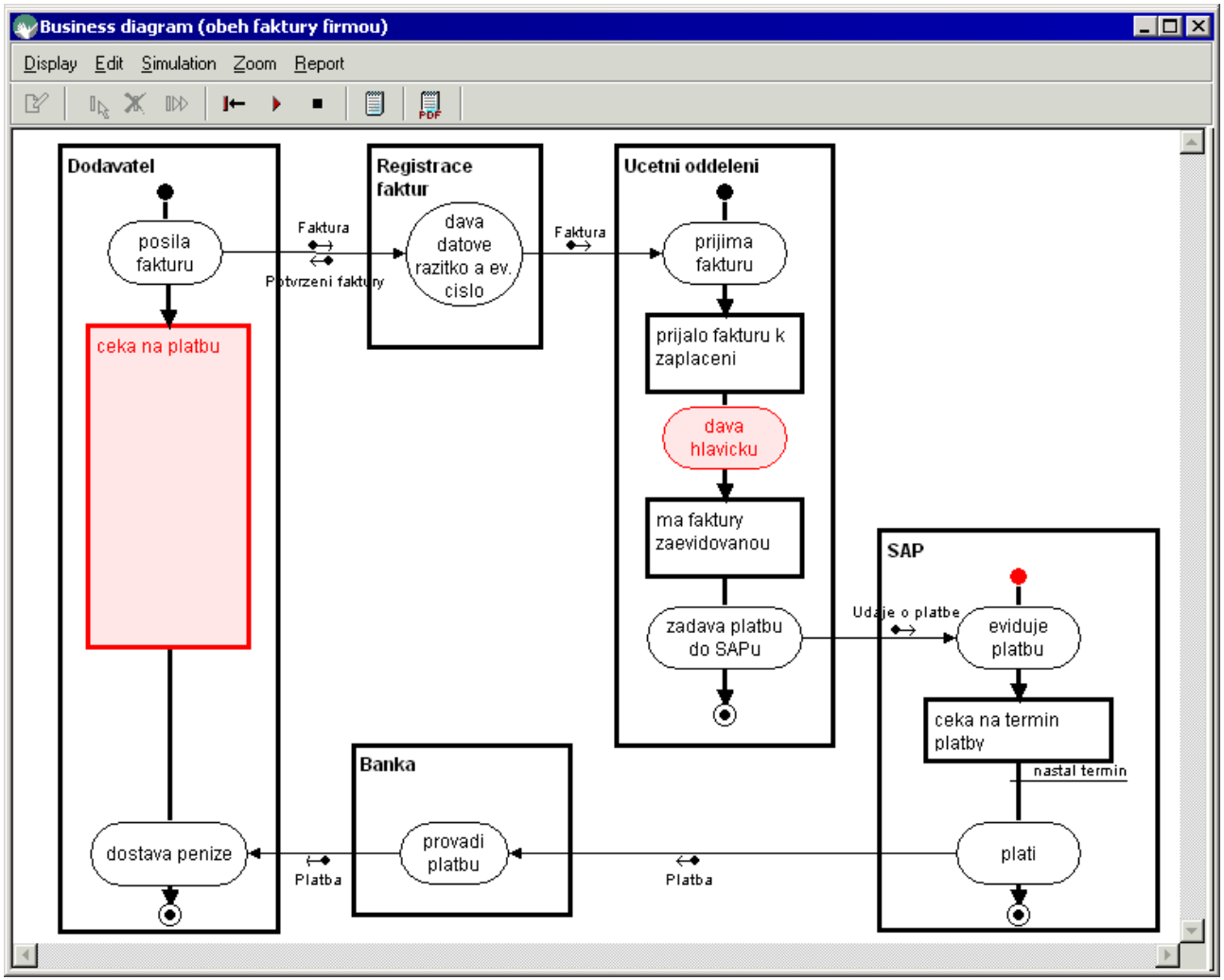
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

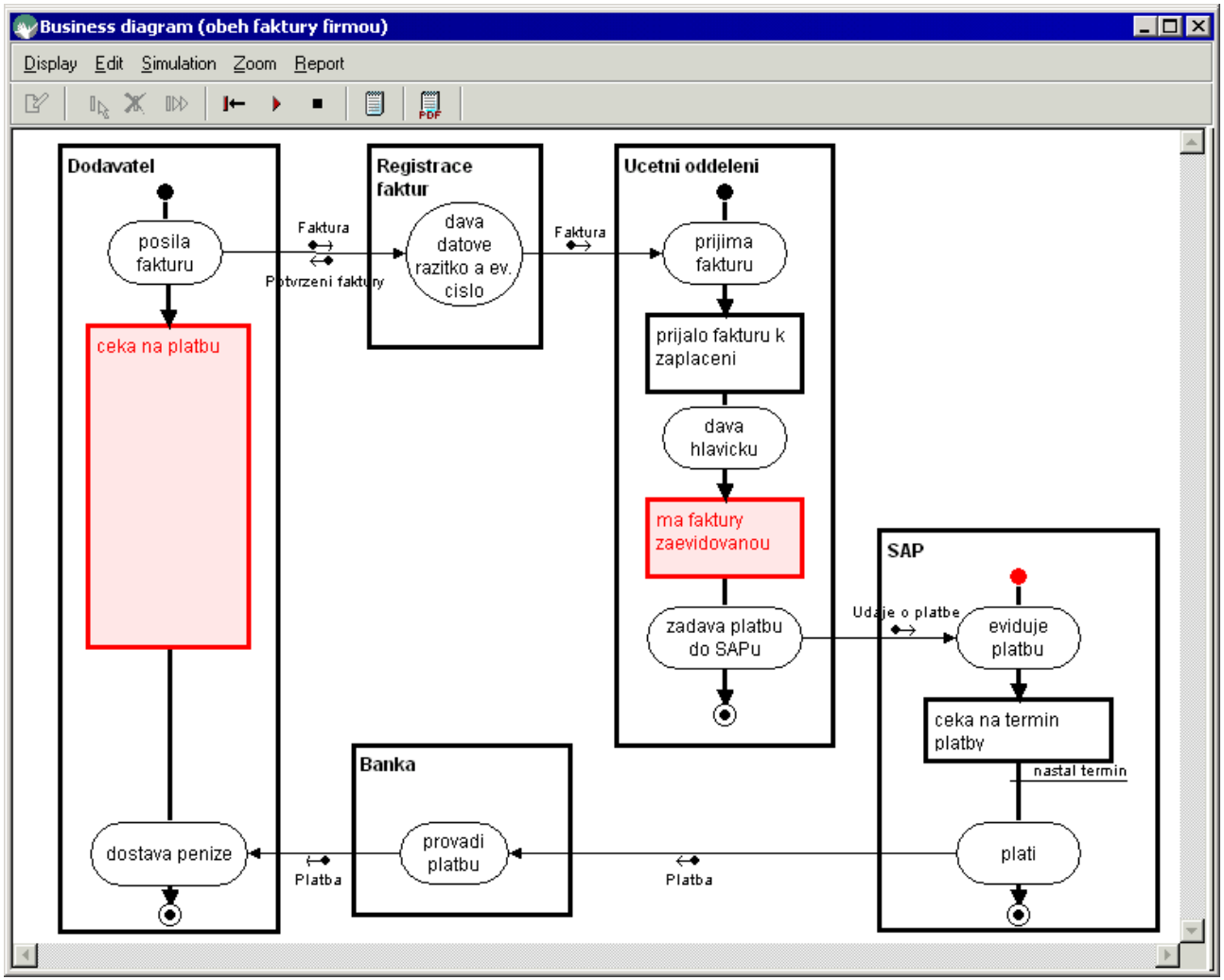
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

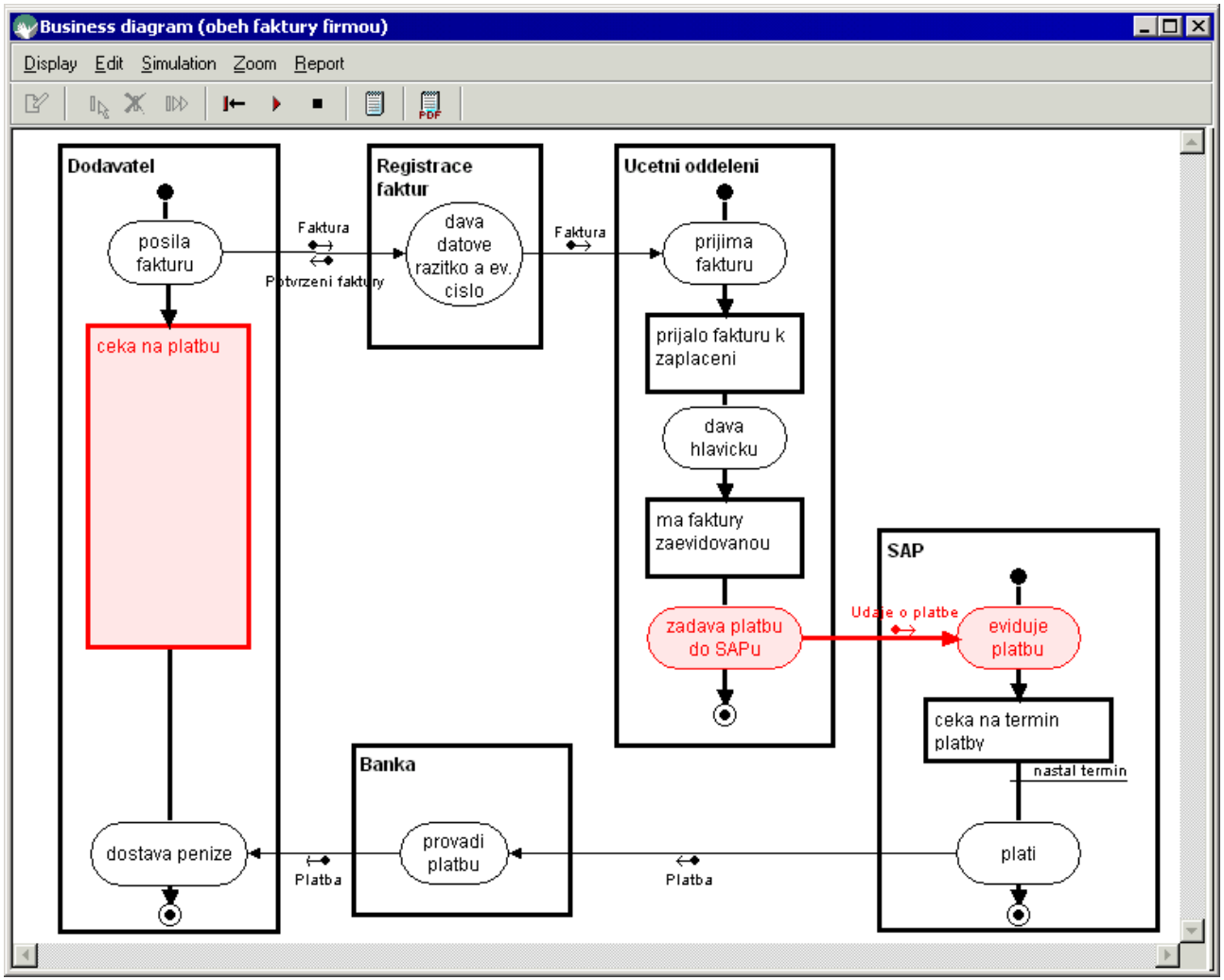
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

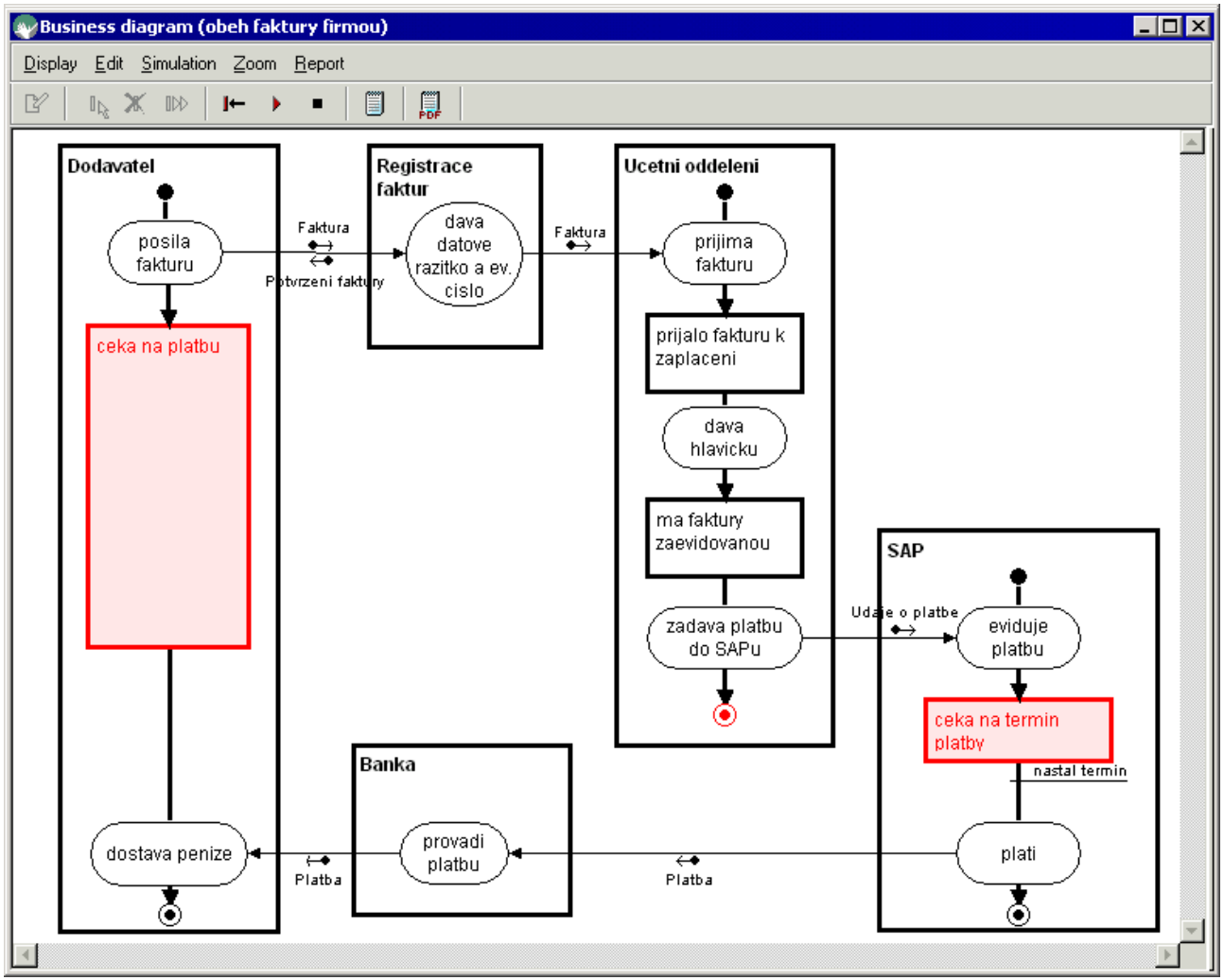
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

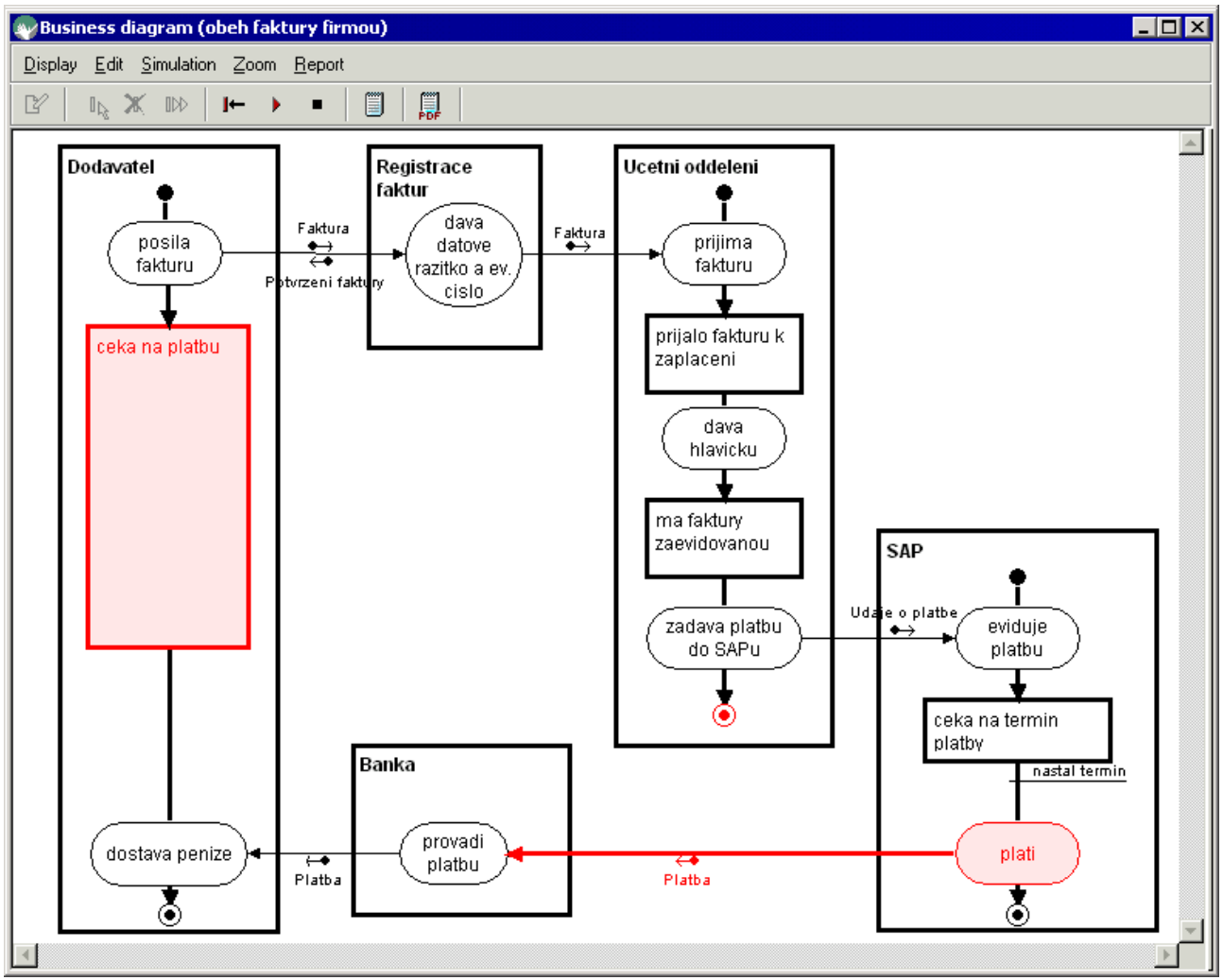
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

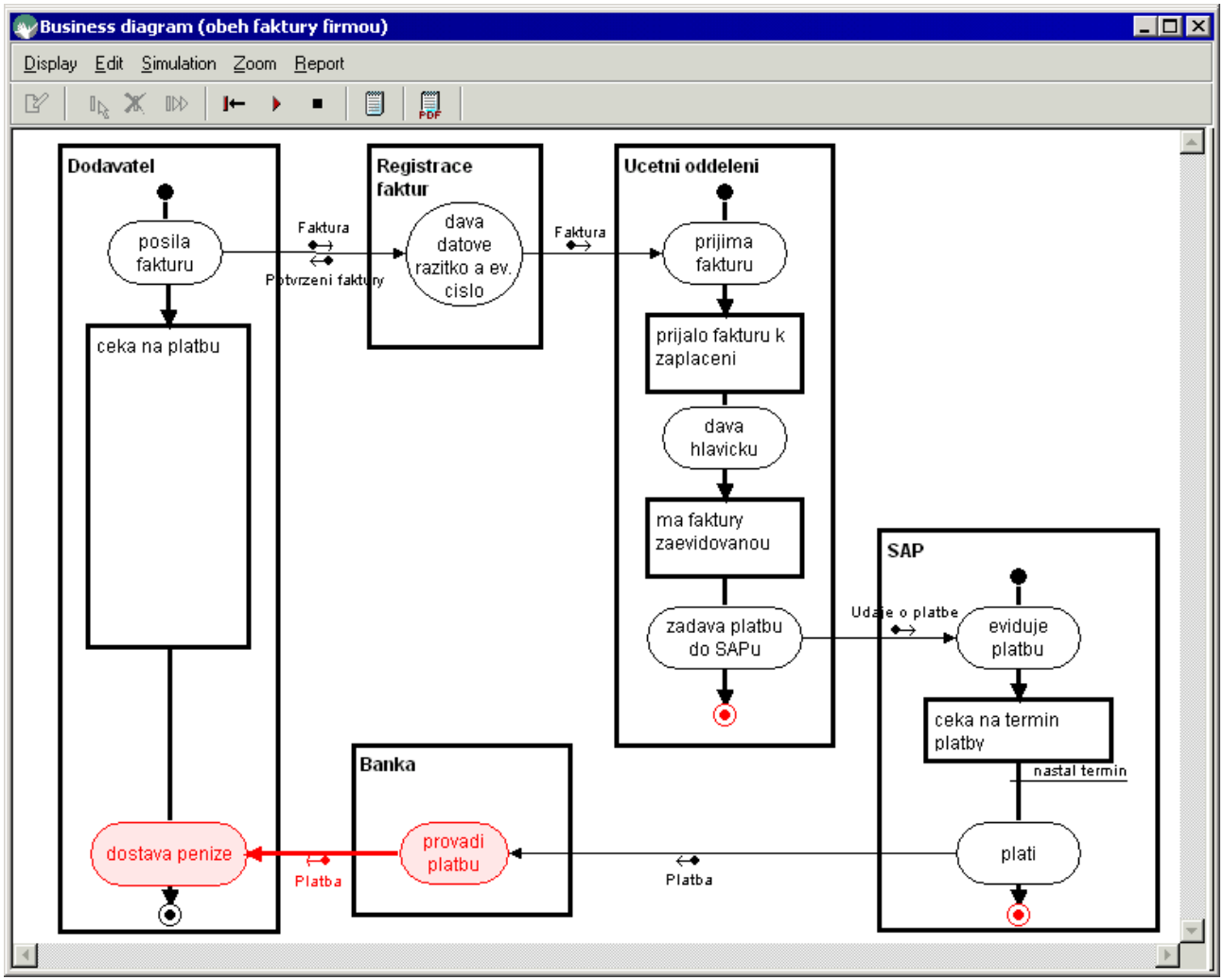
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

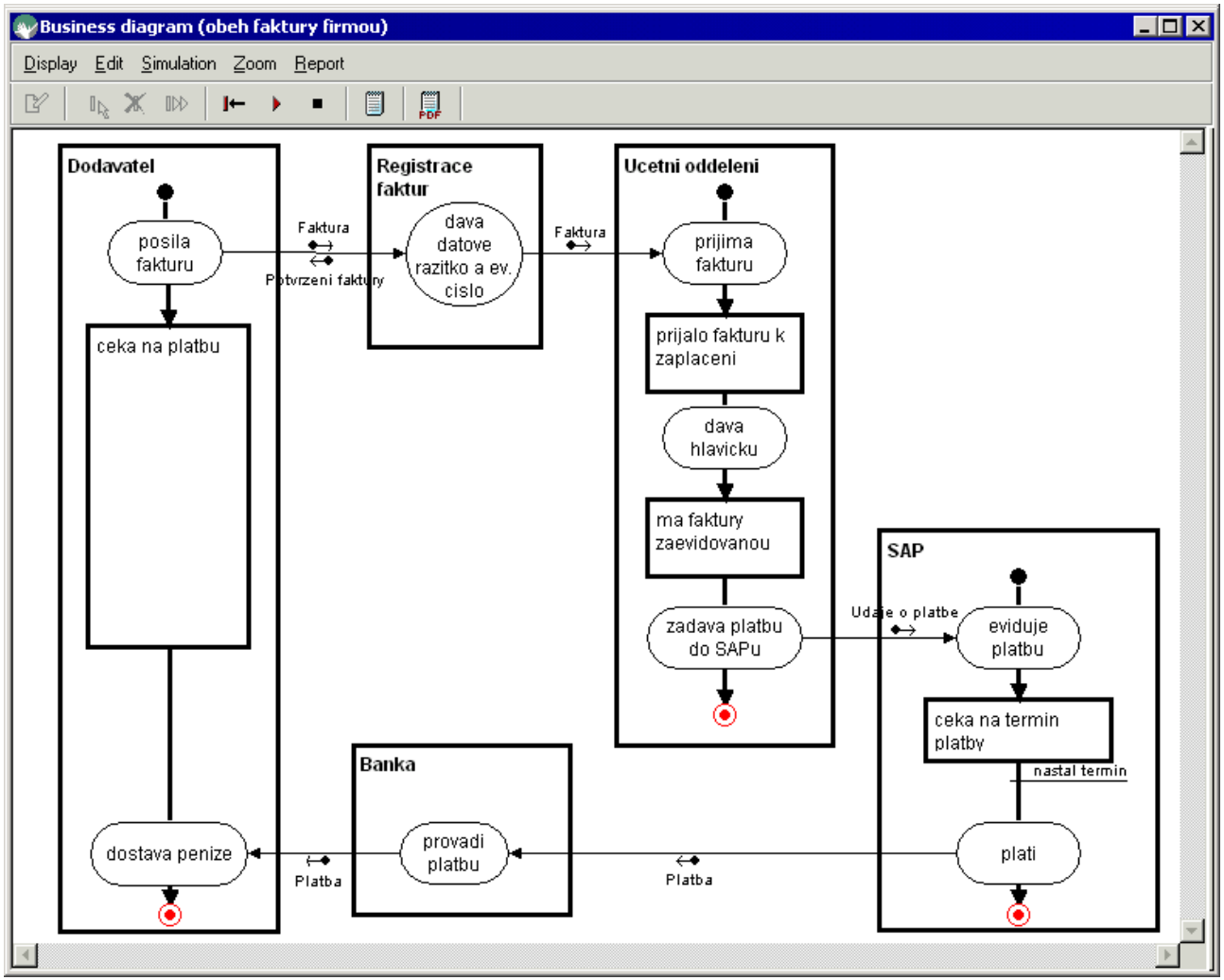
Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



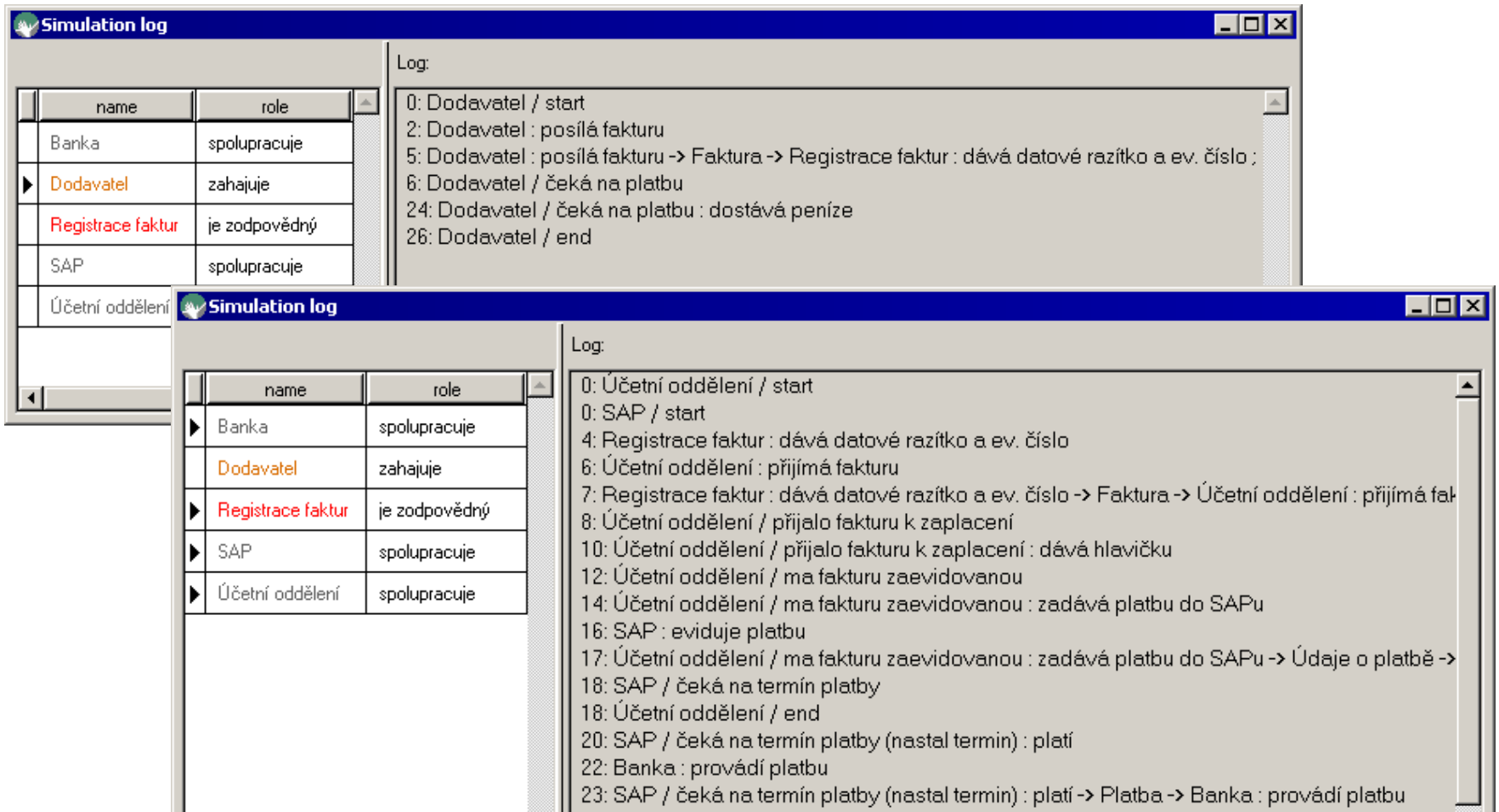
Business Map - Procesní diagram - Simulace

Každý objekt, který v procesu participuje, má svoje v čase proměnlivé chování a komunikuje s druhými objekty.

Výsledný proces je dán sledem událostí vyvolávaných komunikacemi a datovými toky mezi objekty.



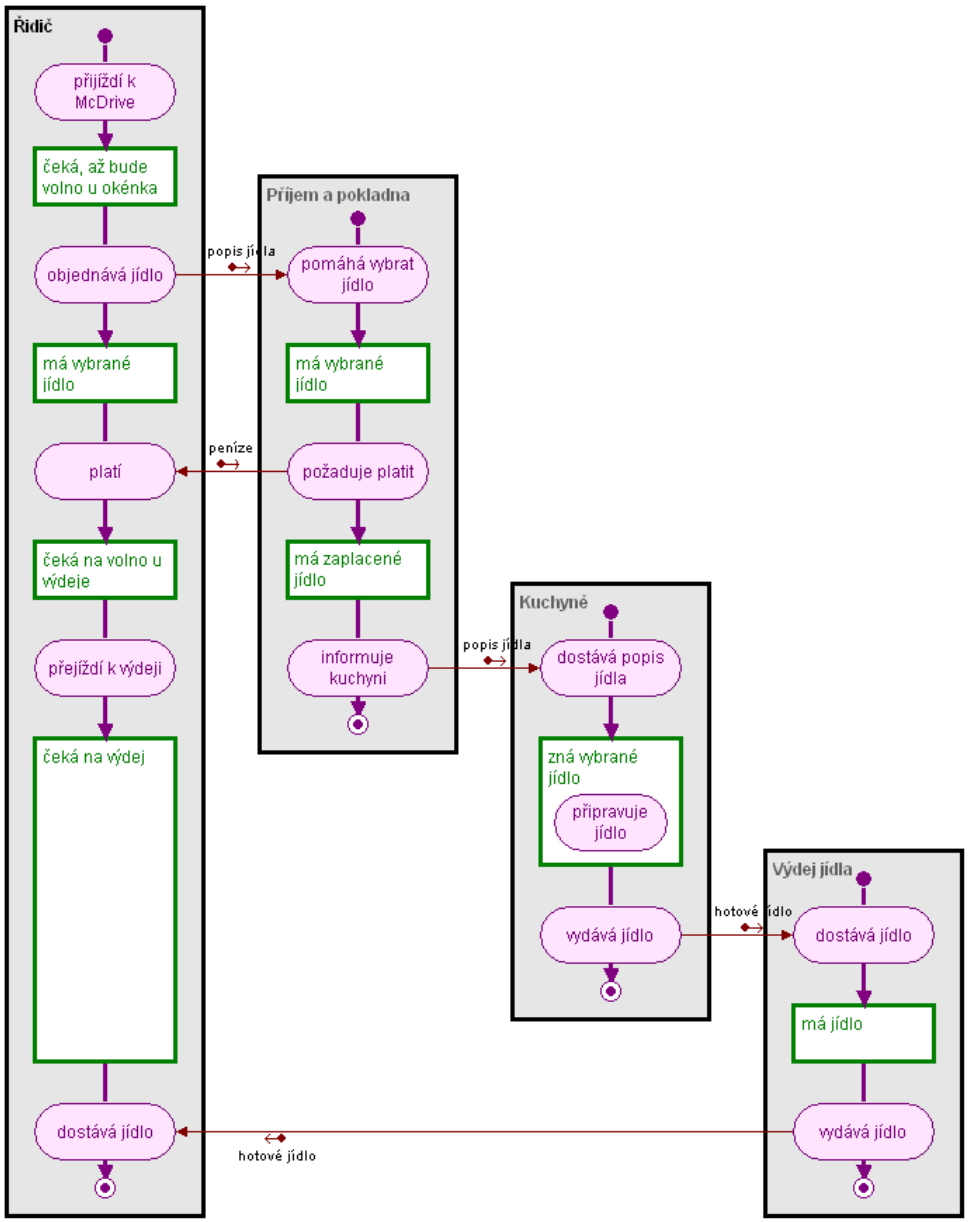
Business Map - Simulační záznam



Díky simulacím lze podrobně analyzovat role jednotlivých objektů nebo skupin objektů v modelovaném procesu.

Další příklady

MCDrive

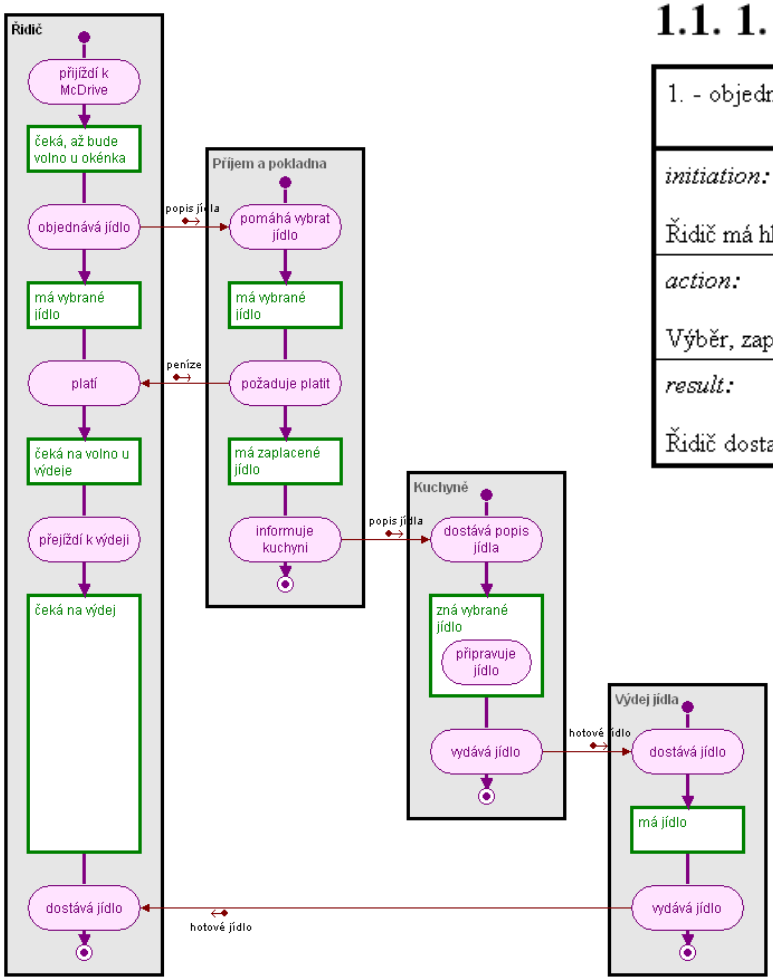

Řidič
přijíždí k McDrive
čeká, až bude volno u okénka
objednává jídlo
má vybrané jídlo
platí
čeká na volno u výdeje
přejíždí k výdeji
čeká na výdej
dostává jídlo
popis jídla
penize
hotové jídlo
Příjem a pokladna
pomáhá vybrat jídlo
má vybrané jídlo
požaduje platit
má zaplacené jídlo
informuje kuchyni
popis jídla
Kuchyně
dostává popis jídla
zná vybrané jídlo
připravuje jídlo
vydává jídlo
hotové jídlo
Výdej jídla
dostává jídlo
má jídlo
vydává jídlo

Scénáře

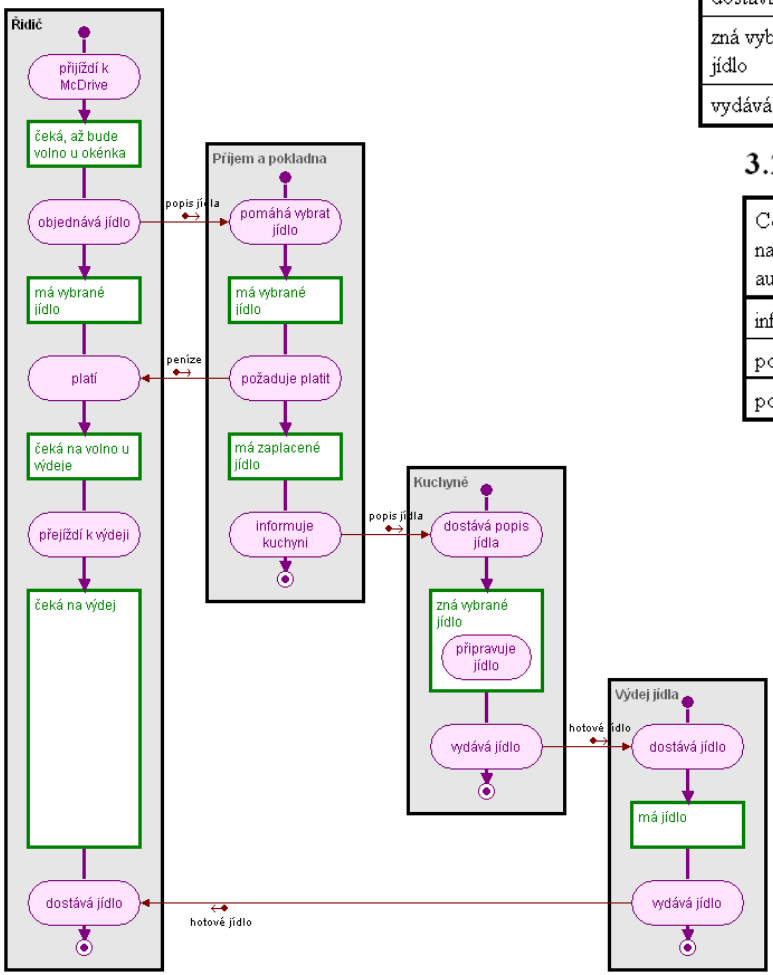
1. Scenarios

1.1. 1. - objednání jídla z auta

1. - objednání jídla z auta	Derived from: zákaznické procesy	
initiation: Řidič má hlad a chce si kopit jídlo v Mc Drive		*roles:* **Kuchyně** cooperates
action: Výběr, zaplacení a vydání jídla do auta		**Příjem a pokladna** cooperates **Řidič** performs
result: Řidič dostal jídlo		**Výdej jídla** cooperates



Modelové karty



3.2.1. Kuchyně

Collaborators in diagram with name '1. Objednání jídla z auta':	Příjem a pokladna	Řidič	Výdej jídla
dostává popis jídla	<<		
zná vybrané jídlo: připravuje jídlo			
vydává jídlo			>>

3.2.2. Příjem a pokladna

Collaborators in diagram with name '1. Objednání jídla z auta':	Kuchyně	Řidič	Výdej jídla
informuje kuchyni	>>		
pomáhá vybrat jídlo		<<	
požaduje platit		>>	

3.2.3. Řidič

Collaborators in diagram with name '1. Objednání jídla z auta':	Kuchyně	Příjem a pokladna	Výdej jídla
dostává jídlo			<<
objednává jídlo		>>	
platí		<<	
přejíždí k výdeji			
přijíždí k McDrive			

3.2.4. Výdej jídla

Collaborators in diagram with name '1. Objednání jídla z auta':	Kuchyně	Příjem a pokladna	Řidič
dostává jídlo	<<		
vydává jídlo			>>

Závěr

http://kii.pef.czu.cz/~merunka/books/DatoveModelovani/

knihy

Datové modelování - Alfa Publishing 2006

Umění systémového návrhu – Objektově orientovaná tvorba informačních systémů pomocí původní metody BORM, Grada, Praha 2003

Management of the Object-Oriented Development Process, Akron Publishing, Virgin Island 2005

Accounting Information Systems, South-Western Publishing, Mason-Ohio 2004

Využití metod umělé inteligence ve vodním hospodářství, nakladatelství Akademie věd ČR, Praha 2004

a dalších cca 5 vysokoškolských skript

časopisy

The BORM methodology: a third-generation fully object-oriented methodology, Knowledge-Based Systems 3(10) 2003, Elsevier Science Publishing, New York.

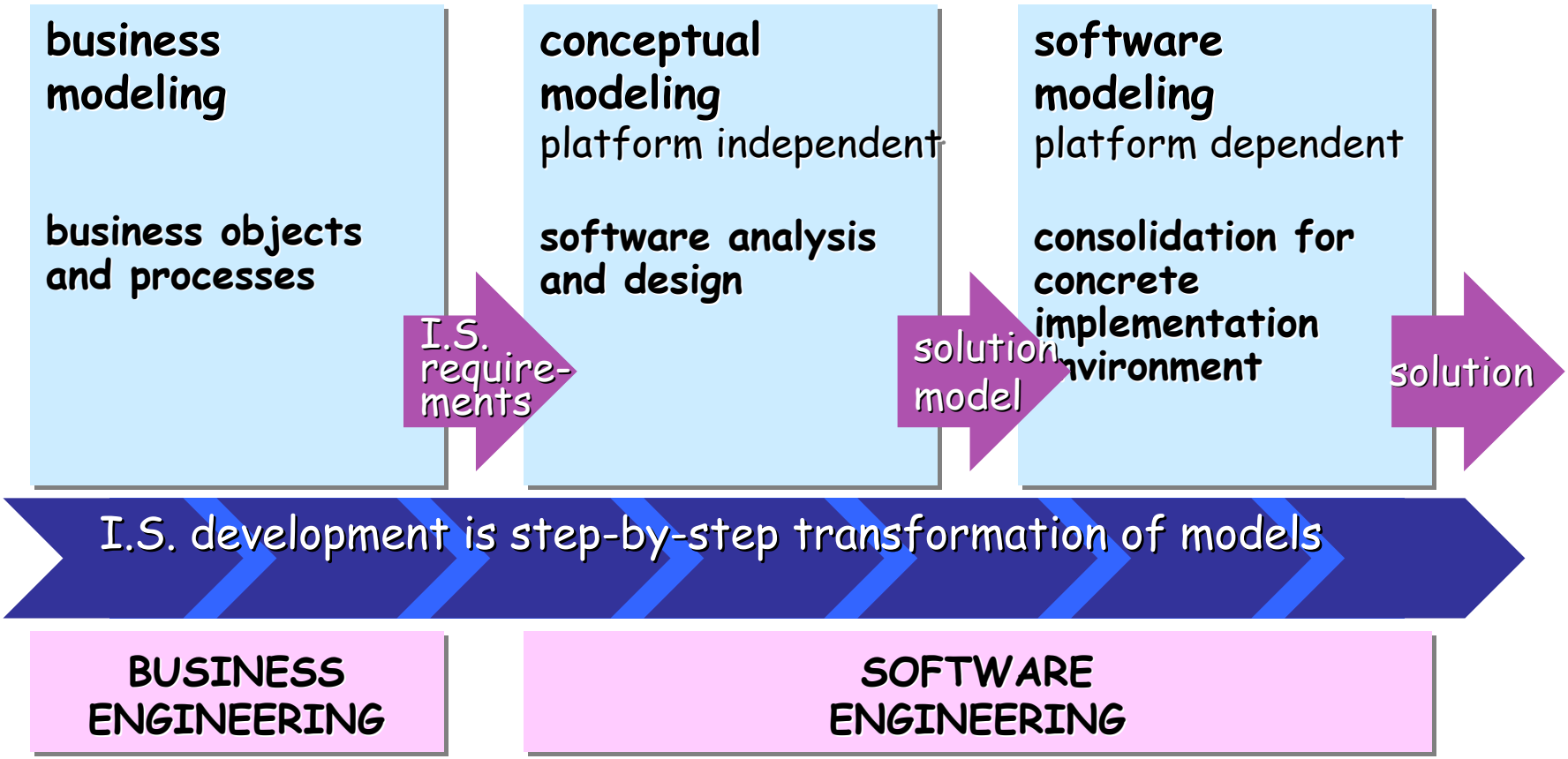
… a další 4 články ve vědeckých časopisech

konference

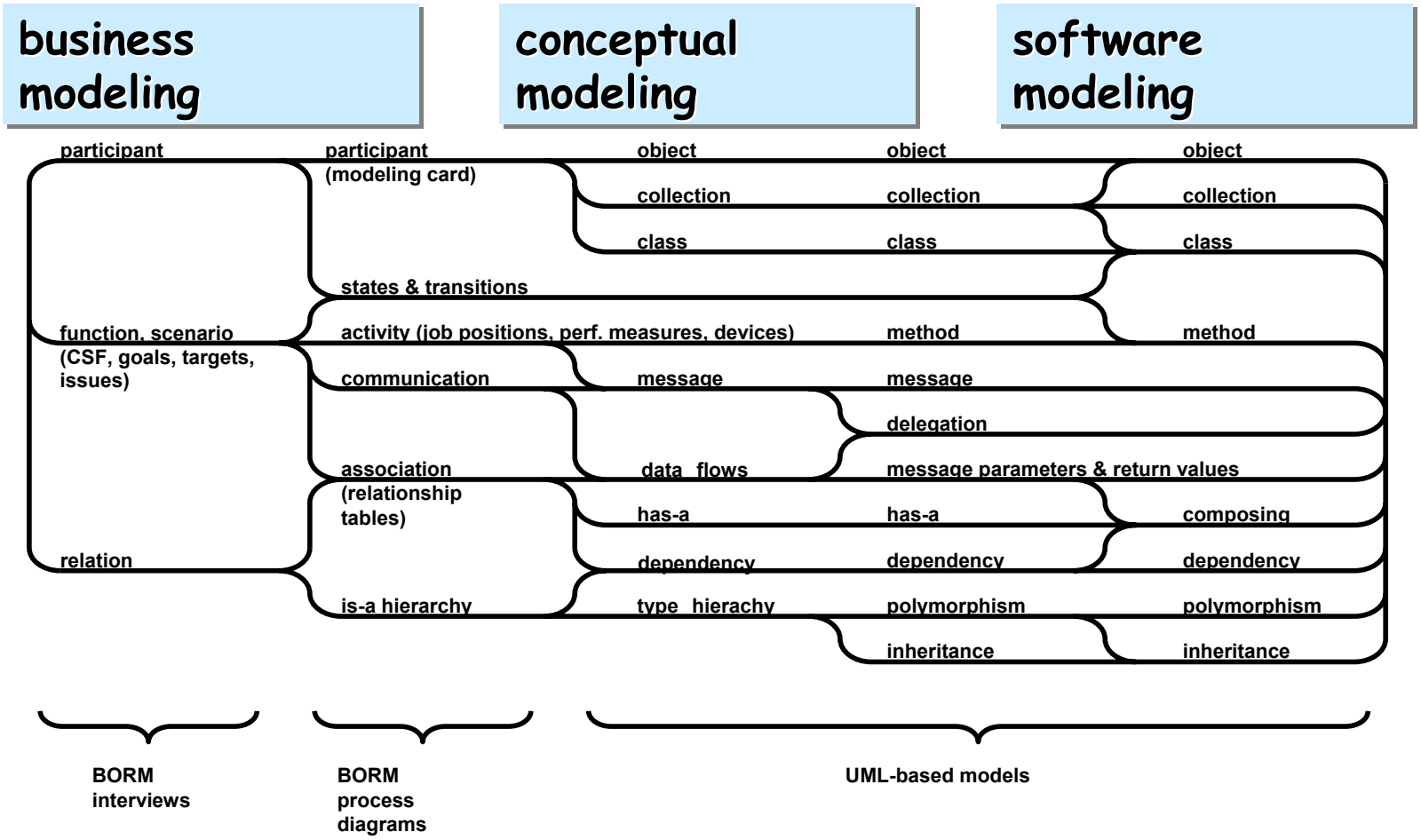
Process Modeling for Object Oriented Analysis using BORM Object Behavioral Analysis, in Proceedings of Fourth International Conference on Requirements Engineering, IEEE Computer Society, Chicago 2000.

… a dalších cca 20 příspěvků na konferencích doma i v zahraničí

BORM Philosophy



BORM Philosophy - Detail



IS-A ≠ subtype-supertype ≠ inheritance

The hierarchy of object in the phase of problem formulation is not totally identical with inheritance or the hierarchy of types.

Nevertheless, UML expresses these three similar, but not identical hierarchies in the same way.

During modeling, it is necessary to look at them as follows:

1. From the designer's perspective – new object designer. This hierarchy is a hierarchy of inheritance because inheritance is a tool for the development of new classes.

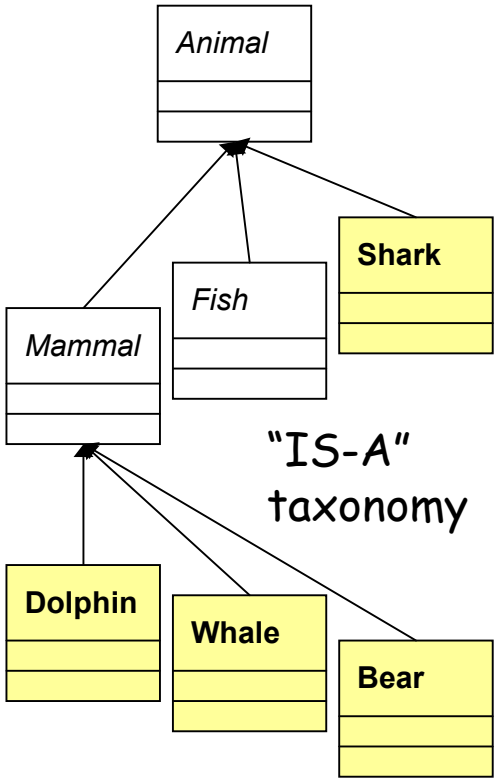
2. From the user's perspective. This perspective can be further divided as follows:

   a) From the perspective of usability – a view of an application programmer who needs to use the objects in his system but does not develop them. This hierarchy is the hierarchy of types.

   b) From the perspective of the user/analyst – the objects of lower-level classes are elements of the same domain as objects of upper-level classes. This hierarchy is called IS-A.

The Example

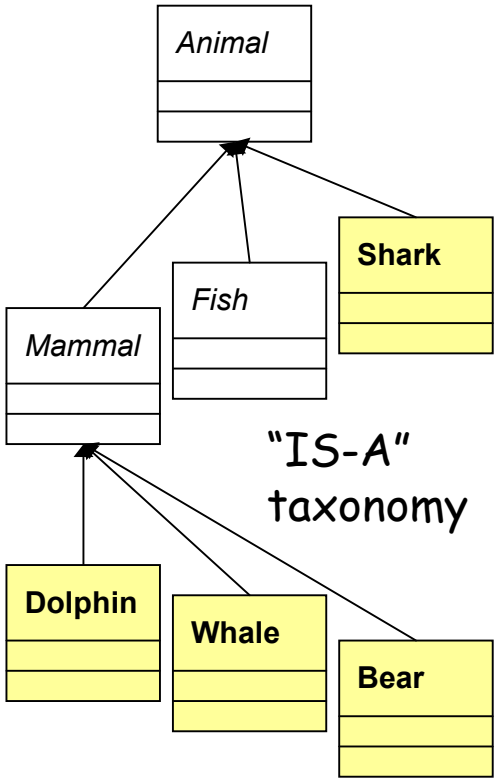
business modeling	conceptual modeling	software modeling



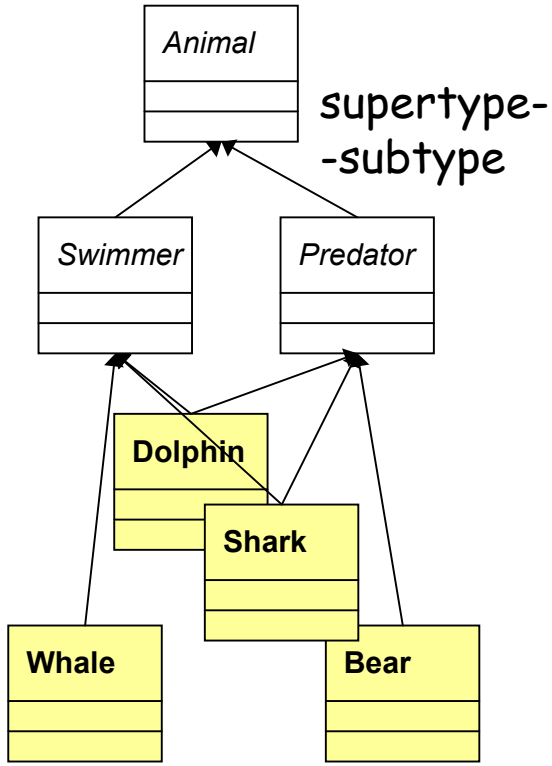
BUSINESS ENGINEERING	SOFTWARE ENGINEERING

The Example

business modeling



conceptual modeling



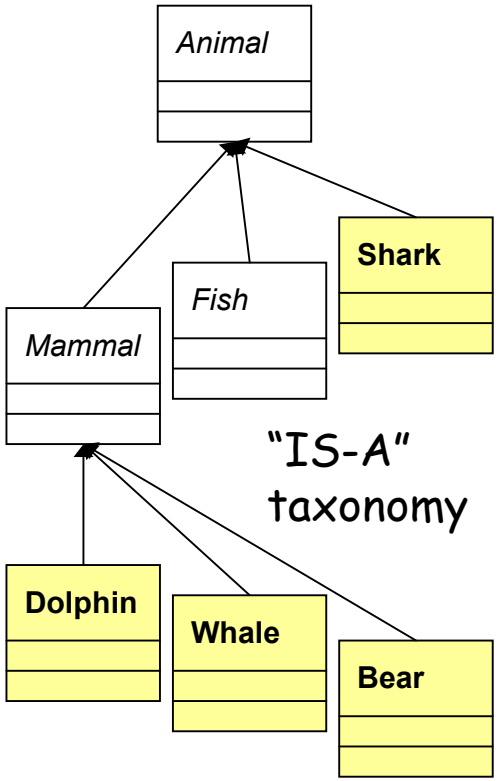
software modeling

BUSINESS ENGINEERING

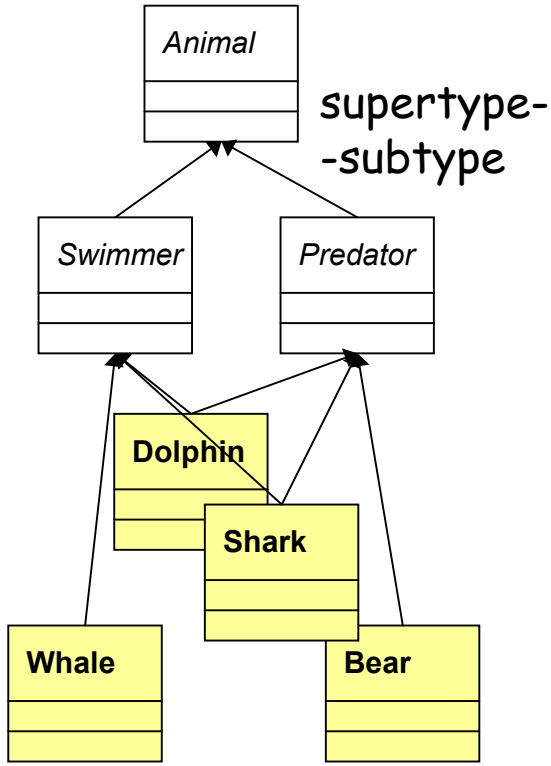
SOFTWARE ENGINEERING

The Example

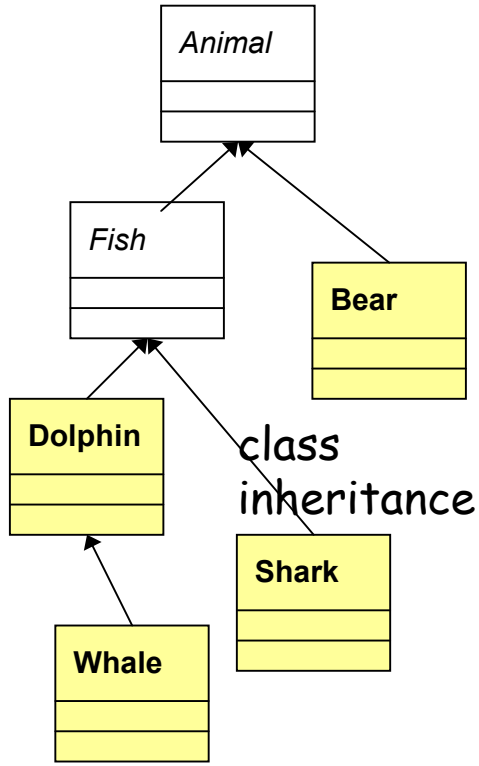
business modeling



conceptual modeling



software modeling

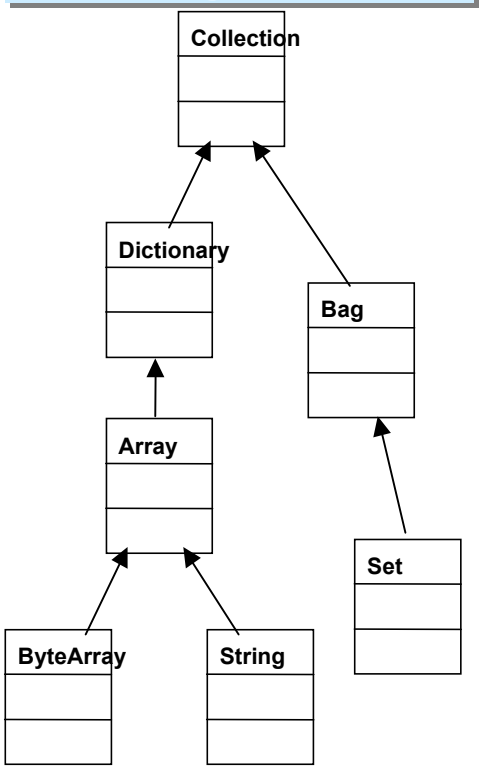


BUSINESS ENGINEERING

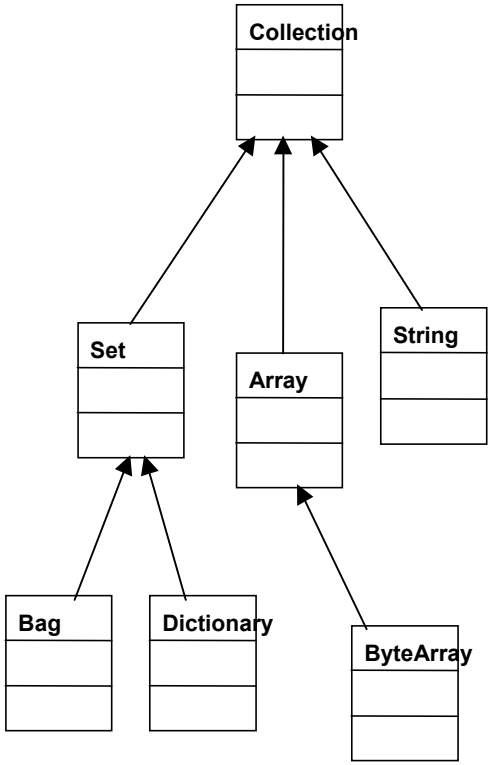
SOFTWARE ENGINEERING

Another Example

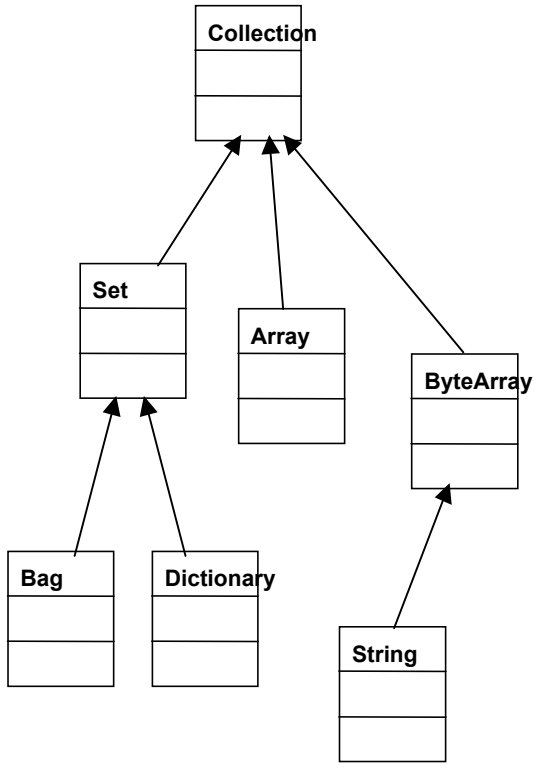
business modeling



conceptual modeling



software modeling



BUSINESS ENGINEERING

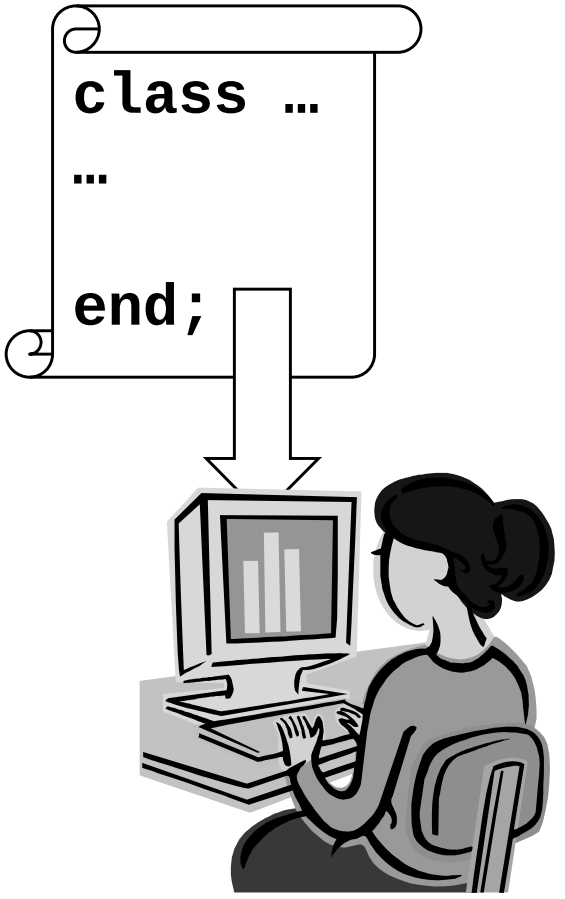
SOFTWARE ENGINEERING

X36SIN
Seminář o metodice projektu „Dlouhán“

Hlavní problém tvorby software

?

Jak to udělat
rychle,
efektivně,
bezchybně,
…



Jak postupovat?

- Jak postupovat by nám měla doporučit nějaká metodika - přehled metodik najdeme např. na „**Google directory**“.
- Dosud jsme probírali metodiky „**BORM**“ a „**Extrémní programování**“.
- Dnes se budeme zabývat metodikou vytvořenou v rámci projektu „**Dlouhán**“.
- Příště budeme probírat metodiky „**UP**“ a „**RUP**“ (17.5.).

Co to je projekt „Dlouhán“?

- Projekt vedený snahou vytvořit smysluplnou metodiku vhodnou pro malé a střední firmy, která by dostatečným způsobem zohledňovala dobré zásady tvorby SW, ale brala v úvahu možnosti menší firmy.
- Název „**Dlouhán**“ je zkratkou za „dlouhý“ celkový název projektu: „Dlouhodobý vzdělávací program řízení vývoje životního cyklu software a zavádění projektového řízení pomocí softwarových nástrojů do malých a středních podniků“:
  - http://www.dlouhan.org/
- Řešitelé:
  - AARON GROUP, s.r.o.: http://www.aarongroup.cz/
  - ILikeThis!, s.r.o.: http://www.ilikethis.cz/
  - ČVUT FEL: http://cs.felk.cvut.cz/webis/research/g20208.html

Představení projektu „Dlouhán“ z pohledu firmy Aaron Group, s.r.o.

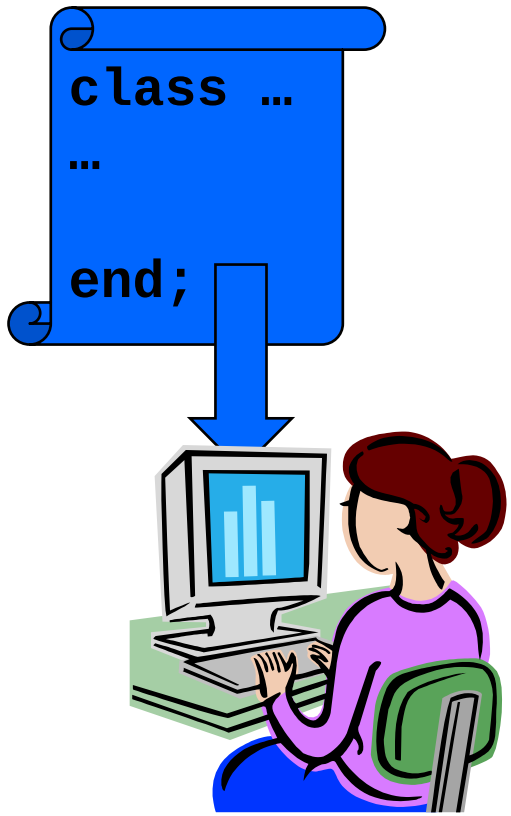
Ondřej Volráb

X36SIN
Seminář o metodice projektu „Dlouhán“

Hlavní problém tvorby software

?

Jak to udělat
rychle,
efektivně,
bezchybně,
…



Jak postupovat?

- Jak postupovat by nám měla doporučit nějaká metodika - přehled metodik najdeme např. na „**Google directory**“.
- Dosud jsme probírali metodiky „**BORM**“ a „**Extrémní programování**“.
- Dnes se budeme zabývat metodikou vytvořenou v rámci projektu „**Dlouhán**“.
- Příště budeme probírat metodiky „**UP**“ a „**RUP**“ (17.5.).

Co to je projekt „Dlouhán“?

- Projekt vedený snahou vytvořit smysluplnou metodiku vhodnou pro malé a střední firmy, která by dostatečným způsobem zohledňovala dobré zásady tvorby SW, ale brala v úvahu možnosti menší firmy.
- Název „**Dlouhán**“ je zkratkou za „dlouhý“ celkový název projektu: „Dlouhodobý vzdělávací program řízení vývoje životního cyklu software a zavádění projektového řízení pomocí softwarových nástrojů do malých a středních podniků“:
  - http://www.dlouhan.org/
- Řešitelé:
  - AARON GROUP, s.r.o.: http://www.aarongroup.cz/
  - ILikeThis!, s.r.o.: http://www.ilikethis.cz/
  - ČVUT FEL: http://cs.felk.cvut.cz/webis/research/g20208.html

Představení projektu „Dlouhán“ z pohledu firmy Aaron Group, s.r.o.

Ondřej Volráb

Jan Vraný

8.3.2006

eXtrémní Programování

Jan Vraný, vranyj@fel.cvut.cz

Co to je XP?

XP je agilní technika vývoje softwaru, která upřednostňuje:

- **teamovou spolupráci a interakci před formálními procesy a nástroji,**
- **fungující software před obsáhlou, komplexní dokumentací,**
- **spolupráci mezi zákazníkem a vývojáři před specifikacemi, zadáními, dohodnutými kontrakty,**
- **rychlou adaptaci na změny v zadání před dodržováním předem stanoveného plánu.**

12 technik používaných v XP

- **Planning game**
- **Small releases**
- **Metaphor**
- **Simple design**
- **Pair programming**
- **Testing**
- **Refactoring**
- **Collective code ownership**
- **Continuous integration**
- **40-hour week**
- **On-site customer**
- **Coding standards**

12 technik používaných v XP

Planning game

Software se vyvíjí v malých iteracích. Vždy se vyvíjí jen to, co zákazník v tu danou chvílí potřebuje.

Iterace se sestává ze:

- **sepsání user stories (zákazník)**
- **rozdělení user stories na úkoly (vývojáři)**
- **casových odhadů na jednotlivé úkoly (vývojáři)**
- **určení priorit (zákazník)**
- **vlastního programování (vývojáři)**
- **akceptace naprogramovaných user stories (zákazník)**

Small Releases

na konci každé iterace (naprogramování user story) se vytvoří nová verze (release). To znamená:

- **zákazník dostává nová verze poměrně často (jednou za 14 dní, jednou týdně, každý den),**
- **existuje okamžitá zpětná vazba, neboť zákazník může software neustále testovat a připomínkovat,**
- **zákazaník má k dispozici funkční část SW velmi rychle a může ho začít používat.**

Metaphor

Metafora je jednoduchá paralela nebo příběh, který popisuje budovaný systém slovy a pojmy, kteří všichni zůčastněni (vývojáři i zákazník) intuitivně chápou.

Slouži především k lepší komunikaci a porozumění mezi členy teamu.

Simple Design

Co přesně má software dělat a jak má vypadat víte v nejlepším případě až když ho potřetí přepisujete celý znova. Scott W. Ambler.

Proto je vhodné udržovat návrh tak jednoduchý, jak jen to jde. Použít to nejjednodušši, co ještě splní akceptační testy.

Neprogramovat pro budoucnost, netvořit frameworky, pokud to neni bezpodmínečně nutné.

Pair Programming

Každá řádka kódu je napsána dvěma programátory u jednoho počítače s jedním monitorem, s jednou klávesnicí.

To zajišťuje:

- **vyšší spolehlivost kódu, neboť programátoři se kontrolují navzájem,**
- **vyšší čitelnost a pochopitelnost kódu, neboť programátoři se kontrolují navzájem,**
- **všeobecné povědomí všech vývojářů o celém kódu**
- **a v neposlední řadě postupné vzájemné učení se vývojářů od sebe.**

Testing

Neustálé testování je jedna z fundamentálních technik XP, bez testování XP nemůže fungovat.

Vývojáři píší jednotkové testy na každou komponentu vyví-jeného systému, dokonce na každou třídu, tyto testy se spouští neustále.

Pokud neprochází všechny jednotkové a akceptační testu user stories, není možné pokračovat ve vývoji.

Refactoring

Refactoring je technika zlepšování stávajícího kódu beze změny dosavadního chování. Refaktoruje kvuli:

- **zjednodušení kódu**
- **optimalizaci kódu**
- **lepší čitelnosti, pochopitelnosti kódu,**
- **budoucí přidání nějaké další funkcionality**

Jistotu, že refaktorováním se nezmění stávající chování nám dávají jednotkové testy.

Collective Code Ownership

V XP teamu, kterýkoliv vývojář je oprávněn kdykoli modifikovat kterýkoliv kus kódu, pokud po modifikaci procházejí testy.

Continuous Integration

Veškerý kód je zpřístupněn všem vývojářům v centrálním repozitáři.

Kdykoli je úkol dokončen (naprogramován) a otestován, je uložen do centrálního repozitáře.

40-hour week

Psychicky nebo intelektuálné unavený vývojář je:

- **nevýkonný,**
- **chybující**
- **velmi naštvaný**

Proto je nutné vývojovému teamu zajistit klid na práci, důstojné pracovní prostředí a hlavně je nepřetěžovat.

On-site customer

On-site customer je člověk od zákazníka, který představuje typického uživatele vyvíjeného software a která je *neustále* k dispozici vývojářům aby:

- **zodpovídal otázky vývojářů během práce**
- **tvořil akceptační testy pro user stories**
- **testoval a podával zpětnou vazbu**

Coding Standards

Je nutné, aby *všichni* programátoři dodržovali stejné "coding standards", tedy:

- **stejné zarovnání kódu,**
- **stejné názvové konvence,**
- **stejně nastavené vývojové prostředí.**

Kdy má cenu použít XP?

- **Když je limitovaný čas nebo nebo financování.**
- **Když zákazník neví přesně co chce nebo není schopen dodat dostatečně přesné zadání.**
- **Když zákazník chce spolupracovat na projektu.**
- **Když se ví, že systém se bude měnit.**
- **Když členové teamu spolu dobře vycházejí a jsou schopni spolupracovat.**

Kdy nebude XP fungovat?

- **Je nutný početný (>15) team vývojářů?**
- **Jak dlouho trvá cyklus uprava kódu - překlad, linkování - spuštění? Více než 5 minut? Vice než hodinu?**
- **Jak douho běží testy? Minuty? Hodiny?**
- **Je nemožné dostat zákazníka na pracoviště?**
- **Nesouhlasí zákazník s XP? Nechápe dobře jeho principy?**

Pokud alespoň na jednu otázku odpovíte ano, XP zřejmě nebude nejlepší volba...

eXtrémní Programování

- **Představuje alternativu ke klasickému vodopádovému modelu vývoje SW.**
- **XP není nekoordinované hackování softwaru zdivočelými programátory - hackery.**
- **Aby XP fungovalo, team musí mít jiste zkušenosti a musí bezpodmínečně dodržovat *všech* 12 technik.**
- **XP není všelék, hodí se jen někdy. Pokud se ale správně nasadí, mívá lepší výsledky za kratší čas.**

Zkušenosti se zaváděním formalizovaných postupů do řízení projektů v malých a středních firmách

Ing. Ondřej Volráb
AARON GROUP
e-mail: volrab@aarongroup.cz

Náplň semináře

- Vymezení pojmů
- Vývoj SW a jeho specifika
- Specifika malých organizací
- Proces zavádění metodiky do organizace
- Přínosy pro organizaci
- Doporučení a diskuse

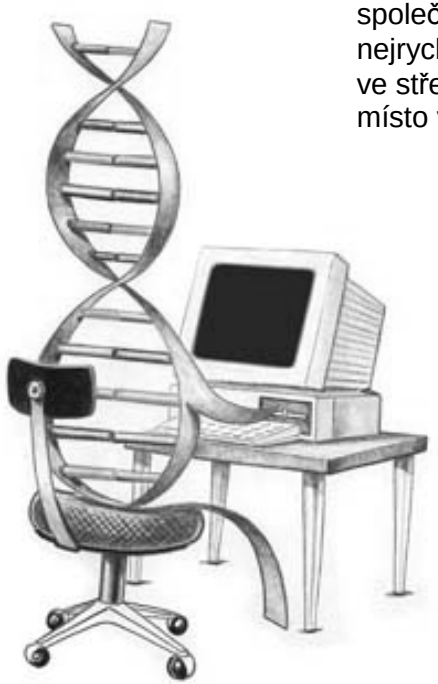
AARON GROUP

10 let

AARON GROUP úspěšně působí na českém trhu od roku 1997. Do obchodního rejstříku byla zapsána v květnu roku 1998 jako společnost s ručením omezeným se základním jměním 600 000 Kč, Dnes má společnost přes 20 stálých zaměstnanců.

Deloitte. Technology Fast 50

... in Central Europe
Po čtvrté v řadě byla AARON GROUP společností Deloitte vyhlášena v žebříčku TOP50 nejrychleji se rozvíjejících technologických společností ve střední Evropě s růstem 551 % v letech 2001-2005 (5. místo v ČR).



motto: **... aby informace pracovaly pro Vás.**

ALTRON

Letiště Praha

ČESKÉ DRÁHY, a.s.

LETUSKA.CZ

ČSA CZECH AIRLINES

| STUDENT | AGENCY |

Náplň semináře

- Vymezení pojmů
  - Řízení projektů
  - Vývoj software
  - Standardizované postupy
- Vývoj SW a jeho specifika
- Specifika malých organizací
- Proces zavádění metodiky do organizace
- Přínosy pro organizaci
- Doporučení a diskuse

Řízení projektů

- Předmětem bylo/je:
  - Řízení projektů ve smyslu vývoje SW a činností bezprostředně souvisejících
- Předmětem nebylo:
  - Řízení dalších typů projektů jako jsou konzultační služby, monitoring provozovaných aplikací, personální řízení, marketing…

Vývoj software

- Má jasně definovaný cíl
  - Business analýza
  - Detailní sběr požadavků
  - Způsob testování výsledku

    ...
- Před zahájením vývoje se co nejpřesněji definuje způsob vývoje, použité metody a nástroje
- Dosažení cíle je garantováno plánem vývoje

Standardizované postupy

- Přesnost plánu závisí na schopnosti odhadnout náročnost práce
- Pro schopnost odhadovat je nezbytné dokázat
  - Změřit velikost výstupů plánované činnosti
  - Definovat, jak bude výstupů dosaženo
- Standardizace postupů je pro plánování nezbytná
  - Umožňuje předem definovat, jak se bude pracovat
    - Předem vím, jak se co dělá
    - Jsem schopen zajistit, že všichni pracovníci práci dělají určitým způsobem (včetně nových pracovníků)
  - Celkové množství práce je možné změřit před jejím vykonáním

Náplň semináře

- Vymezení pojmů

- Vývoj SW a jeho specifika

- Specifika malých organizací
- Proces zavádění metodiky do organizace
- Přínosy pro organizaci
- Doporučení a diskuse

Čím je vývoj SW specifický

- Stále se mění pracovní prostředky
  - V podstatě v každém SW projektu se používají nové technologie (nástroje, postupy, šablony apod.)
- Složitý odhad celkové pracnosti
  - Pro nové technologie neplatí staré odhady
  - Změna pracovních nástrojů se promítá do změny pracovních postupů
  - Náročnost požadavků, které má projekt splnit, často není na první pohled zřejmá
- Změnové řízení v rámci projektu
  - Požadavky na vyvíjený sw se v průběhu projektu často mění
  - Odhad dopadu požadované změny na projekt vyžaduje detailní znalost aktuálního stavu a podrobnou změnovou analýzu

Jak specifika software řešit

- Přijímat nové technologie opatrně
  - Ověřovat je v rámci interních projektů nebo studie proveditelnosti komerčních projektů
- Neustále zlepšovat odhady složitosti
  - Uchovávat historii odhadů
  - Vytvářet převodní tabulku výpočtu složitosti a časové náročnosti při změně technologie
- Detailní změnové řízení
  - Každou požadovanou změnu důkladně ohodnotit

Náplň semináře

- Vymezení pojmů
- Vývoj SW a jeho specifika

- Specifika malých organizací

- Proces zavádění metodiky do organizace
- Přínosy pro organizaci
- Doporučení a diskuse

Specifika malých organizací

- Ad-hoc přístup
  - Malé firmy mají konkurenční výhodu v dynamičnosti, kterou nesmí standardizací postupů ztratit
  - Malý tým zjednodušuje komunikaci a je možná nižší úroveň standardizace
- Menší vnitřní organizace
  - Pracovníci mají větší podíl na řízení a organizační změny jsou proto obtížnější
  - Existují obvykle jen minimální mechanismy interních kontrol – nejsou na ně kapacity ani vůle
- Nedostatek zdrojů
  - Malé organizace mají menší finanční a personální rezervy
  - Malé organizace nejsou zvyklé využívat externích konzultantů
  - Menší přínos ze zlepšení – zákazníci očekávají od menší firmy vysokou míru flexibility a ne pevně dané postupy.

Náplň semináře

- Vymezení pojmů
- Vývoj SW a jeho specifika
- Specifika malých organizací

- Proces zavádění metodiky do organizace
  - Zahájení projektu
  - Zpracování procesních oblastí
  - Vypracování metodiky
  - Nasazení metodiky
    - Školení pracovníků
    - Pilotní projekty
  - Oponentura adaptace metodiky

- Přínosy pro organizaci
- Doporučení a diskuse

Zahájení projektu

- Stanovení cíle – příprava na ISO certifikaci
- Harmonogram
- Zdroje potřebné k úspěchu projektu
- Jak vyhodnotit výsledek

- Definice postupů
  - Existuje řada metodik přímo navržených pro řízení vývoje SW (PMBOK, RUP, ITIL, BORM, agilní metodiky…)
  - Výhody standardních metodik
    - Nezapomene se na žádnou důležitou oblast
    - Využívání standardu je kladným signálem pro zákazníky
  - Nevýhody standardních metodiky
    - Aplikace vede k potlačení dobrých stránek a kompetenčních výhod konkrétní společnosti
    - Velmi často je zavedení pouze formální
    - Jsou obvykle určeny pro velké firmy a velké projekty

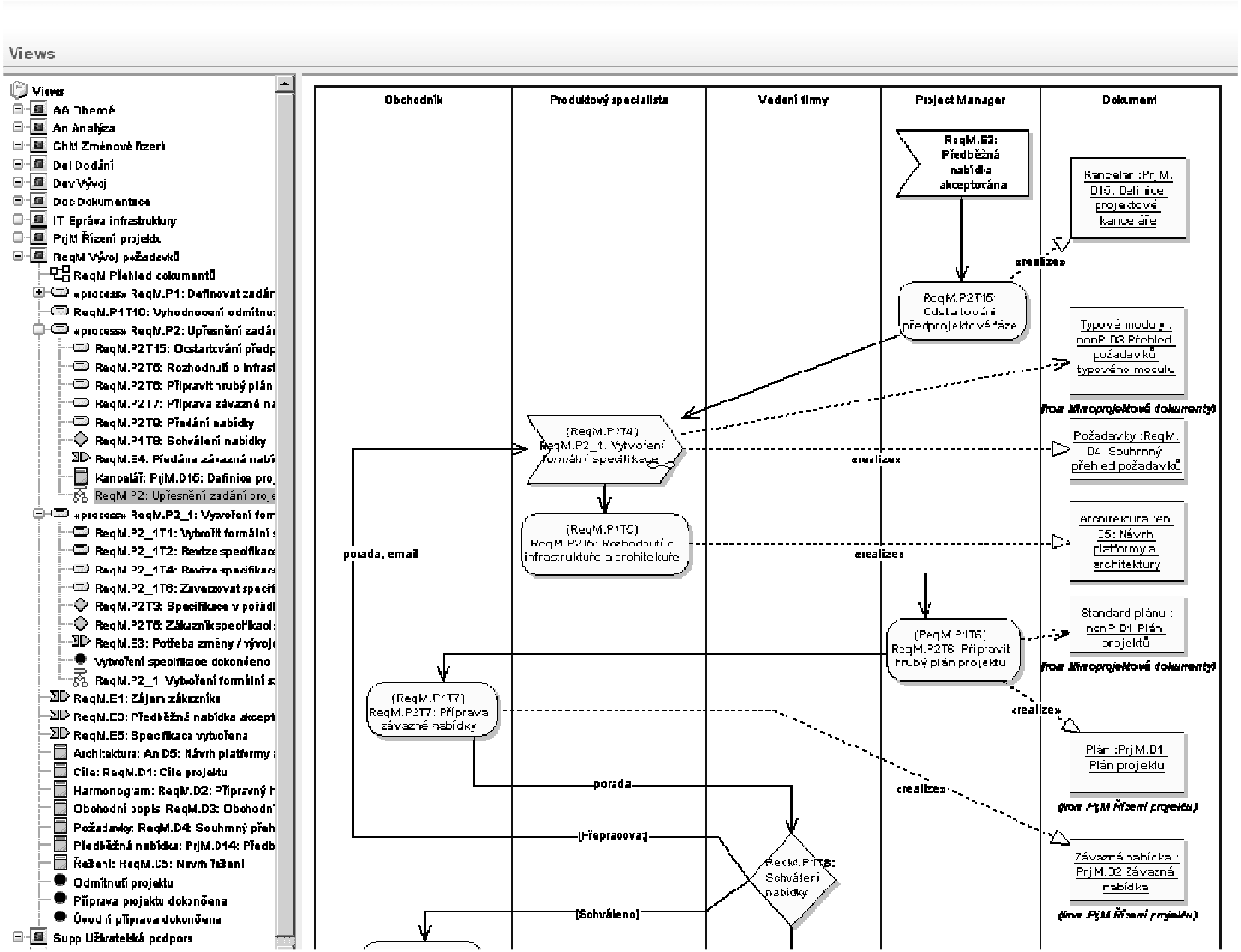
Zpracování procesních oblastí

- Vymezení a pokrytí procesních oblastí:
  - Vývoj požadavků
  - Řízení projektu
  - Analýza
  - Tvorba dokumentace software
  - Změnové řízení
  - Vývoj řešení
  - Testování
  - Předání hotového díla
  - Uživatelská podpora
  - Správa infrastruktury

Vypracování metodiky

- Detailní vymezení každé procesní oblasti – co je jejím obsahem, kde začíná a končí
- Procesní mapy, jejich popis a způsob rozšiřování
- Doporučené standardní metodiky a jejich kombinace
- Doporučené znalosti a způsob jejich získání (vybraná školení)
- Pracovní postupy (style guides) pro jednotlivé činnosti
- Šablony standardních dokumentů

Ukázka procesních map



Oponentura adaptace metodiky

- Každá zpracovaná procesní oblast prošla oponenturou ze strany:
  - Obou firem (AARON GROUP, ILikeThis!)
  - ČVUT
  - Externího konzultanta
- Všechny nasazené procesy jsou průběžně oponovány externím konzultantem

Nasazení metodiky

- Školení pracovníků
  - Systém pravidelného školení rozděleného dle procesních oblastí a participujících rolí
  - Opakování školení je nutnost!
  - Získání zpětné vazby
- Pilotní projekty
  - Ověření životaschopnosti metodiky a její průběžné vylepšování na základě reálných projektů
    - Komerčních (zohlednění vyšší časové náročnosti)
    - Interních (s podporou managementu)
  - Neověřuje se jen metodika samotná, ale rovněž míra přijetí ze strany všech pracovníků

Klíčové faktory úspěchu

- Stanovení důvodu, proč to firma dělá
- Cíl musí být všem ve firmě jasný
  - Každý pracovník sleduje změny ze své vlastní role
  - Nebezpečí, že cíle změn se budou rozcházet
- Standardizace postupů je zásadní zásah do firmy, který může také uškodit!
- Rizika špatné práce
  - Poškození firmy špatně navrženými procesy
  - Nevhodné načasování (nedostatek času)
  - Nedostatečné zdroje

Náplň semináře

- Vymezení pojmů
- Vývoj SW a jeho specifika
- Specifika malých organizací
- Proces zavádění metodiky do organizace

- Přínosy pro organizaci

- Doporučení a diskuse

Přínosy pro malou organizaci

- Zaměření na to, co přinese největší efekt
  - Formalizace postupů plánování projektů
  - Zpřesňování způsobů odhadů náročnosti
  - Zlepšování postupů interních kontrol
- Vymezení kompetencí a odpovědností
  - V rámci projektů
  - V rámci celé organizace
- Standardizace postupů vzhledem k zákazníkovi
  - Proces schvalování zadání a harmonogramu projektu
  - Proces změnového řízení
  - Proces předávání výsledků projektu
- Standardizace komunikace
  - Zápisy z porad a jejich správa
  - Standardizace projektové adresářové struktury

Zabezpečení trvalého zlepšování

- Nic není na poprvé dokonalé
  - Procesy a postupy je třeba trvale zlepšovat
  - Je třeba definovat vlastníka procesů
  - Je nutné interně auditovat, jestli všechny činnosti
    - Slouží svému účelu
    - Jsou efektivní
    - Jsou vykonávány správnými lidmi
- Nic není dokonalé věčně
  - Vývoj firmy = potřeba upravovat její procesy
  - Nutnost definovat osobu odpovědnou za
    - Sledování změn v projektech a aktualizaci pracovních postupů
    - Sledování zkušeností a schopností pracovníků
    - Průběžně školit stávající i nové pracovníky

Náplň semináře

- Vymezení pojmů
- Vývoj SW a jeho specifika
- Specifika malých organizací
- Proces zavádění metodiky do organizace
- Přínosy pro organizaci

- Doporučení a diskuse

Konzultant a podpůrný SW

- Konzultant
  - Ano, pro posouzení současného stavu
  - Ano, pro získání know-how
  - Ne, pro dlouhodobý rozvoj a údržbu metodiky

- Software pro řízení procesů
  - Ano, pokud je dostatečně flexibilní pro dlouhodobou údržbu pracovníky firmy
  - Ne, pokud je vhodný jen pro určitý typ projektů

Diskuse

...děkuji za pozornost

UML NARUBY

Úvod do jazyka UML

1.1.2007

Bc. Tomáš Černý

email: tom.cerny@gmail.com

Abstrakt

V tomto článku bych chtěl čtenáře seznámit se základy jazyka UML. Cílem není v žádném případě vyčerpávající přehled různých diagramů, nýbrž jen základní nástin problematiky a stručné popsání významu jednotlivých konstruktů. Jednotlivé konstrukty jsou doplněny jednoduchým digramem, který slouží jako vzorový příklad použití konstruktu.

Po přečtení článku by čtenář měl být schopen rozlišit digramy pro popis struktury či dynamiky popisované aplikace. Měl by také vědět, kde a v jaké situaci konkrétní konstrukt použít.

Rejstřík:

Rejstřík obrázků:

Konvence

V této práci budou dodržovány běžné konvence, které usnadní orientaci při čtení.
Zde je stručný přehled typografických konvencí

- Nové pojmy jsou při svém prvním výskytu zvýrazněný *kurzívou.*
- Důležité části textu budou zvýrazněný **tlouštku** fontu**.**
- Vše co se týká programování bude zapsáno `neproporcionálním` písmem.
- Vše co se týká vstupů, výstupů z aplikace bude zapsáno tučným **`neproporcionálním`** písmem.

Kromě typografických konvencí bude použito speciálního odstavce, který doplní, vysvětlí informaci týkající se textu

Poznámka

Text doplňující informace, tato poznámka blíže objasňuje problém

Jazyk UML

1 Co je to UML

UML (Unified Modeling Language) je jazyk pro vytváření modelů, nebo také souhrn pravidel pro modelování, souhrn grafických notací k vyjádření analytických návrhových modelů. Jiný pohledem může UML definovat jako jazyk pro vizualizaci, specifikaci, stavbu a dokumentaci projektu.

Jazyk má hlavní čtyři části:

• notace	syntaxe	SuperStructure
• metamodel	sémantika	Infrastructure
• OCL		Object Constraint Language
• specifikace do výměnných formátů (Corba, XML)		Interchange

1.1 Softwarové inženýrství a UML

Softwarové inženýrství používá jazyk *UML* jako grafický jazyk pro návrh, dokumentaci a specifikaci softwarových projektů, a to nejen objektově orientovaným přístupem. Jazyk umožňuje modelovat jednoduché i složité aplikace standardním způsobem pomocí jednotné syntaxe. Jednotný způsob zápisu umožňuje sdílení dokumentů mezi různými návrháři. Snadná pochopitelnost jazyka umožňuje vyjasnění požadavků pro analytika i zadavatele. Jazyk poskytuje pro konkrétní případy soubor diagramů, tyto diagramy se týkají konkrétní části problému a neberou si za cíl popsat chování aplikace jako celku.

Co je standardem jazyka UML je definováno standardizační skupinou *Object Management Group (OMG).* Ve světě softwarový firem se můžeme setkat s velkou spoustou mutací jazyka. Různé firmy mohou doplnit a rozšířit jazyk, aby lépe sloužil interním metodikám, a dokonce nabízejí školení pro zájemce, protože vylepšení se může uplatnit i v jiných firmách.

UML je základním kamenem ze kterého vycházejí objektově orientované nástroje *CASE (Computer Aided Software Engineering).* Zmíněné nástroje podporují návrh a analýzu projektů, a to tak že propojí jednotlivé digramy a jejich sdílením je zpřístupní vývojovému týmu. CASE nástroje umožní snadné vytváření a editaci standardních diagramů, práci s nimi a dále také nabízejí umělou inteligenci, která na základě digramů a jejích propojení umí vygenerovat částí kódu. Mezi nejznámější funkčnosti těchto nástrojů patří generování *SQL* příkazů na základě

*datového modelu. N*ástroje CASE jsou ideální pro návrh informačních a redakčních systémů, protože umožňují definovat veškeré potřebné diagramy. Trh nabízí široké spektrum produktů od rozsáhlých jako *Rational Rose* po jednodušší jako je *Visio*. Produkt Visio zde budu používat i já.

Kritikou jazyka UML je, že zohledňuje analytické přístupy jen z pohledu jazyků typu C++, Java a .NET a nikoliv jazyky typu SmallTalk.

2 Návrh aplikace

Postup při návrhu můžeme opět rozdělit do několika kroků, může to být úvodní studie a analýza. Analýzu, jelikož bývá rozsáhlejší, lze rozdělit na analýzu datovou, funkční a dynamickou. Při návrhu aplikace lze používat i digramy, které nejsou specifikované jazykem UML

Úvodní studie by měla obsahovat

- Vytyčení cíle
- Úvodní studie
- Katalog požadavků
- Diagram USE CASE (UML)
- Kontextový diagram
- Datový slovník
- Rozpočet
- Návrh řešení skrze HW a SW
- Harmonogram prací (Gant)
- Přidělení úloh členům týmu – matice zodpovědnosti

Analýza

- Datový model
    - Class diagram (UML)
    - E-R diagram
    - Popis modelu a integritní omezení
    - Datový slovník
- Funkční model
    - USE CASE (UML)
    - Sekvenční diagramy (UML)
    - Diagram aktivit (UML)
    - Kontextový diagram
    - Diagram datových toků
    - Mini-specifikace
    - Hierarchie funkcí
    - Datový slovník
- Dynamický model
    - Stavový diagram (UML)
    - Sekvenční diagram (UML)
    - Datový slovník
    - Regulární výrazy

3 Diagramy UML

Diagramy jsou základním konstruktem jazyka. Digramy můžeme rozdělit do tří základních tříd, kde první se týká struktury projektu, dále chování, pod nímž lze najít související celek diagramů interakce. UML 2.0 definuje celkem 13 typů diagramů.

Strukturní diagramy

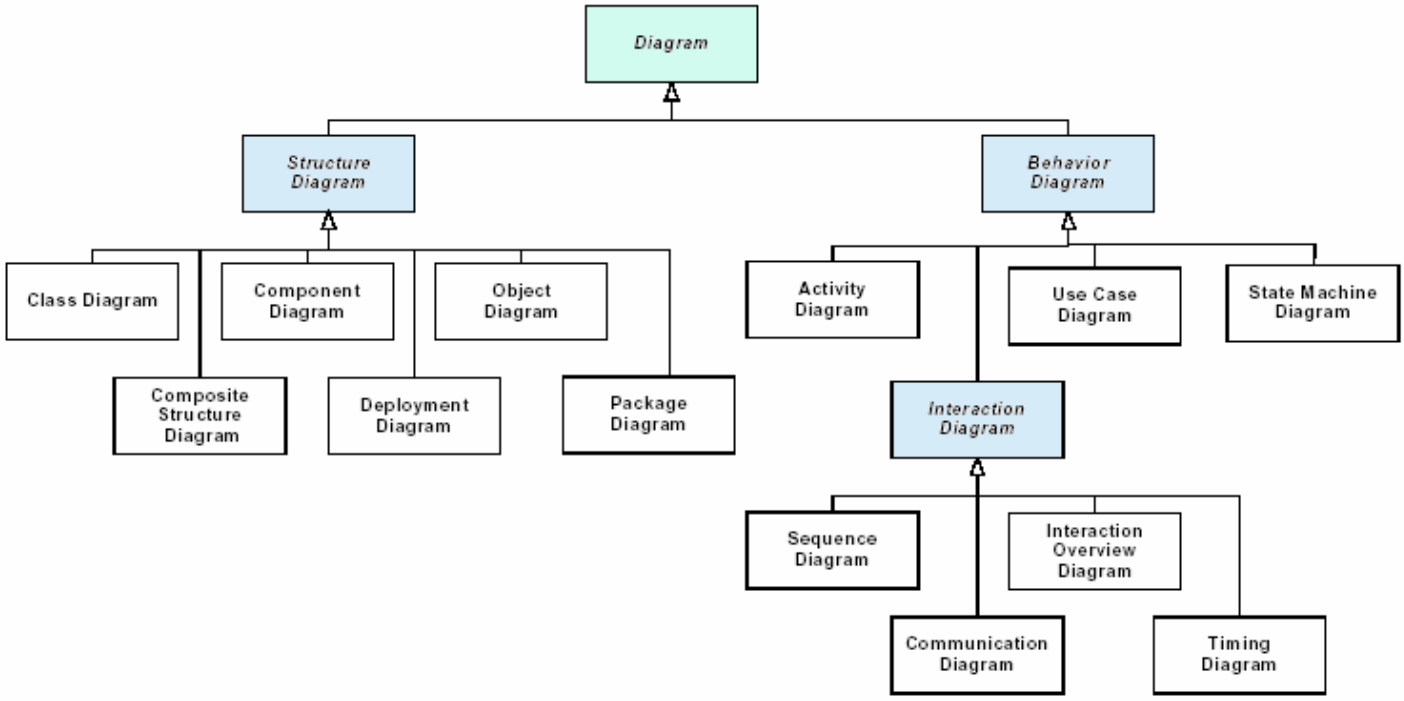
- Diagram tříd (Class diagram)
- Diagram komponent (Component diagram)
- Composite structure diagram
- Diagram naszení (Deployment diagram)
- Objektový diagram (Object diagram)
- Package diagram

Diagramy chování

- Diagram aktivit (Activity diagram)
- Stavový diagram (State Machine diagram)
- Případy užití (Use case diagram)

Diagramy interakce, podmnožina digramů chování

- Diagram kolaborace (Communication diagram)
- Interaction overview diagram (UML 2.0)
- Scénáře událostí (Sequence diagram)
- UML Timing Diagram (UML 2.0)

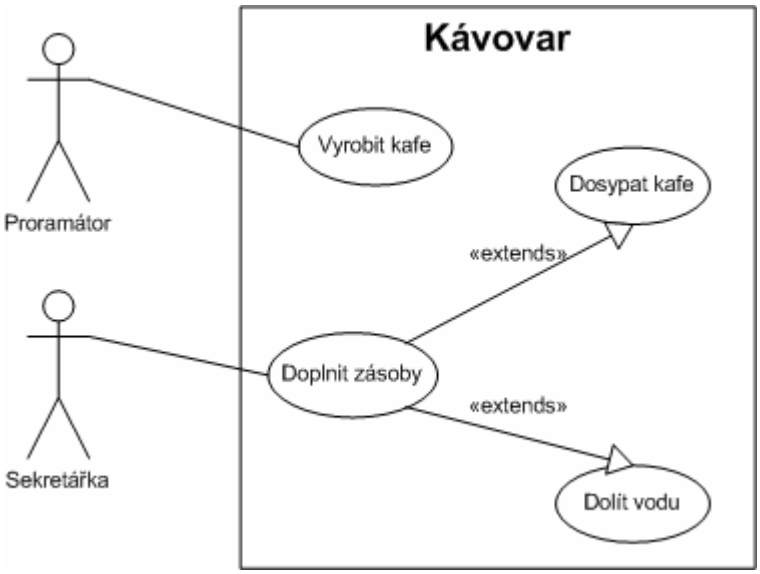


Obrázek 1. Zařazení diagramů

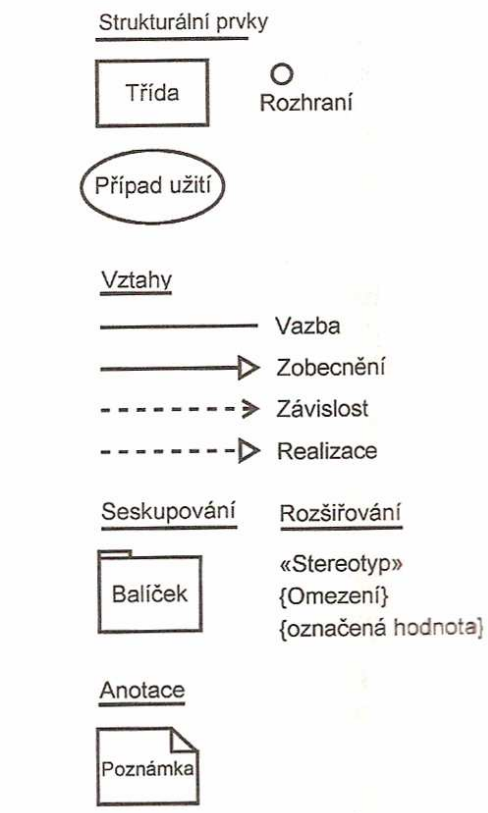
3.1 Diagramy chování

3.1.1 USE CASE – (model jednání, případy užití)

Diagramy use-case nebo-li případy užití (někdy též model jednání) jsou v praxi velmi oblíbené a často používané. Jsou vhodné především pro znázornění možných případů použití aplikace nebo také znázornění událostí, na které systém bude reagovat. Use-case diagram rozlišuje aktéry, jenž komunikují se systémem, dále vymezuje hranici systému, jenž uzavírá události, které v systému mohou nastat při vzájemné komunikaci s aktérem.



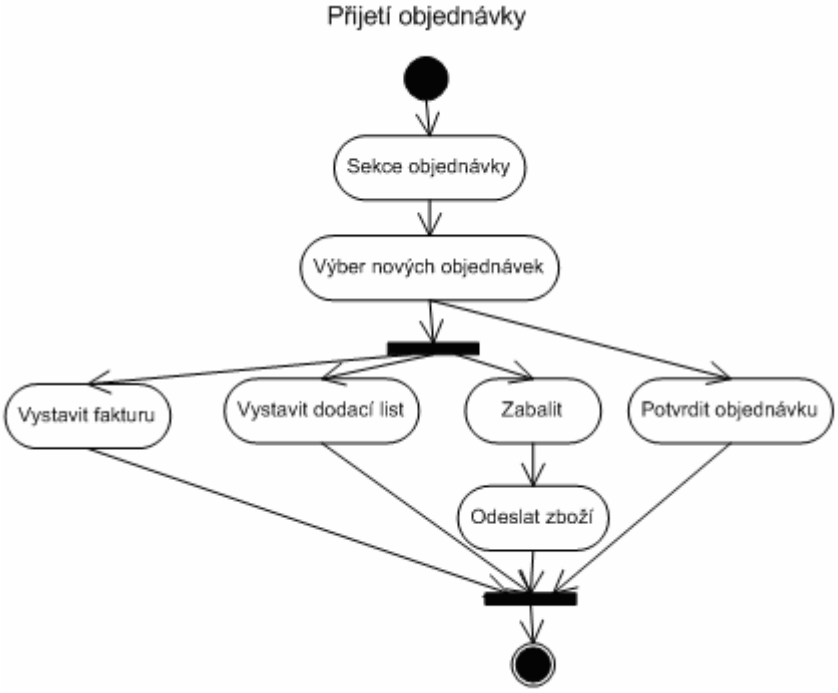
Obrázek 2. USE CASE - kávovar



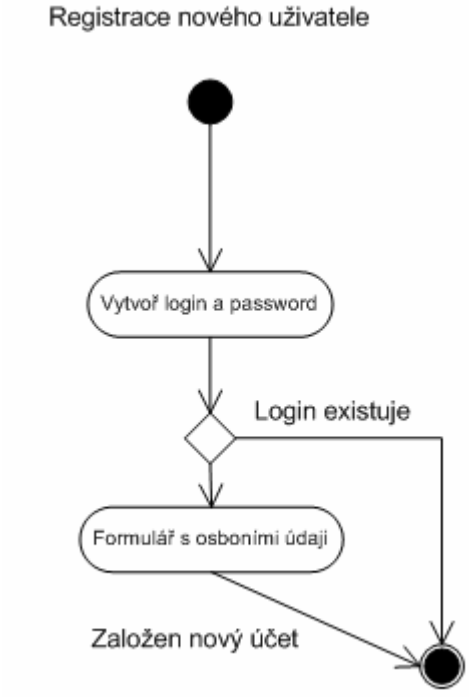
Obrázek 3. USE CASE – prvky

3.1.2 Diagram aktivit (Aktivity Diagram)

Diagram aktivit je k reprezentaci dynamiky procesů v systému, zaměřené hlavně na vnitřní chování. Zobrazení řídících toků (přechodů) mezi akcemi (aktivitami) v systému od počátečního bodu po jeden nebo více koncových bodů. Důraz se klade na zobrazení pořadí aktivit. Je vhodný zejména pro modelování průběhu jednotlivých Use-Case a operací v třídách. Diagram umožňuje zobrazit rozdělení na paralelní aktivity a specifikovat synchronizační bod.



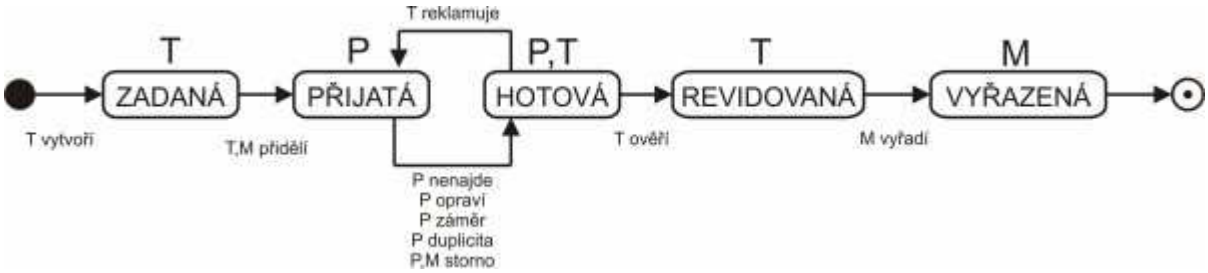
Obrázek 4. Diagram aktivit, příklad 1



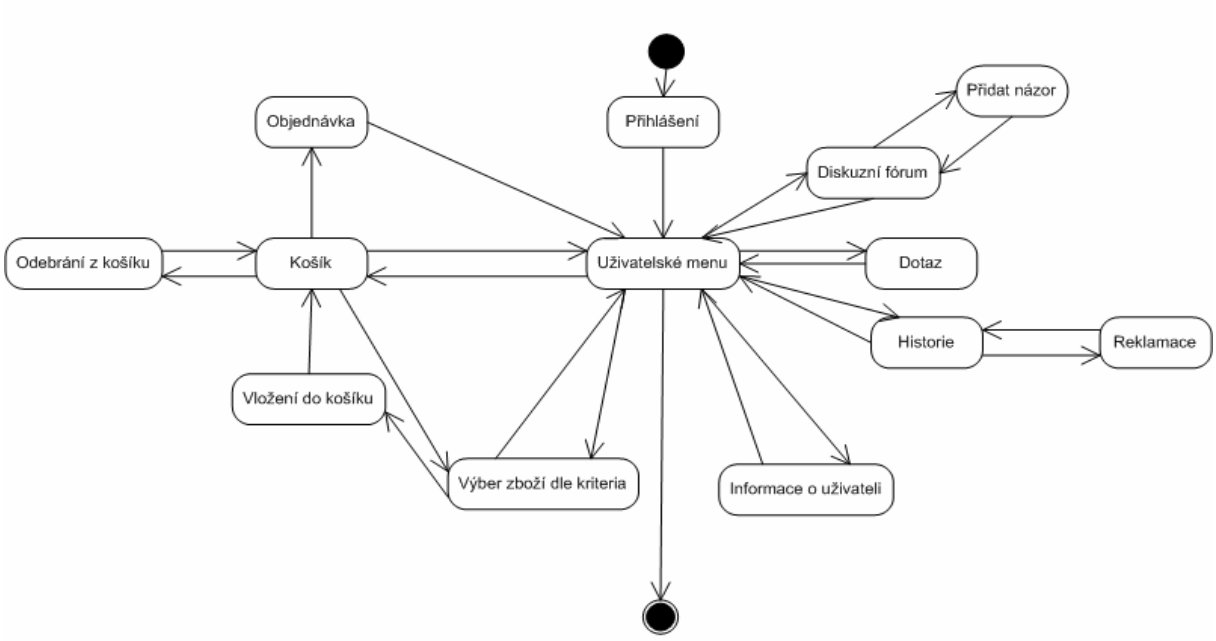
Obrázek 5. Diagram aktivit, příklad 2

3.1.3 Stavový diagram (State Machine diagram)

Objekt systému je u daném časovém okamžiku v určitém stavu. Mezi těmito stavy může přecházet pomocí akcí či událostí v systému. Diagram má zpravidla jeden vstpní a jeden výstupní stav. Je zde patrná analogie s automaty. Automat je pětice M(Q,T,d,q,F), kde Q je konečná množina vnitřních stavů, T je konečná vstupní abeceda, d je zobrazení z Q x T do Q, q je počáteční stav a F je množina koncových stavů. Pokud se přeneseme k této teorii, můžeme diagram převézt na regulární výrazy či gramatiku.



Obrázek 6. Stavový digram - Stavy chyby v systému



Obrázek 7. Stavový digram - pohyb uživatele v systému

Regulární výrazy

Regulární výraz nad abecedou A je definován jako

1. **θ, ε, a** jsou regulární výrazy pro a ϵ A (θ nulový a ε jednotkový prvek)
2. pokud **a, b** jsou regulární výrazy pak i **(a|b),(a·b),(a)*** jsou regulární výrazy

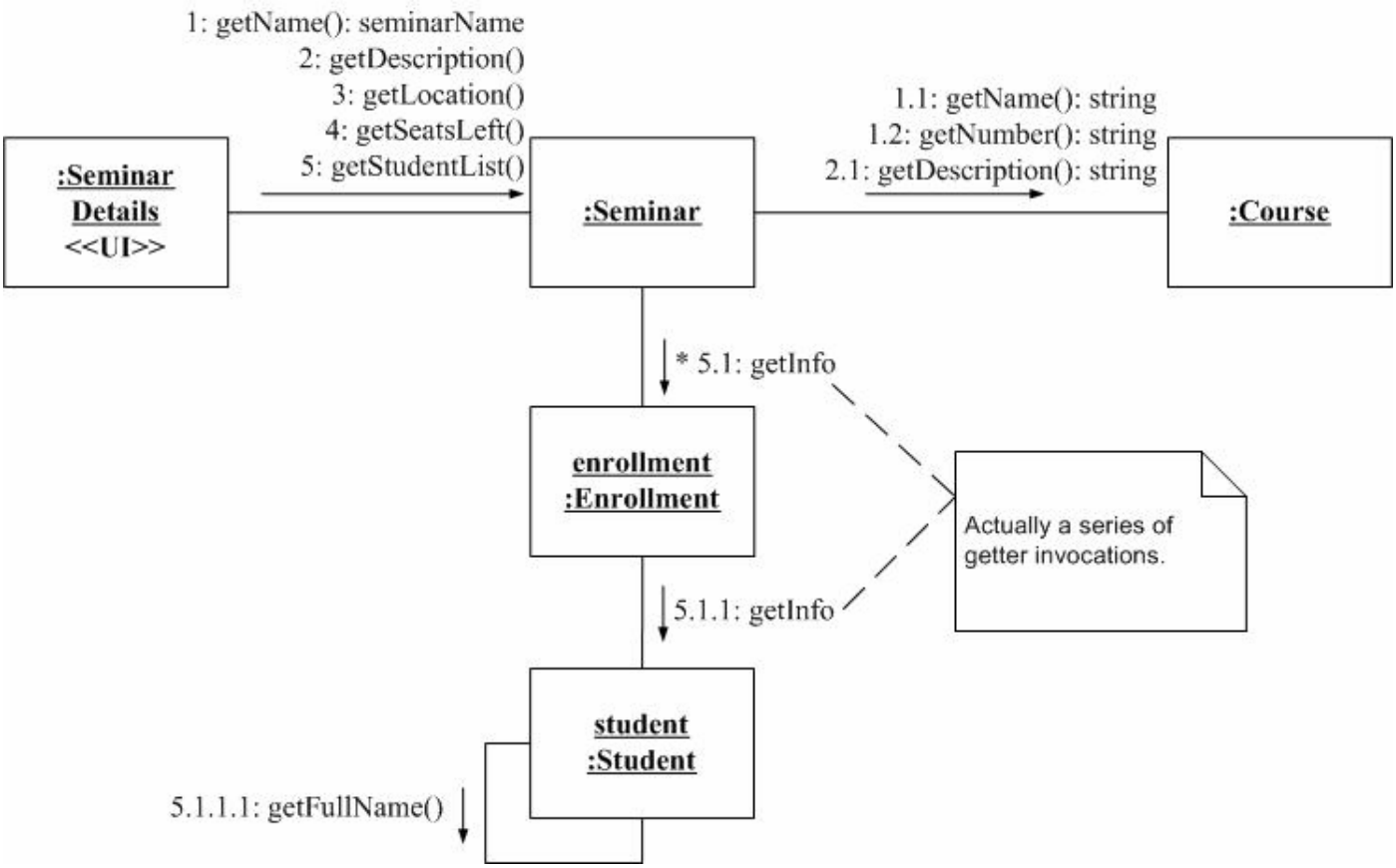
Zde lze definici ještě rozšířit a říci, že

<regV> = <akce> <regV> = #<reakce> <regV> = <paralelní>||<události>

3.2 Diagramy interakce

3.2.1 Diagram spolupráce (kolaborace, Communication diagram)

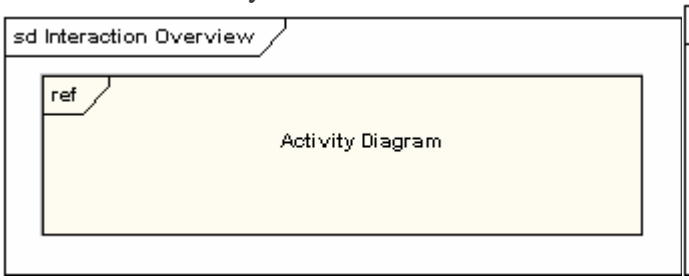
Diagram spolupráce ukazuje interakci mezi objekty systému, je podobný sekvenčnímu diagramu. Oproti sekvenčnímu diagramu který ukazuje, co se děje v čase, popisuje co se děje v prostoru. Objekty si posílají zprávy a dle nich reagují.



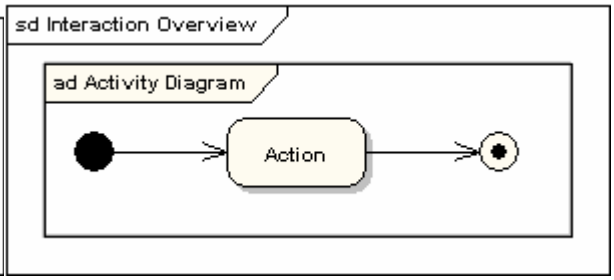
Obrázek 8. Diagram spolupráce

3.2.2 Interaction overview diagram (UML 2.0)

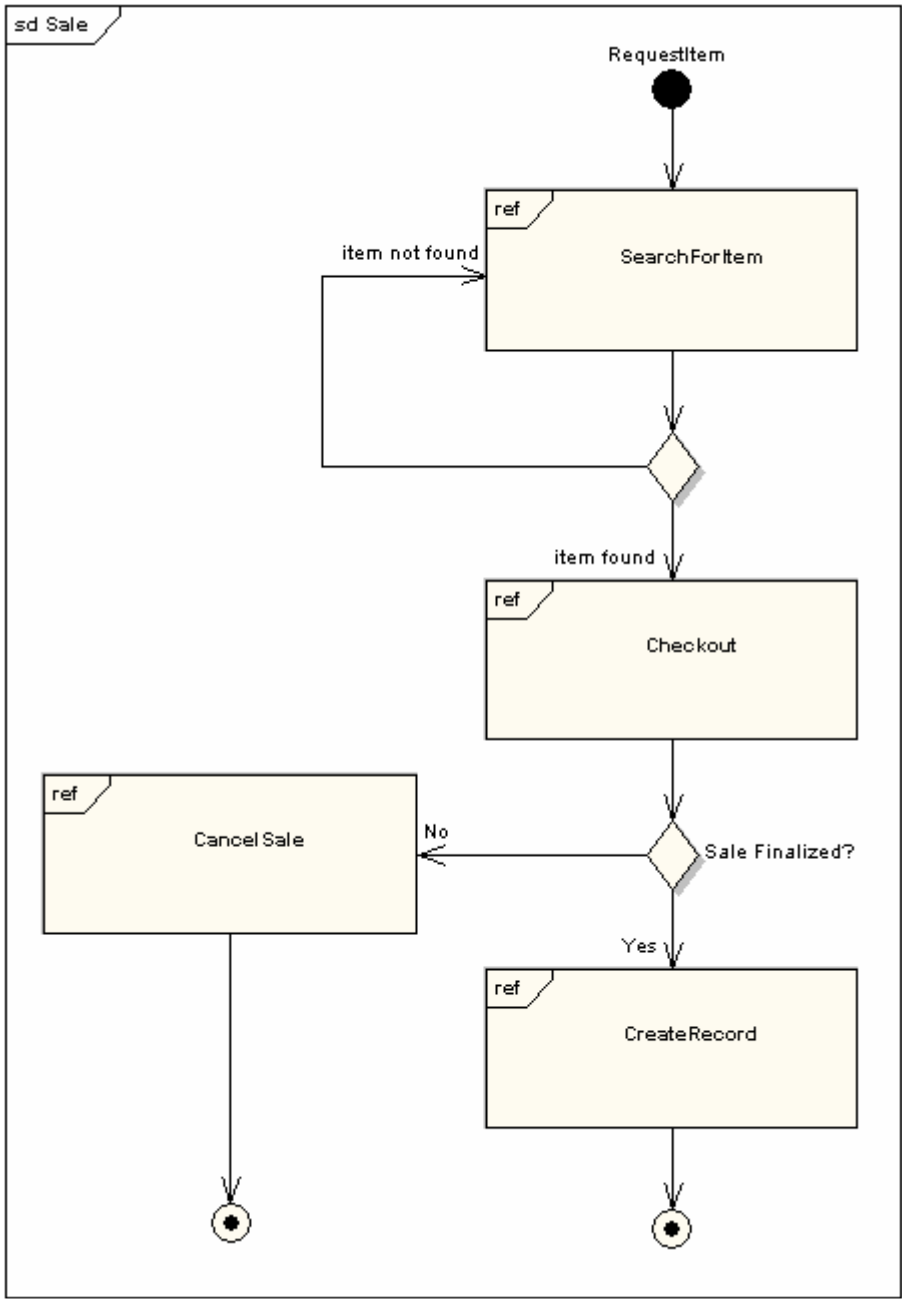
Diagram pro představu spolupráce mezi dalšími interakčními diagramy pro ilustraci kontroly toku. Jsou variantou diagramů aktivit, tedy rozhodovací body, forky a bariery, počáteční a koncové stavy.



Obrázek 9. Interaction Occurrence



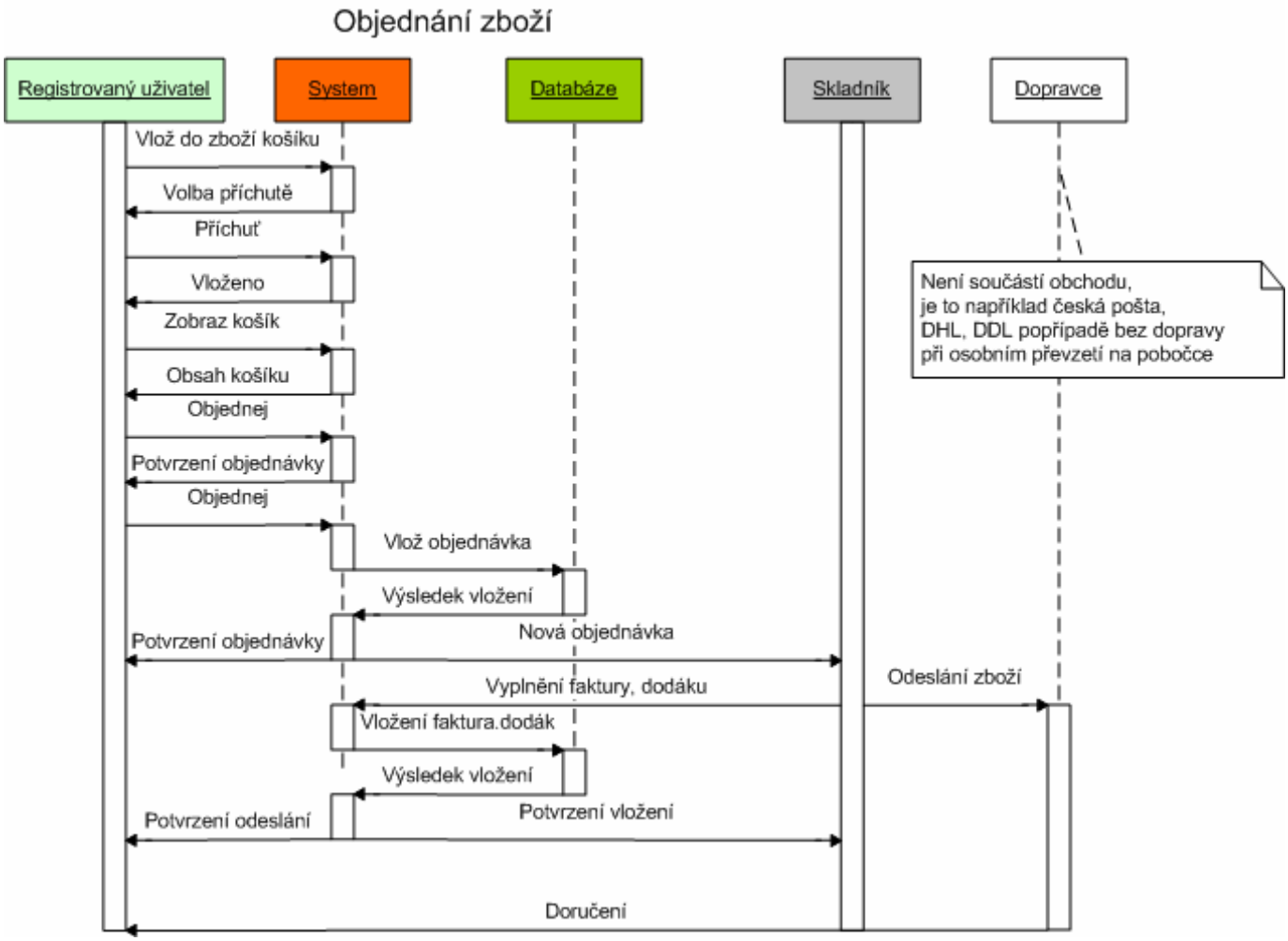
Obrázek 10. Interaction Element



Obrázek 11. Interaction Overview Diagram

3.2.3 Sekvenční diagramy (scénáře událostí, Sequence diagram)

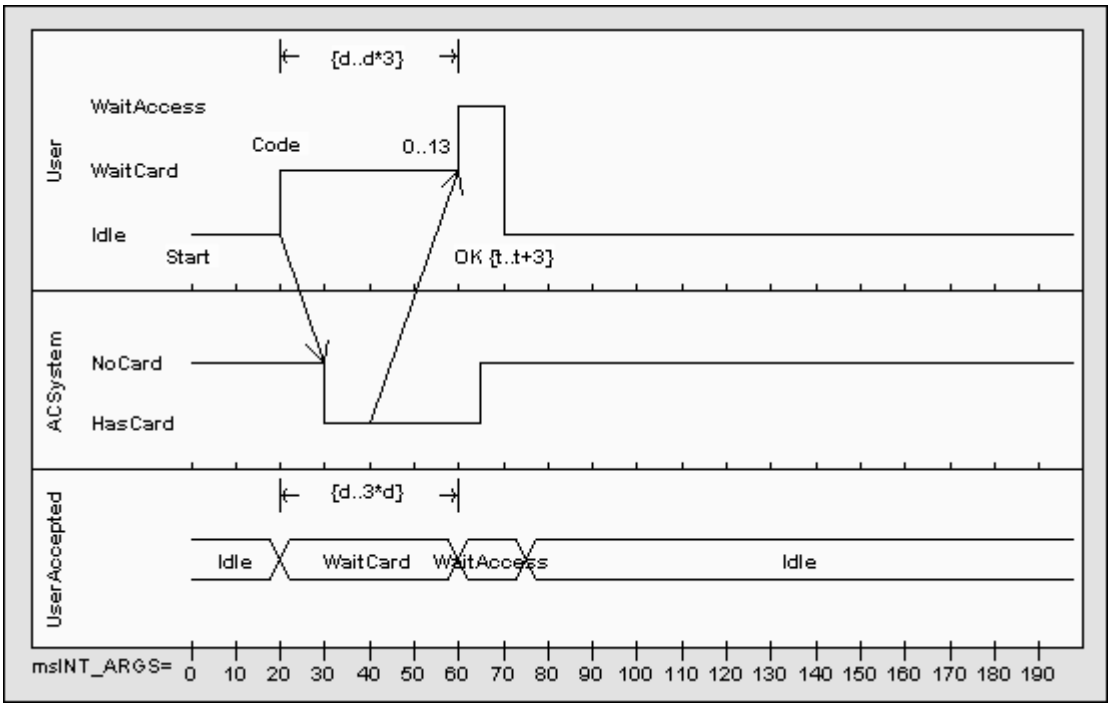
Sekvenční diagramy (scénáře událostí) popisují vzájemnou interakci mezi objekty systému v závislosti na čase. Sekvenční diagram má dvě dimenze, vertikální osa představuje čas a na horizontální ose jsou zobrazeny různé objekty. Čas plyne ze shora dolu. Měřítko na časové je vhodné zejména u real-time systémů.



Obrázek 12, Sekvenční diagram - Objednání zboží

3.2.4 UML Timing Diagram (UML 2.0)

Tento diagram definuje chování různých objektů v časovém měřítku. Poskytuje vizuální reprezentaci objektu jenž mění stav a reaguje v čase. Vhodné pro hardwarově řízené či vestavěné softwarové komponenty. Příkladem lze uvézt řadič, ale i časově řízený byznys proces.



Obrázek 13. Timing diagram

3.3 Strukturní diagramy

3.3.1 Datový model

Datový model má za cíl navrhnout kvalitní datovou strukturu pro konkrétní aplikaci a databázový systém, do nějž bude aplikace ukládat data. Při datovém modelování obvykle vytváříme nejprve právě *konceptuální datový model*. Konceptuální datový model představuje zobecnění oproti konkrétní implementaci datové struktury v relační, objektové, případně nativní XML databázi. Díky zobecnění získáme nezávislost modelu na konkrétním databázovém systému, a je možno tento model kdykoliv převést do konkrétního implementačního prostředí (na logický model).

Poznámka

V této části bych rád upozornil na nesrovnalosti v různých literaturách. Některé literatury uvádí pohled na datové modely jako konceptuální, logický a fyzický, jiné mají škálování jen dvou úrovňové a to logický a fyzický.

Konceptuální model je nezávislý na výsledném implementačním prostředí a nezabývá se cizími klíči, není ovlivněn budoucími prostředky řešení (ER model, Class diagram)

Logický model se vztahuje ke konkrétnímu datovému modelu a používá jeho konstrukční dotazovací a manipulační prostředky (síťový model, XML model, Objektový model, Relační model)

Fyzický model je vlastní uložení dat a pro programátora i analytika je transparentní (sekvenční soubor, hromada, indexy, cluster, B-stromy)

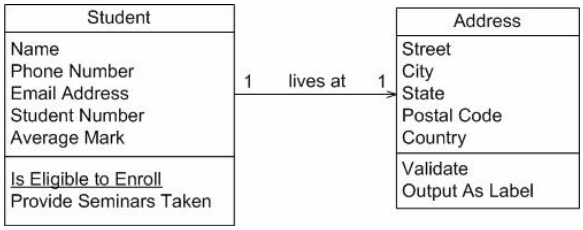
Class diagram – model tříd

Tento model (class diagram) popisuje statickou strukturu systému, znázorňuje datový model systému od konceptuální úrovně. Zachycuje analýzu dat. Rozlišuje reálné objekty tzv. entity, atributy entit ,jejich vztahy (kardinality), dále pak vyjádření vztahů tříd jako nadtyp podtyp, abstraktní typ, generalizace – tedy tzv. *instanční vztahy*.

Jedna entita nebo také třída má název, proměnné a metody (C++, Java)

Vztahy z hlediska kardinalit

0..1	žádná nebo jedna
1	přesně jedna
0..* or *	nula až mnoho
1..*	jeden až mnoho



Obrázek 14. Model tříd vztah 1-1

Obrázek 15. Model tříd vztah 0..*-1

Instanční vztahy

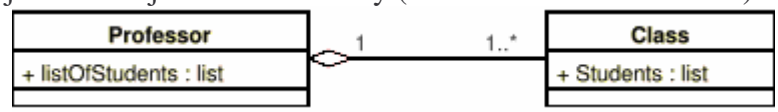
1. Spojení – obyčejný vztah mezi třídami

Obrázek 16. Class diagram - spojení

2. Asociace – vztah mezi třídami jenž je pojmenovaný, obousměrné či jednosměrné

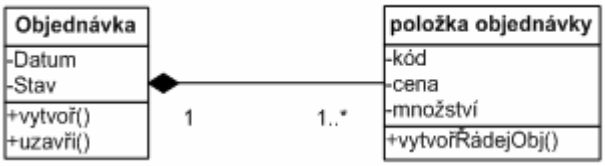
Obrázek 17. Class diagram - asociace

3. Agregace – jedna třída je částí druhé třídy (Visio tento vztah nenabízí)



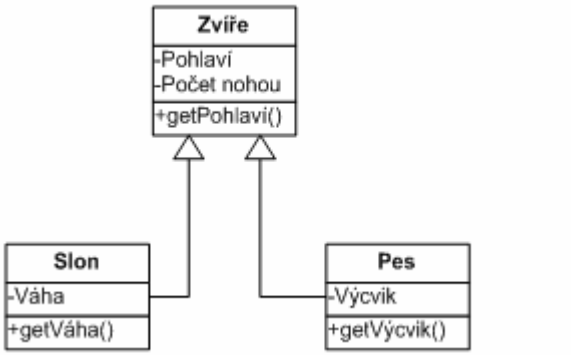
Obrázek 18. Class diagram -agregace

4. Kompozice – je speciální případ agregace – slabý entitní typ, tedy neexistuje bez hlavní třídy

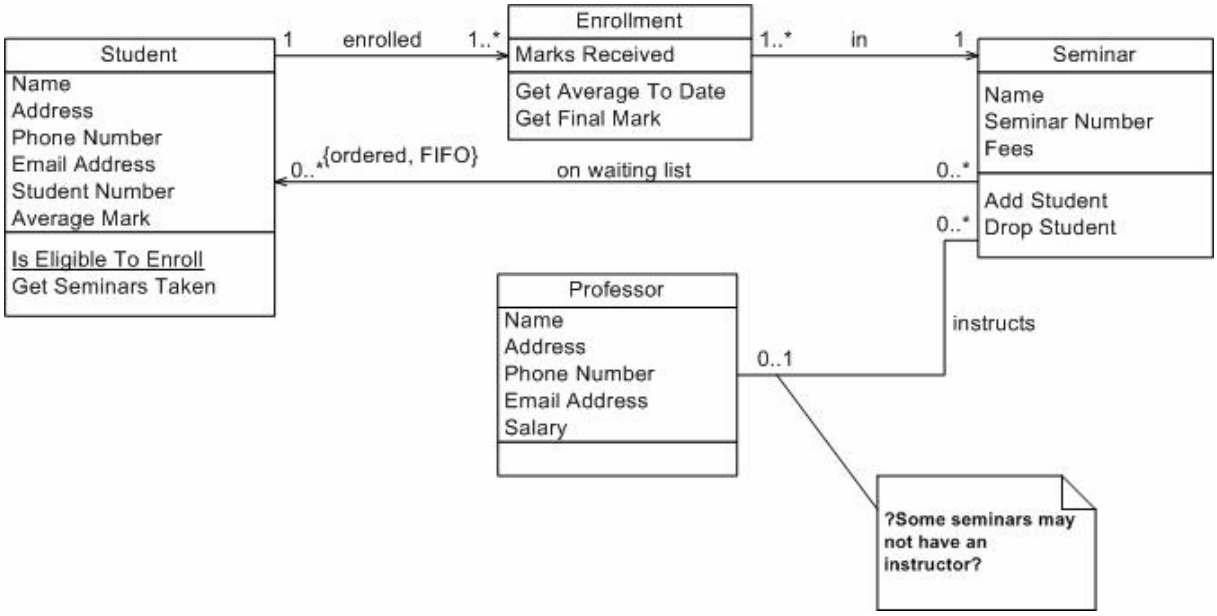


Obrázek 19. Class diagram - kompozice

5. Generalizace (ISA) – specializace – pro objektový návrh, dědění



Obrázek 20. Class diagram - generalizace



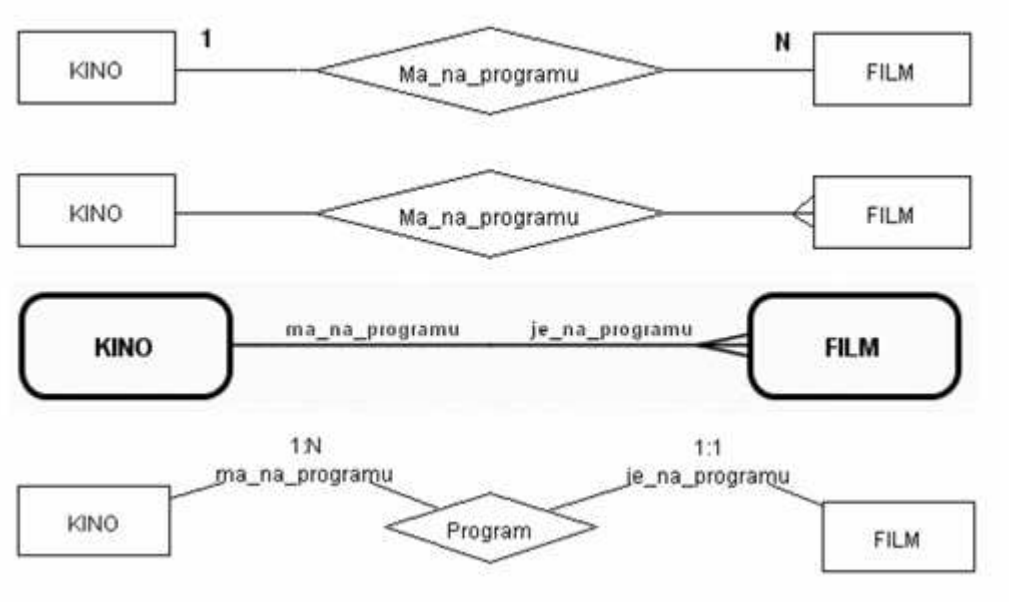
Obrázek 21. Příkla modelu tříd

ER model (není součástí UML)

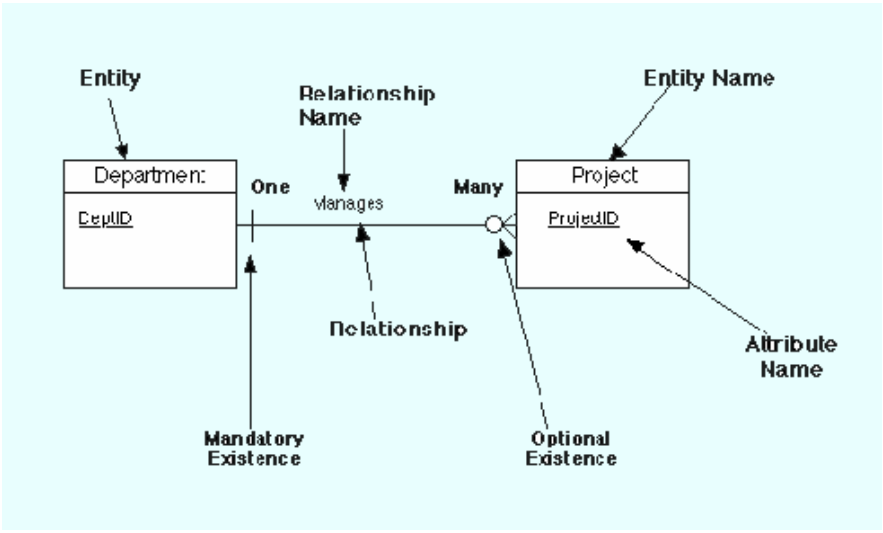
Poměrně známým datovým modelem je **ER**(*Entity relationship*)model, který má opět několik verzí a to binární a obecně n-ární. Tento model je vhodnější pro modelování z pohledu návrháře databází, jelikož lépe vystihuje návrh a snadněji se problém dekomponuje.
ER model je modelem konceptuálním, tedy umožňuje popsat data nezávisle na jejich uložení, popsat formálně realitu. Z tohoto modelu lze snadno přejít na logický model.

Zachycuje entity, vztahy a atributy. Entita je objekt reálného světa, jako osoba, kniha a lze snadno odlišit od jiných entit. Je více než patrné, že entity mají různé atributy, ty nejvýznamnější jsou ty, které je přímo identifikují. Mezi entitami jistě existují vztahy jako vztah mezi zmíněnou osobou a knihou může být například zapůjčena či přečtena. Lehkým analyzováním problému dojdeme k závěru, že čtenář si může přečíst několik knih, ale kniha může být v jistém časovém okamžiku propůjčena jen jedné osobě, tomuto vztahu se říká kardinalita vztahu(0-1, 0-N, 1-1, 1-N, N-M).

Při návrhu model vycházíme ze známých dekompozic lze jmenovat metodu shora dolů, zdola nahoru, zevnitř ven a slučování již existujících dat.



Obrázek 22. Typy ER notací



Obrázek 23. Příklad ER modelu - Crow's Feet notace

Obrázek 24. Příklad ER modelu – Chen notace

Popis datového modelu

Pro jednoznační vymezení datového modelu, lze doplnit některé složité části slovně, případně popsat domény datových typů u konkrétních atributů.

primární klíč je označen (*)

Produkt

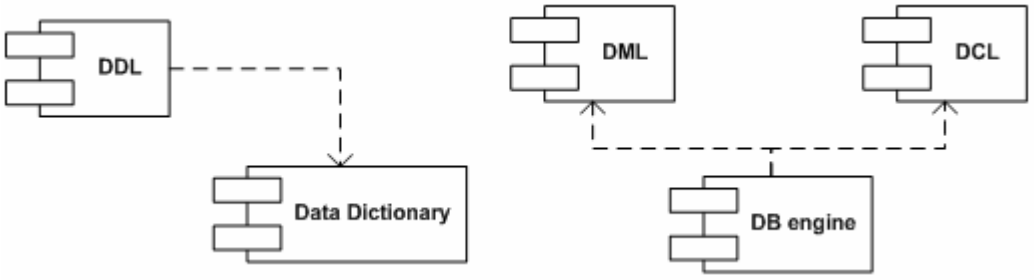
Obsahuje informace o produktu, který je či byl v nabídce obchodu
Produkt je zadán skladníkem
Produkt je na žádné až více Fakturách
Produkt je předmětem žádné až více Reklamací
Produkt je v žádné nebo více Akcích
Produkt je nebo není Novinka
Produkt je v právě Kategorii

Atribut	Popis
ID_produkt (*)	kód produktu
Stav	zda je produkt skladem, na cestě nebo dočasně nedostupný
Nazev	název produktu
Cena	cena produktu
Popis	popis produktu
Cena_bez	cena bez daně, nebo cena doporučená výrobcem (dle zadavatele)
Vyrobce	označení výrobce produktu

3.3.2 Diagram komponent (Component diagram)

Tento diagram se týká pouze počítačových systémů. Vývoj moderního programového vybavení probíhá po jednotlivých komponentách. Jsou předurčeny pro větší projekty, kde je každá komponenta vyvíjena zvlášť, či někým jiným. Komponenta může obsahovat mnoho propojených tříd a lze na ni pohlížet jako na ucelenou část reálného světa.

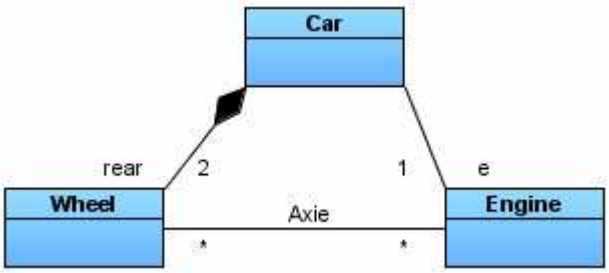
Obrázek 25. Komponenta



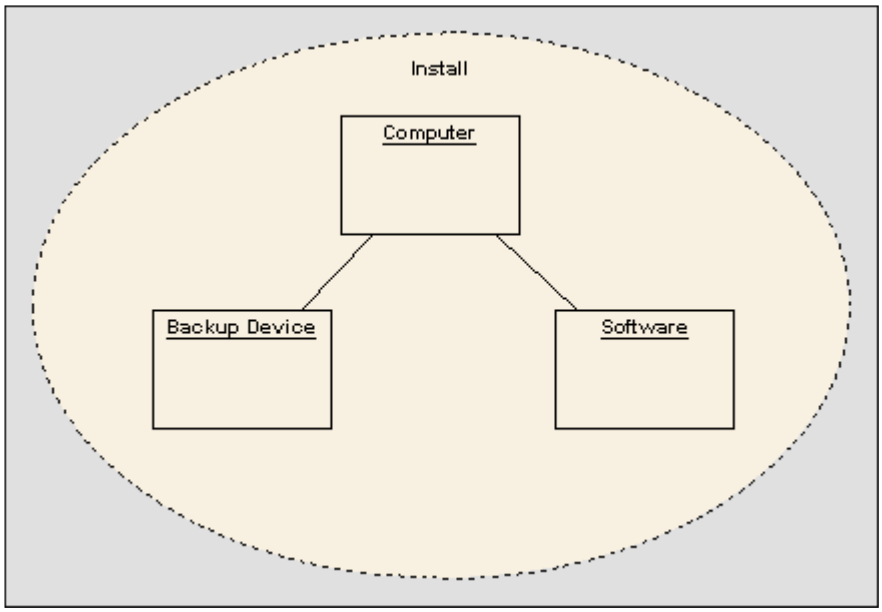
Obrázek 26. Diagram komponent - DB

3.3.3 Composite structure diagram

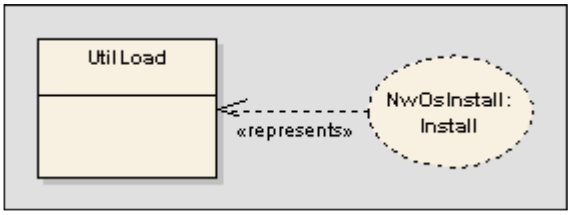
Tento diagram odráží vnitřní spolupráci tříd, rozhraní a komponent k popisu funkcionality. Je podobný diagramům tříd. Rozdíl je, že modeluji specifické použití struktury. Jsou hlavně pro vyjádření run-time systémů, což nelze vždy popsat statickými diagramy.



Obrázek 27. Composite structure diagram - auto



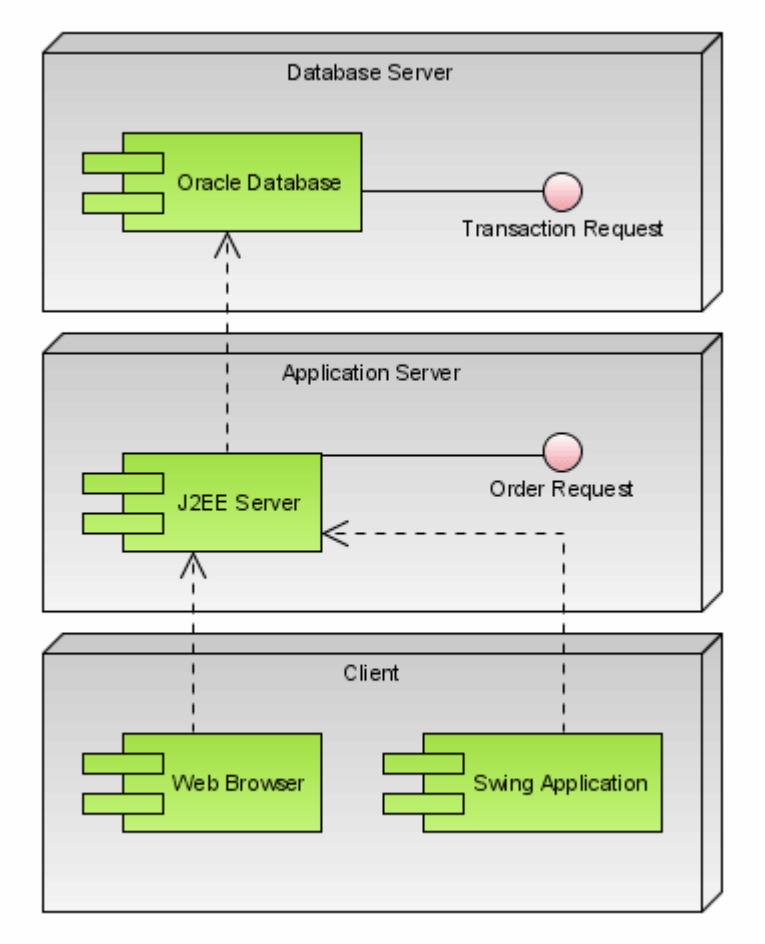
Obrázek 28.Composite structure diagram – instalace



Obrázek 29. Aplikace předchozí instalace

3.3.4 Diagram nasazení (Deployment diagram)

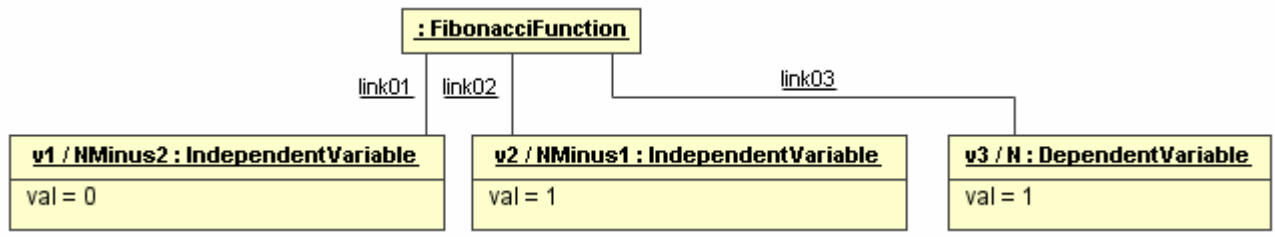
Tento diagram ukazuje fyzickou architekturu počítačového systému. Ukazuje propojení počítačů a zařízení a také software který je na daných zařízení nainstalován.



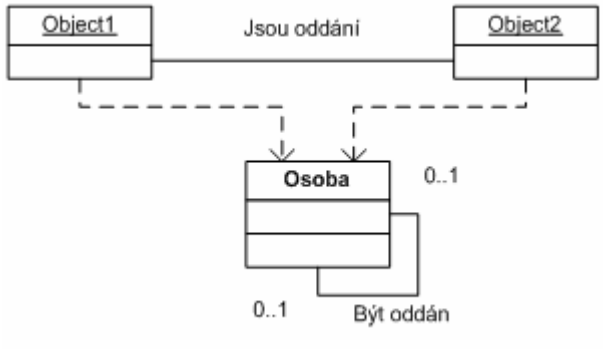
Obrázek 30. Diagram nasazení – 3 vrstvy

3.3.5 Diagram Objektů (Object diagram)

Tento diagram bývá často zaměňován s diagramem tříd. Jeho účelem je vyjádření vztahu mezi instancemi dané třídy. Ukazuje tedy chování systém z pohledu instancí.



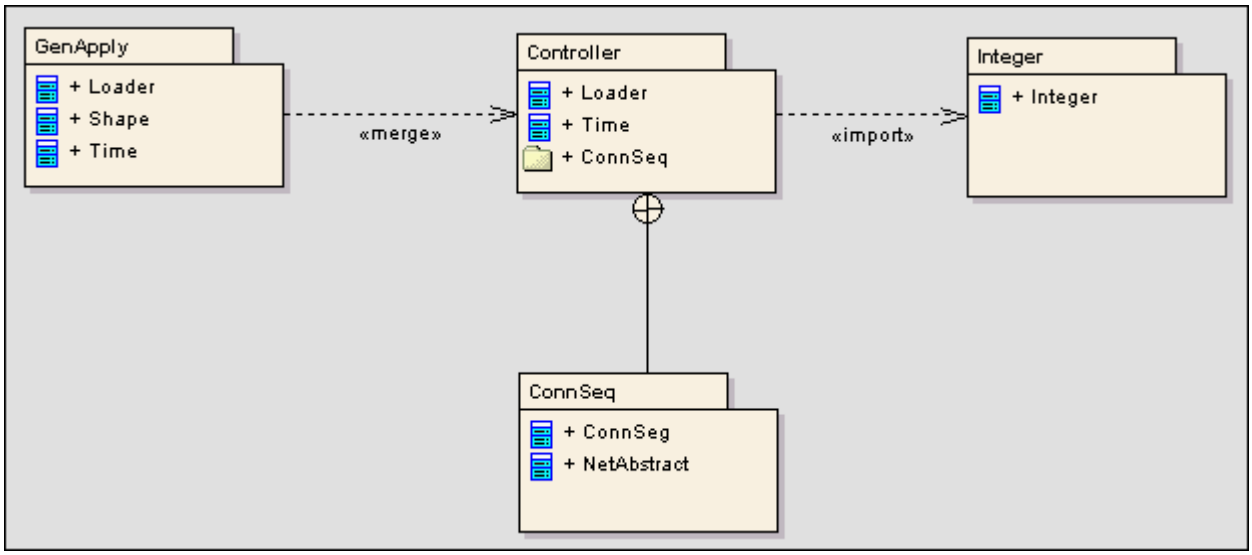
Obrázek 31. Diagram objektů - fibonači



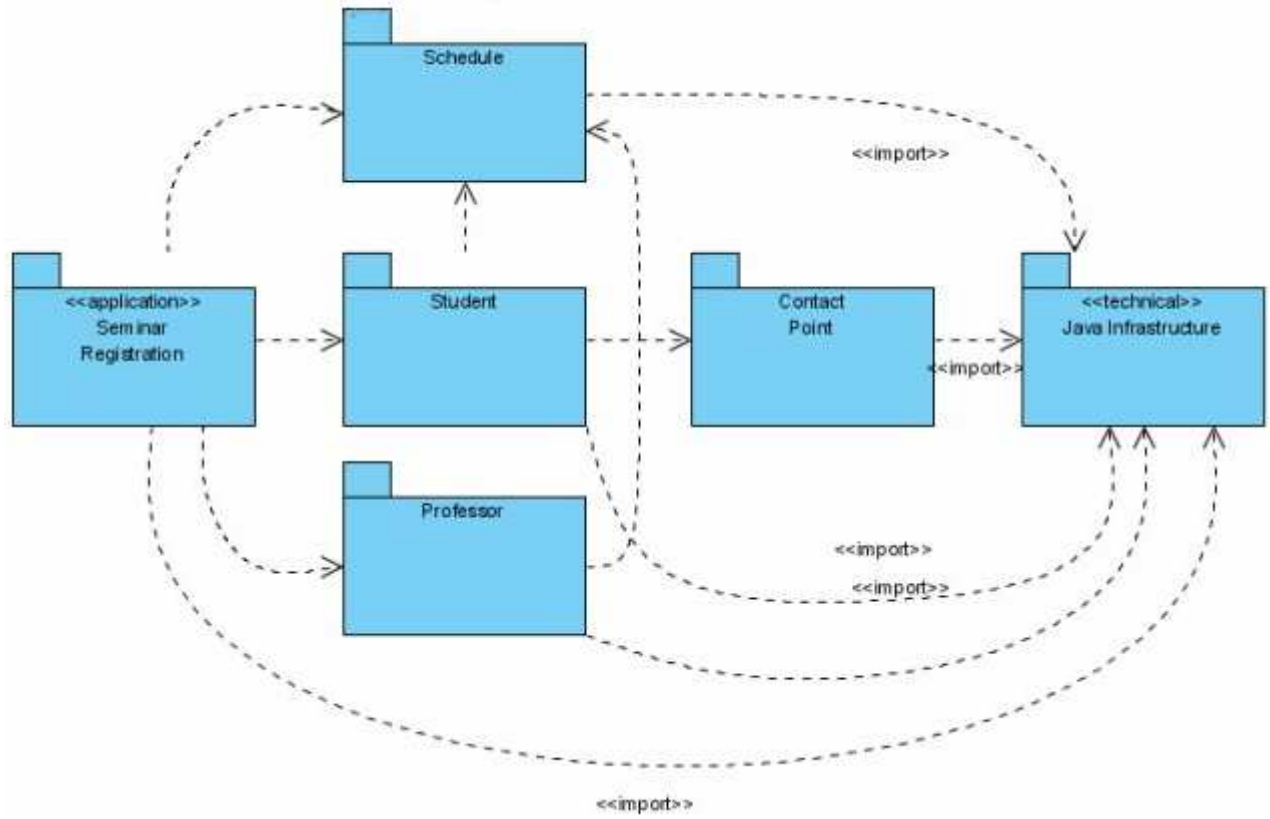
Obrázek 32. Diagram Objektů - manželství

3.3.6 Package diagram

Ukazuje jak je systém rozdělen do logických balíčků, ukazujíce závislosti mezi nimi. Balíčky jsou organizovány k maximalizaci vnitřních souvislostí a zároveň k minimalizaci externího spojení mezi balíčky. Každý baliček může být individuální nebo skupinový, právě pro druhý z nich je zde tento diagram. Balíček obsahuje své atributy.



Obrázek 33. Package diagram



Obrázek 34. Package diagram - zápis

4 Object Constraint Language (OCL)

Tento jazyk popisuje integritní omezení modelů jazyka UML a je standardem jazyka. V UML je především proto, že popisuje omezení, která nelze z diagramů vyčíst. Tento nedostatek by mohl být odstraněn, doplněním textu ke každému diagramu. Ovšem v praxi se čtou jen texty krátké a tedy je třeba omezení popsat jednodušeji než zdlouhavým textem. Tento jazyk se skládá ze čtyř částí:

- Klíčová slova (IF, ELSE, OR, NOT..)
- Operace (množinové, aritmeticko-logické)
- Vlastnosti pro specifikování kontextu (kontext –třída, vlastnost-atribut)
- Kontext pro definici situace, kdy platí popis omezení

Navigace je následující

```
object.selector
```
- položka objektu, atribut, jméno role

```
object.selector[kvalifikátor]
```
- objekt specifikovaný kvalifikátorem

```
sada –> select(logická podmínka)
```
- objekty vybrané z dané sady dle logické podmínky

Představme si problém jak vyjádřit integritní omezení, že se speciální platové ohodnocení týká jen zaměstnanců starších 50 let. Tedy vybereme jen tyto instance.

```
self.employee.select(p:Person | p.Age > 50)
```

Schopností jazyka OCL je i výraz, který má vyjádřit podmínku, že všechny instance osoby mají rozdílné jméno.

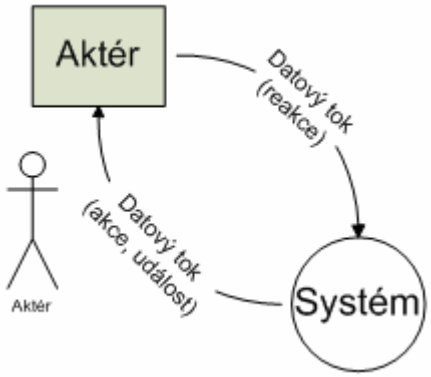
```
Person.allInstances?forAll(p1, p2
       | p1 <> p2 implies p1.name <> p2.name)
```

5 Další diagramy a konstrukty vhodné pro modelování

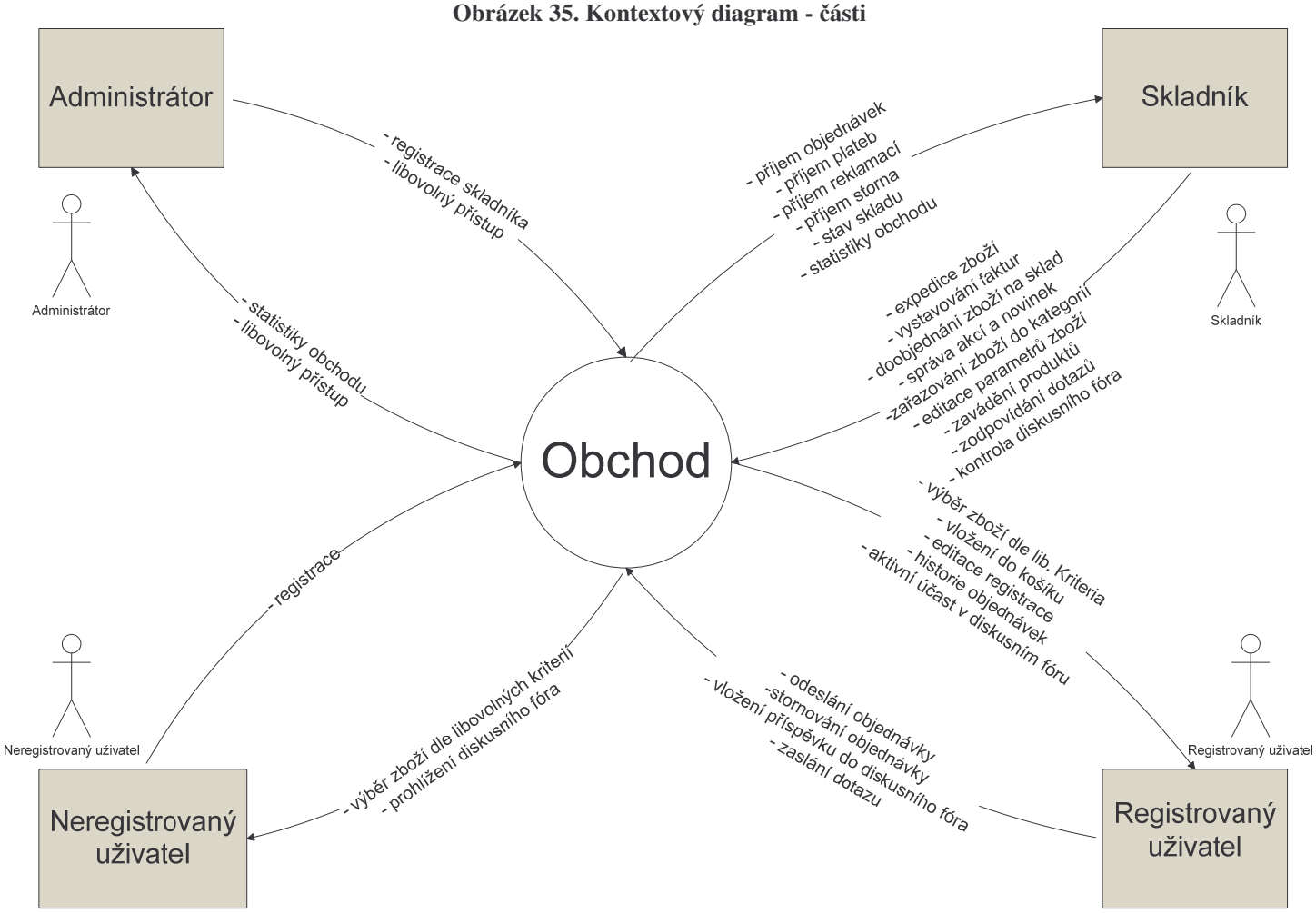
* (nejsou součástí UML)

5.1 Kontextový diagram (Context diagram)

Tento diagram slouží pro evidenci aktérů a datových toků v systému. Je vhodný především pro prezentaci klientovy, protože je velice jednoduchý a přehledný.



Obrázek 35. Kontextový diagram - části

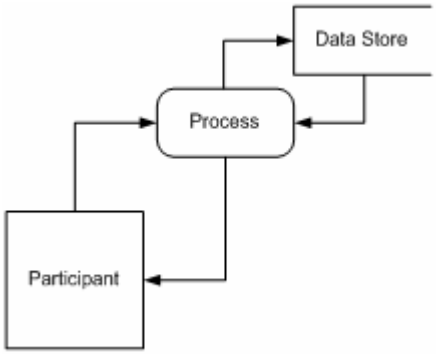


Obrázek 36. Kontextový diagram – obchod

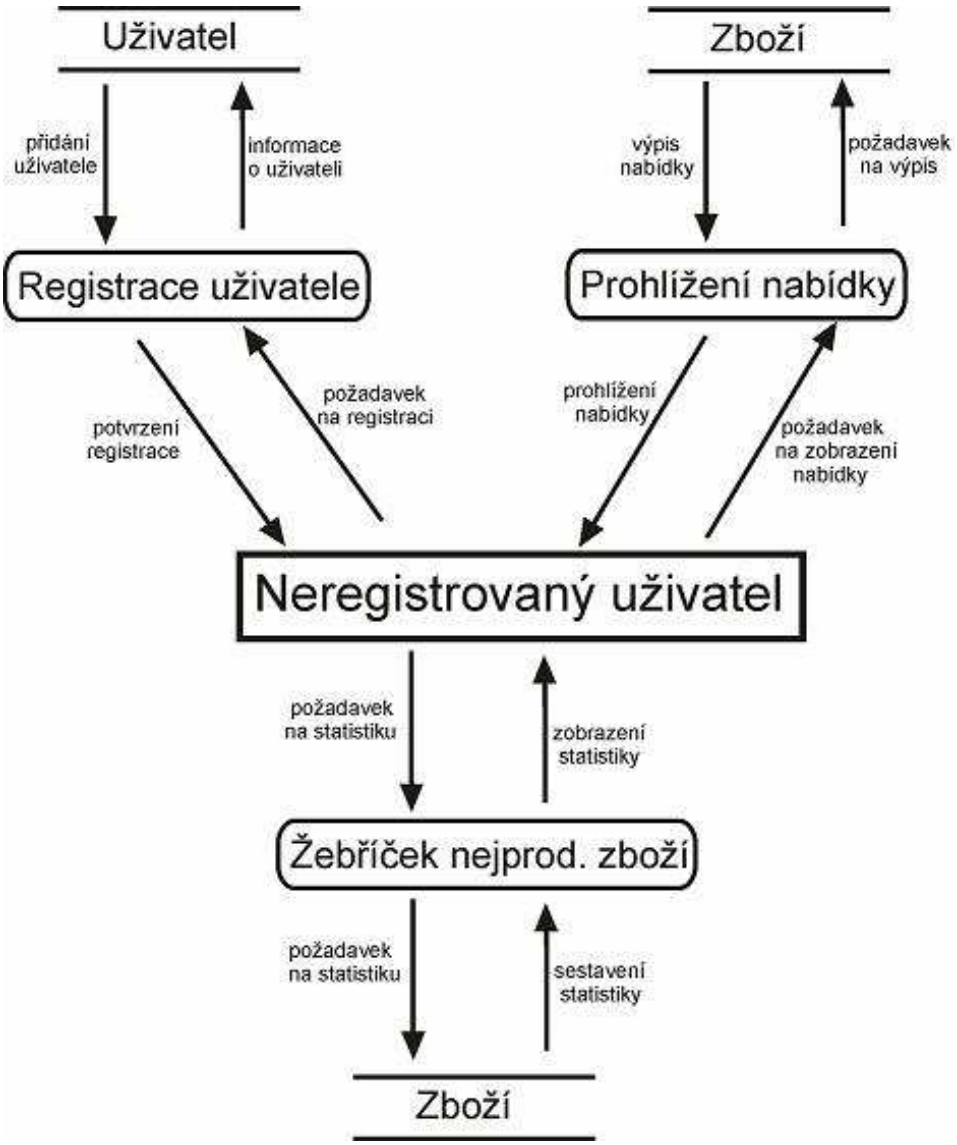
5.2 Diagram datových toků (Data flow diagram)

Tento diagram zachycuje vazby funkcí a toků dat. Jeho hlavními komponentami jsou:

- Funkce (procesy akce)
- Toky dat
- Datové paměti
- Aktéři



Obrázek 37. Diagram datových toků – komponenty



Obrázek 38. Diagram datových toků

5.3 Mini-specifikace

Někdy místo diagramu použijeme mini-specifikaci, ta je v některých případech lepší. Specifikuje se slovně postup kroků, který musíme vykonat pro dosažením některého cíle. Vycházet lze z diagramu USE-CASE. Při specifikaci lze použít klíčová slova jako IF, ELSE, LOOP. Velmi blízko mini-specifikaci je *test case*, používaný při testování software, nebo také jednomu bodu v *příručce*, navíc aby tomu nebylo málo podobným je i jeden *akceptační test*.

I. Mini-specifikace - objednávka

1. Zákazník se zaloguje
2. Vybere zboží
3. Objedná
4. Systém potvrdí platbu

II. Příručka - Odeslání objednávky

Předpokládaje že již máte vybrané všechno požadované zboží v nákupním košíku, můžete ho objednat.

1. Pokud nejste na stránce s výpisem obsahu košíku, klikněte na volbu Košík v Moje menu.
2. Systém zobrazí stránku s informace o obsahu košíku.
3. Klikněte na tlačítko Objednat.
4. Systém zobrazí formulář potvrzení objednávky, která slouží k uvedení doplňujících informací o platbě a vaší kontrole dodacích a fakturačních údajů.
5. Zvolte způsob dopravy a popřípadě doplňující informace.
6. Po samotném potvrzení objednávky systém vyhodnotí objednávku jako kompletní a pošle ji ke zpracování skladníkovi.

III. Akceptační test - Přihlášení uživatele

Předpoklady

Počítač s nakonfigurovanou klientskou aplikací.

Postup

Spusťte aplikaci. Zadejte uživatelské jméno a heslo.

Možné reakce

OK – Spojení s databází navázáno a uživatelské jméno a heslo ověřeno.
Chyba – Spojení s databází nenavázáno.
Chyba – Neplatné jméno nebo heslo.

IV. Test Case – Registrace nový účet

INPUT:

1. Naviguj se do Hlavního okna
2. Klikni na Nová registrace
3. Zadej login a password a potvrď
4. Vyplň formulář a využij kopii a odešli
5. Potvrď

OUTPUT:

Jsi registrován a přihlášen jako uživatel MO1

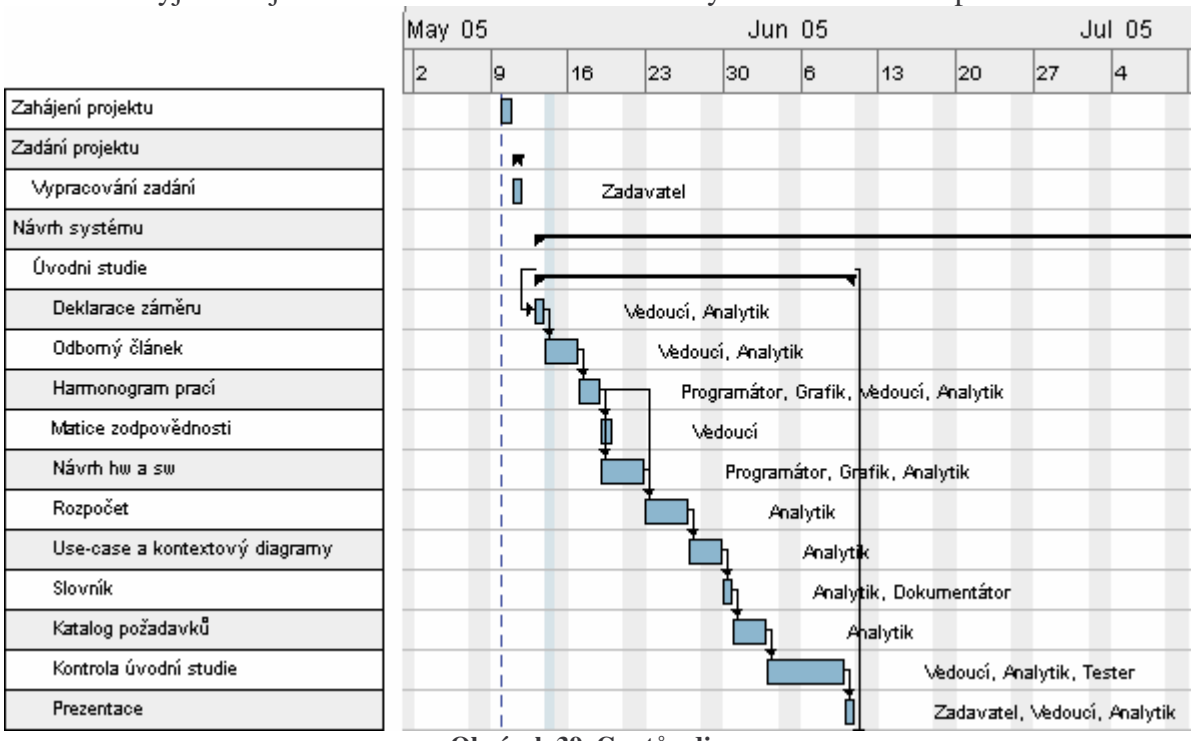
5.4 Hierarchie funkcí

Někdy je dobré popsat hierarchický strom funkcí, tedy jak jsou provázány a volány. Jedná se tedy o hierarchii jak ji známe. Tato hierarchie je dobrá především pro dohledání návaznosti dalších funkcí.

5.5 Gantův Diagram

Při vývoji aplikace je třeba vytvořit časový harmonogram projektu, ten slouží k časové orientaci, odhadu pro rozpočet, dále také k vytyčení *milestones*, tedy milníku projektu. V rámci milníků dochází ke zpětnému testování a prezentacím již hotových částí. Digram je poskytuje vymezení časové osy pro jednotlivé části projektu a tedy kdokoliv dozví v jaké fázi je daný projekt. Model je někdy označován jako model vodopád.

Základní vyjádření je formou seznamu aktivit s časovým rozsahem a zodpovědnou osobou.



Obrázek 39. Gantův diagram

Název aktivity	Začátek	Konec	Kdo
Analytická část	**11.6.2005**	**3.8.2005**	
Datový model	11.6.2005	21.6.2005	Analytik, Programátor
Popis datového modelu	21.6.2005	26.6.2005	Dokumentátor, Analytik
Diagram datových toků	26.6.2005	29.6.2005	Analytik
Stavový diagram	29.6.2005	3.7.2005	Analytik
Scénáře událostí	3.7.2005	11.7.2005	Tester, Analytik
Akceptační test	11.7.2005	18.7.2005	Tester, Analytik, Vedoucí
Uživatelská příručka I.	18.7.2005	25.7.2005	Dokumentátor
Kontrola analytické části	25.7.2005	2.8.2005	Tester, Dokum., Analytik, Vedoucí, Program.
Prezentace	2.8.2005	3.8.2005	Analytik, Vedoucí, Zadavatel
Architektura a GUI	**3.8.2005**	**31.8.2005**	
Návrh architrktury	3.8.2005	12.8.2005	Analytik, Programátor
Reprezentace dat	12.8.2005	19.8.2005	Dokumentátor, Analytik, Programátor
Návrh GUI	3.8.2005	22.8.2005	Grafik
Kontrola	22.8.2005	30.8.2005	Tester, Analytik, Vedoucí, Grafik, Program.
Prezentace	30.8.2005	31.8.2005	Analytik, Ved., Grafik, Zadavatel, Program.

5.6 Datový slovník (Data dictionary)

Součástí dokumentace bývá datový slovník především pro sjednocení používaných termínů a symbolů v rámci dokumentace. Je to především proto, že projekt může být výsledkem snažení několika vědních oborů a odvětví a v každém může termín vyjadřovat něco jiného. Dokumentaci budou rozumět díky slovníku všechny zúčastněné strany.

Jedním z takovýchto slovníků může být EBNF, rozšířená Bacus Naurova Forma, používaná při lexikální analýze v problematice jazyků a překladačů. Tato forma lze vyjádřit diagramem i regulárními výrazy.

```
* repetiční-symbol
- odečet-symbol
, konkatenace-symbol
| definice-separace-symbol
= definice-symbol
; terminátor-symbol

číslo          = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
číslice        = číslo { číslo }*
znak           = 'a' | 'b' .. | 'z' | 'A' … | 'Z'
identifikátor  = znak { číslo | znak }*
řetězec        = { číslo | znak }*
```

Datový slovník lze užít i jednodušeji

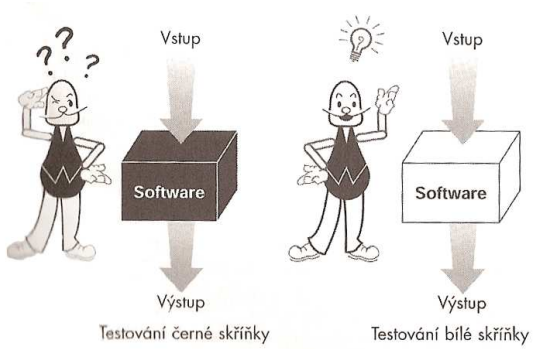
IČO = řetězec, který slouží jako Identifikační číslo, přidělují ho pracoviště správy registru ekonomických subjektů ČSÚ (44018967)
DIČ = řetězec, který slouží jako Daňové identifikační číslo (CZ1234567890)

6 Testování

Testování by mělo při vývoji software tvořit celkem 40% celkového úsilí, přičemž samotná implementace pouze 20%, zbývajících 40% je určeno pro analýzu. Základními pojmy v testování je *validace* a *verifikace,* nebo *black box* a *white box testování.*

6.1 Validace a verifikace

- Verifikace
  Proces s cílem potvrdit, že aplikace vyhovuje zadané specifikaci
  ~ zadaná specifikace
- Validace
  Proces kontroluje zda aplikace vyhovuje požadavkům uživatele
  ~ původní požadavky
  Validace bývá ověřována v průběhu vývoje při komunikaci s potenciálním zákazníkem.



Obrázek 40, White box a Black box

6.2 Black Box

Při black box testování tester ví, co má software dělat

- Pokud aplikaci něco předloží, ví co dostane
- Neví ale proč a jak se k tomu došlo
- Základem je pozorování

Výhodou black box testování je, že testy může provádět téměř kdokoliv, takovéto testování může provádět například externí osoba, která má o aplikaci zájem, a je ji poskytnuta již první testovací verze aplikace. Výhod to má hned několik, osoba může směrovat správným směrem, najde chyby aplikace o kterých se neví, a navíc jelikož to bývá cílová osoba, pro kterou bude aplikace určena, již vidí jak aplikace bude vypadat a může si postupně zvykat na ovládaní a způsob práce. Tímto krokem (poskytnutí první verze zákazníkovi) se vyvarujeme nechtěného efektu, že nám při předání zákazník odmítne aplikaci převzít. Navíc zdarma otestuje aplikaci, a upozorní na případná vylepšení. Problémem může být ovšem nerozhodný zákazník, který stále mění svá rozhodnutí, v takovém případě je lepší zvážit vynaložené úsilí, případně si vyžádat postupné finanční zálohy.

Black box testy jsou součástí standardních testů, které jsou mapovány v podobě Test Case.

6.3 White Box

Při White Box tester ví, co má software dělat, dokonce i jak to dělá

- Tester má přístup ke zdrojovému kódu a dokumentaci
- Dle kódu ví na co se zaměřit a co je kritické

Pro white box testování je velmi vhodná UML dokumentace, jelikož dle digramu lze vytvořit mnoho testů. Častým případem vyřešení takto nalezené chyby není oprava systému, ale oprava dokumentace. Dalším příkladem white box testů je procházení kódu a testování kritických míst, což můžou být například proměnné, které čtou svoji hodnotu z URL a tedy mohou být napadeny hackerem. To může mít za následek, že se hacker dostane k údajům cizích osob a může je snadno zneužít, případně zničit celou aplikaci. Jiným příkladem je možnost přetečení hodnot, nejčastěji bitových masek.

White box testy jsou součástí standardních testů, které jsou mapovány v podobě Test Case.

7 Reference

[1] UML srozumitelně, Hana Kanisová; Miroslav Müller, Computer Press, 2006

[2] Wikipedie, otevřená encyklopedie, http://en.wikipedia.org/wiki/Wiki

[3] Dokumentace UML diagramů
www.sparxsystems.com.au/EAUserGuide/index.html?interactionoverviewdiagram.htm

[4] Dokumentace UML diagramů
www.visual-paradigm.com/VPGallery/diagrams/CompositeStructureDiagram.html

[5] Změny analytické části návrhu SW projektu po vyhodnocení implementace, Tomáš Černý, Bakalářská práce, ČVUT – FEL , 2007

[6] Myslíme v jazyku UML, Joseph Schmuller, Grada, 2001

[7] Softwarové Inženýrství, 36SI, Doc.Ing. Karel Richta, CSc.., ČVUT – FEL , 2005

[8] Databázové Systémy, X36DBS, Ing. Michal Valenta, Ph.D., ČVUT – FEL , 2005

Případy užití IS pro praktické lékaře

Dokument obsahuje podrobné specifikace případů užití informačního systému pro praktické lékaře. Každý případ užití je popsán pomocí hlavního toku událostí. Z důvodu přehlednosti jsou vedlejší toky událostí specifikovány pouze u vybraných případů užití.

Obsah

Slovníček pojmů

.

Termín	Popis
Anamnéza	Souhrn informací z určité oblasti usnadňující stanovení diagnózy. Je součástí vyšetřování pacienta lékařem. (rodinná, sociální, alergická, pracovní, atd.)
Charakter dávky	Určuje, zda se jedná o dávku řádnou (obsahuje pouze původní doklady), nebo opravnou (obsahuje opravené doklady dříve odmítnuté).
Dávka	Pomocná jednotka pro vyúčtování a předávání dokladů pojišťovně.
Dekurz	Záznamy jednotlivých vyšetření, nálezů, neschopností, vystavených receptů a dalších informací do karty pacienta.
Diagnóza	Rozeznání nemoci a její pojmenování. Číselné kódy jednotlivých diagnóz lze najít v mezinárodní klasifikaci nemocí (MKN).
Číselník	viz. Globální číselník
Globální číselník	Seznam jednotlivých položek dodávaných pojišťovnou, pravidelně aktualizovaný. Typem položky obsažené v číselníku je dán i typ číselníku (Léky, Diagnózy, apod.) Přesný popis formátu souboru je dán datovým rozhraním pojišťovny.
IČP	Identifikační číslo zdravotnického zařízení.
Kapitace	Paušální platba za registrované pacienty, ve které jsou zahrnuté některé standardní výkony prováděné pro danou věkovou skupinu pacientů – již se neuvádějí ve vyúčtování výkonů za registrované pacienty.
Kapitační koeficient	Koeficient, kterým se násobí základní hodnota kapitace. Koeficient je závislý na věku pacienta. Velikost kapitačního koeficientu pro jednotlivé věkové skupiny vydává pojišťovna v podobě číselníku.
Katalog	Seznam položek, které lékař často používá. Jednotlivé položky katalogu musí lékař do systému zadávat ručně. Datové rozhraní katalogu není nijak upřesněno. Katalogy, které lékař může v systému využít: Spolupracující lékaři, Adresy, Magistraliter.
Lékařská zpráva	Dokument, kterým lékař předává jinému zdravotnickému zařízení informaci o výsledcích provedeného vyšetření nebo ošetření.
Lokální číselník	Výběr položek z globálního číselníku, které lékař často používá. Zjednodušuje lékaři vyhledávání často používaných položek.
LSPP	Lékařská služba první pomoci.
Magistraliter	Předpis pro přípravu léčiva v lékárně. Vhodné pro přípravky, které nejsou vyráběny farmaceutickými společnostmi.
Mezinárodní klasifikace nemocí (MKN)	Mezinárodně platný seznam nemocí, úrazů a příčin smrti s přiřazenými čísly.
Recept	Formulář, na který lékař pacientovi předepisuje léky nebo magistraliter. Na základě receptu vydá lék pacientovi lékárna. Na jednom receptu mohou být uvedeny maximálně dva druhy

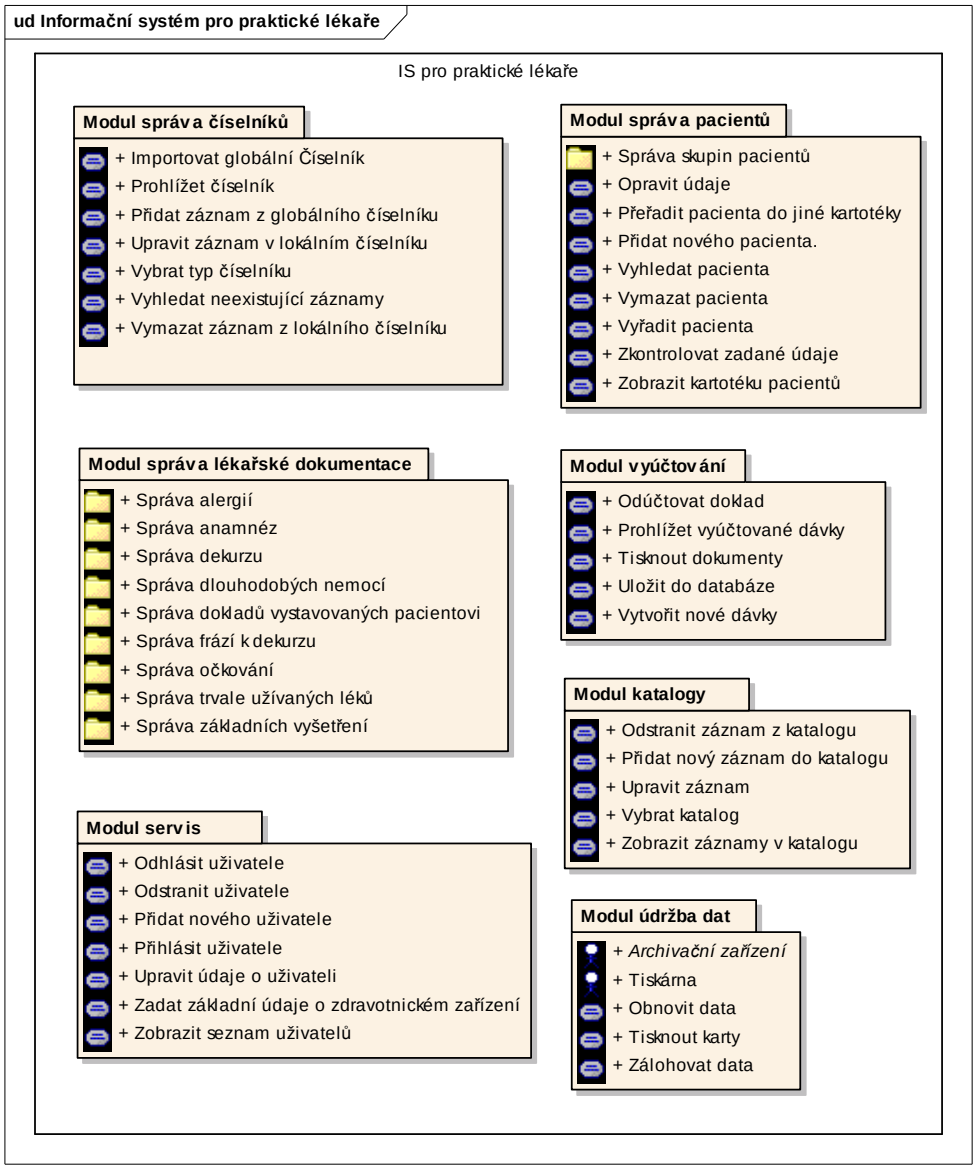
	léků.
Sedimentace	Vyšetření krve prováděné v laboratoří nebo přímo v ordinaci.
Typ dávky	Určuje zda dávka obsahuje pouze doklady jednoho druhu (jednoduchá dávka) nebo obsahuje-li doklady více druhů (smíšená dávka).
Výměnný list	Dokument, kterým lékař žádá jiné zdravotnické zařízení o provedení vyšetření nebo ošetření pacienta.
VZP	Všeobecná zdravotní pojišťovna
ZP, ZUM	Zdravotnické prostředky. Pomůcky nebo materiál, který lékař poskytne pacientovi. Vyúčtování takovýchto prostředků se provádí spolu s vyúčtování zdravotních výkonů pojišťovně.
ZUM	viz. ZP

Případy užití

1. IS pro praktické lékaře

Hlavní balíček obsahující všechny případy užití informačního systému.

Informační systém pro praktické lékaře



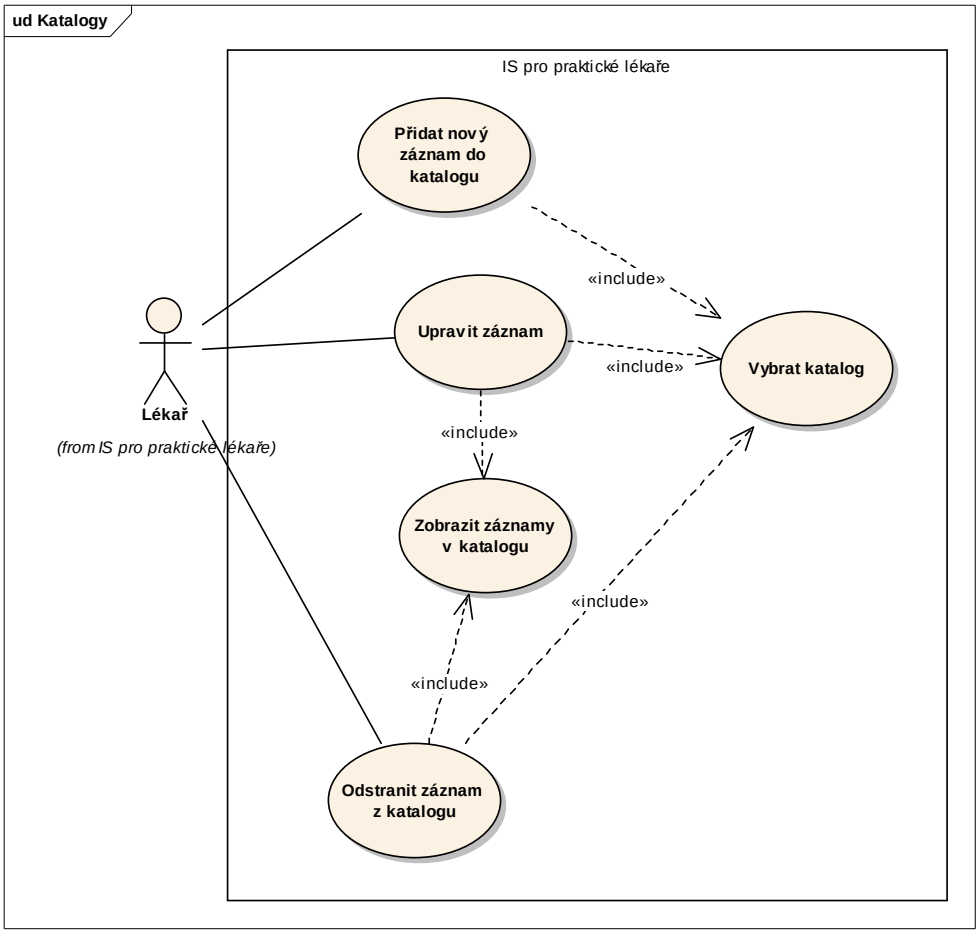
Obrázek:: Informační systém pro praktické lékaře

2. Modul katalogy

Případy užití v tomto balíčku jsou shodné pro všechny katalogy v systému a slouží pro rychlejší zadávání opakujících se údajů. V systému budou evidovány katalogy: Adresy, Spolupracující lékaři, Magistraliter. Katalogy mohou být prázdné, veškeré záznamy v katalogu si musí uživatel sám nadefinovat.
Záznam v katalogu bude obsahovat následující položky:

1. Magistraliter - kód, název, doplatek, počet dávek, popis.
2. Spolupracující lékaři - příjmení, jméno, odbornost, IČZ.
3 . Adresy - jméno, ulice, číslo popisné, obec, PSČ.

Katalogy



Obrázek:: Katalogy

Odstranit záznam z katalogu

Odstraní vybraný záznam z katalogu.

Tok událostí:

1.1.1 Basic Path

Odstranění záznamu z katalogu

1. Případ užití začíná, když chce lékař odstranit některý ze záznamů v katalogu.
2. INCLUDE (Vybrat katalog).

3. Systém požádá lékaře o výběr záznamu, který chce odstranit.
4. INCLUDE (Zobrazit záznamy v katalogu).
5. Systém odstraní vybraný záznam.

Tok událostí:

1.1.2 **Alternate**

Zrušení odstranění záznamu

1. Lékař může kdykoliv zrušit odstraňování záznamu stiskem tlačítka Storno.

Přidat nový záznam do katalogu

Zobrazí formulář umožňující zadat všechny položky záznamu daného katalogu. Přidá nový záznam do katalogu.

Tok událostí:

1.1.3 **Basic Path**

Přidání nového záznamu do katalogu

1. Případ užití začíná, když chce lékař zadat do katalogu nový záznam.
2. INCLUDE (Vybrat katalog).
3. Systém zobrazí formulář umožňující zadat jednotlivé položky záznamu, podle vybraného katalogu:
   Magistraliter - kód, název, doplatek, počet dávek, popis,
   Spolupracující lékaři - příjmení, jméno, odbornost, IČZ,
   Adresy - jméno, ulice, číslo popisné, obec, PSČ.
4. Lékař vyplní požadované údaje.
5. Systém přidá do katalogu nový záznam.

Tok událostí:

1.1.4 **Alternate**

Zrušení přidávání záznamu

1. Lékař může zadávání nového záznamu kdykoliv přerušit.

Upravit záznam

Umožňuje upravit jednotlivé položky u vybraného záznamu v katalogu. Seznam položek jednotlivých záznamů v katalogu je uveden v popisu tohoto balíčku.

Tok událostí:

1.1.5 **Basic Path**

Upravení záznamu v katalogu

1. Případ užití začíná, když chce lékař upravit některý ze záznamů v katalogu.
2. INCLUDE (Vybrat katalog)
3. Systém požádá lékaře o výběr záznamu z katalogu, který chce upravovat.
4. INCLUDE (Zobrazit položky katalogu).
5. Systém zobrazí formulář umožňující upravit veškeré položky u vybraného záznamu.
6. Lékař upraví požadované údaje.
7. Systém uloží do záznamu všechny změny provedené lékařem.

Tok událostí:

1.1.6 **Alternate**

Zrušení provedených úprav

1. Lékař může veškeré provedené změny odvolat stisknutím tlačítka Storno.

Vybrat katalog

Umožňuje lékaři vybrat katalog, se kterým chce pracovat.

Tok událostí:

1.1.7 **Basic Path**

Vybrání katalogu

1. Případ užití začíná, jestliže lékař potřebuje vybrat katalog, se kterým chce pracovat.
2. Systém zobrazí seznam evidovaných katalogů: Magistraliter, Spolupracující lékaři, Adresy.
3. Lékař jeden ze zobrazených katalogů vybere.

Zobrazit záznamy v katalogu

Zobrazí jednotlivé záznamy v katalogu. Umožňuje je seřadit podle vybrané položky. Dále umožňuje jeden ze zobrazených záznamů vybrat. Položky, které se budou u jednotlivých katalogů zobrazovat:

Magistraliter - kód, název,
Spolupracující lékaři - IČZ, příjmení, jméno,
Adresy - jméno, obec.

Tok událostí:

1.1.8 **Basic Path**

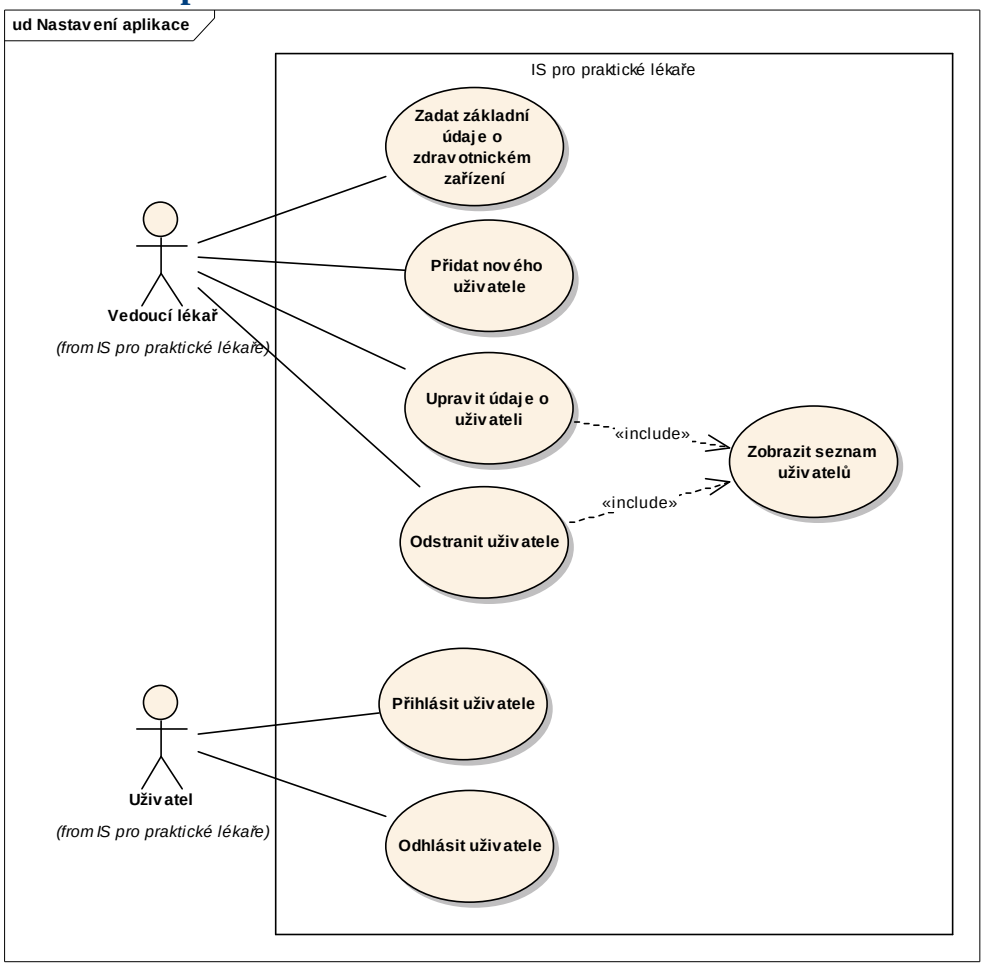
Zobrazení záznamů v katalogu

1. Případ užití začíná, jestliže se uživatel rozhodne zadat záznam z některého katalogu evidovaného v systému.
2. Systém zobrazí seznam všech záznamů ve zvoleném katalogu.
3. Uživatel jeden ze zobrazených záznamů vybere.

3. Modul servis

Balíček obsahuje případy užití související s nastavením aplikace a správou uživatelů.

Nastavení aplikace



Obrázek:: Nastavení aplikace

Odhlásit uživatele

Odhlásí uživatele ze systému.

Tok událostí:

1.1.9 Basic Path

Odhlášení uživatele

1. Případ užití začíná, jestliže chce uživatel ukončit práci se systémem.
2. Uživatel stiskne tlačítko odhlásit.
3. Systém uživatele odhlásí.

Odstranit uživatele

Odstraní ze systému zvoleného uživatele. Odstraněný uživatel již nebude moci využívat služeb IS.

Tok událostí:

1.1.10 **Basic Path**

Odstranění uživatele

1. Případ užití začíná, jestliže chce vedoucí lékař odstranit některého z uživatelů systému.
2. INCLUDE (Zobrazit seznam uživatelů).
3. Systém odstraní vybraného uživatele.

Přidat nového uživatele

Umožní přidat do IS nového uživatele, který bude moci IS využívat. Při vytváření nového uživatele je nutné zadat uživatelské jméno a heslo, kterým se bude do IS přihlašovat. Dále je nutné zadat zda se jedná o vedoucího lékaře, lékaře nebo sestru.

Tok událostí:

1.1.11 **Basic Path**

Přidání nového uživatele

1. Případ užití začíná, jestliže se vedoucí lékař rozhodne vytvořit účet pro nového uživatele, který bude moci systém využívat.
2. Systém zobrazí formulář umožňující zadat: jméno, příjmení, uživatelské jméno, heslo a zařazení (vedoucí lékař, lékař, sestra).
3. Systém uloží nového uživatele do IS.

Přihlásit uživatele

Umožní uživateli, po zadání hesla, se přihlásit do systému v jedné z následujících rolí: Vedoucí lékař, Lékař, Sestra.

Tok událostí:

1.1.12 **Basic Path**

Přihlášení uživatele.

1. Případ užití začíná, jestliže chce uživatel začít pracovat se systémem.
2. Systém zobrazí formulář pro zadání uživatelského jména a hesla.
3. Uživatel zadá svoje uživatelské jméno a heslo.
4. Systém ověří zadané uživatelské jméno a heslo.
   4.1 Jestliže uživatel zadá neplatné jméno nebo heslo pokračuje případ užití bodem 2.
5. Systém přihlásí uživatele do systému s rolí, která byla zadanému uživatelskému jménu přiřazena.

Upravit údaje o uživateli

Umožňuje změnit veškeré údaje evidované u uživatele.

Tok událostí:

1.1.13 **Basic Path**

Upravení údajů o uživateli

1. Případ užití začíná, jestliže chce vedoucí lékař změnit údaje u některého z uživatelů.
2. INCLUDE (Zobrazit seznam uživatelů).
3. Systém zobrazí formulář umožňující změnit všechny údaje, které jsou u uživatele evidovány: jméno, příjmení, uživatelské jméno a heslo).
4. Lékař provede požadované změny.

5. Systém uloží nové údaje o uživateli.

Zadat základní údaje o zdravotnickém zařízení

Umožňuje zadat všechny údaje týkající se zdravotního zařízení, které IS využívá. Údaje, které je možné nastavit: IČO, DIČ, identifikační číslo zařízení IČZ, název zařízení, adresa zařízení, banka, číslo účtu, odbornost.

Tok událostí:

1.1.14 **Basic Path**

Zadání základních údajů o zařízení

1. Případ užití začíná, jestliže chce vedoucí lékař zadat či změnit některé z údajů evidovaných u zařízení.
2. Systém zobrazí formulář umožňující zadat: IČO, DIČ, identifikační číslo zařízení IČZ, název zařízení, adresa zařízení, banka, číslo účtu, odbornost.
3. Lékař zadá nové údaje.
4. Systém uloží nové informace o zdravotnickém zařízení.

Zobrazit seznam uživatelů

Zobrazí seznam všech uživatelů, kteří mohou IS využívat.

Tok událostí:

1.1.15 **Basic Path**

Zobrazení seznamu uživatelů

1. Případ užití začíná, jestliže uživatel potřebuje vybrat jednoho uživatele ze seznamu uživatelů.
2. Systém zobrazí jméno, příjmení a zařazení všech uživatelů, kteří mohou systém používat.
3. Uživatel jednoho ze zobrazených uživatelů vybere.

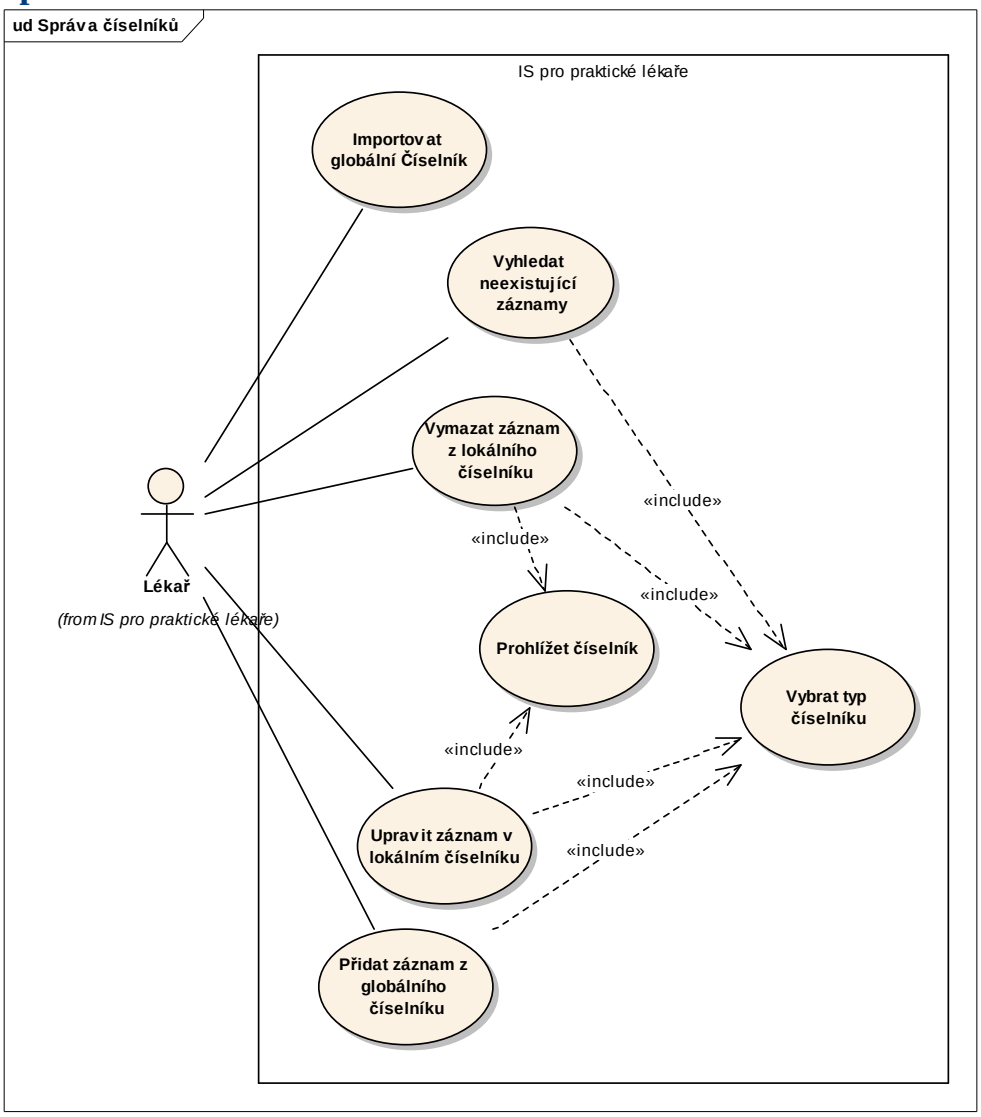
4. Modul správa číselníků

Případy užití v tomto balíčku jsou shodné pro všechny číselníky vydávané zdravotní pojišťovnou, a které tento systém využívá. Seznam číselníků, se kterými bude tento IS pracovat:

1. Mezinárodní klasifikace nemocí verze 10 - JDG4.XXX (Diagnózy)
2. Zdravotní výkony - VYKONY.XXX (Výkony)
3. Hromadně vyráběné léčivé přípravky a potraviny pro zvláštní lékařské účely - LEKY.XXX (Léky)
4. Zdravotnické prostředky - PZT.XXX (Zdravotnické prostředky)
5. Smluvní odbornosti pracovišť - ODBORN.XXX (Odbornosti)
6. Kapitační koeficienty
7. Pojišťovny.

Za úplným názvem číselníku je jméno souboru, ve kterém pojišťovna daný číselník distribuuje. V závorce je potom uveden zkrácený název číselníku využívaný v této analýze.

Správa číselníků



Obrázek:: Správa číselníků

Importovat globální Číselník

Provede načtení položek číselníku ze souboru dodávaného pojišťovnou do databáze.

Tok událostí:

1.1.16 **Basic Path**

Importování globálního číselníku

1. Případ užití začíná, jestliže chce lékař načíst do systému nový číselník od pojišťovny.
2. Systém umožní lékaři vybrat soubor, který má být do systému importován.
3. Lékař vybere požadovaný soubor.
4. Systém provede kontrolu správnosti formátu vybraného souboru.
5. V případě neplatného formátu systém zobrazí upozornění o neplatnosti číselníku.
6. Systém uloží veškeré údaje ze souboru do vlastní databáze.

Příloha:

D:\Dokumenty\Diplomka\VZP\DRCiselniky.pdf

Popis datového rozhraní číselníků vydávaných VZP.

Příloha:

D:\Dokumenty\Diplomka\VZP\DRCiselnikyZmeny.pdf

Popis změn v datovém rozhraní číselníků vydávaných VZP.

Příloha:

D:\Dokumenty\Diplomka\VZP\NoveDRCiselnikuLeky.pdf

Nové datové rozhraní číselníku Léky.

Prohlížet číselník

Zobrazí záznamy v lokálním číselníku u každého záznamu zobrazí pouze kód a název. Umožňuje výběr způsobu řazení zobrazených záznamů podle kódu nebo názvu. Umožní vybrat jeden ze zobrazených záznamů.

Tok událostí:

1.1.17 **Basic Path**

Výběr záznamu z číselníku

1. Případ užití začíná, jestliže se uživatel rozhodne zadat do formuláře záznam z číselníku.
2. Systém zobrazí seznam kódů a názvů všech záznamů zvoleného lokálního číselníku.
3. Uživatel jeden ze zobrazených záznamů vybere.

Přidat záznam z globálního číselníku

Zobrazí seznam záznamů ve vybraném globálním číselníku. Umožňuje přidání vybraných záznamů z globálního číselníku do číselníku lokálního.

Tok událostí:

1.1.18 **Basic Path**

Přidání záznamu z globálního číselníku

1. Případ užití začíná, když chce lékař přidat do lokálního číselníku nový záznam z číselníku globálního.
2. INCLUDE (Vybrat typ číselníku).
3. Systém zobrazí všechny záznamy v globálního číselníku.
4. Lékař vybere záznamy, které chce přidat do lokálního číselníku.
5. Systém přidá do lokálního číselníku vybrané záznamy.

Upravit záznam v lokálním číselníku

Umožňuje upravit záznam v lokálním číselníku.

Tok událostí:

1.1.19 **Basic Path**

Upravení záznamu v lokálním číselníku

1. Případ užití začíná, jestliže chce lékař upravit záznam v lokálním číselníku.
2. INCLUDE (Vybrat typ číselníku).
3. Systém požádá lékaře o výběr záznamu ve vybraném číselníku, který chce upravovat.
3. INCLUDE (Prohlížet číselník).
4. Systém zobrazí formulář umožňující upravit veškeré položky vybraného záznamu.
5. Lékař upraví požadované údaje.
6. Systém uloží upravený záznam do lokálního číselníku.

Vybrat typ číselníku

Umožní vybrat číselník, s kterým chce uživatel pracovat.

Tok událostí:

1.1.20 **Basic Path**

Vybrání typu číselníku

1. Případ užití začíná, jestliže chce uživatel vybrat jeden z číselníků evidovaných v systému.
2. Systém zobrazí seznam číselníků, které jsou v IS evidovány: Diagnózy, Výkony, Léky, Zdravotní prostředky, Odbornosti, Kapitační koeficienty, Pojišťovny.
3. Uživatel jeden z číselníků vybere.

Vyhledat neexistující záznamy

Zobrazí záznamy z lokálního číselníku, které byly zrušeny v číselníku globálním. Umožňuje tyto záznamy z lokálních číselníků vymazat. Používá se při stažení nových globálních číselníků.

Tok událostí:

1.1.21 **Basic Path**

Vyhledání neexistujících záznamů

1. Případ užití začíná, jestliže se lékař rozhodne zkontrolovat platnost záznamů v lokálním číselníku.
2. INCLUDE (Vybrat typ číselníku).
3. Systém prohledá záznamy v lokálním číselníku a vyhledá všechny, které nenalezne v číselníku globálním.
4. Systém zobrazí všechny vyhledané záznamy.
5. Lékař označí záznamy, které chce z lokálního číselníku odstranit.
6. Systém odstraní všechny vybrané záznamy.

Vymazat záznam z lokálního číselníku

Odstraní vybraný záznam z lokálního číselníku.

Tok událostí:

1.1.22 **Basic Path**

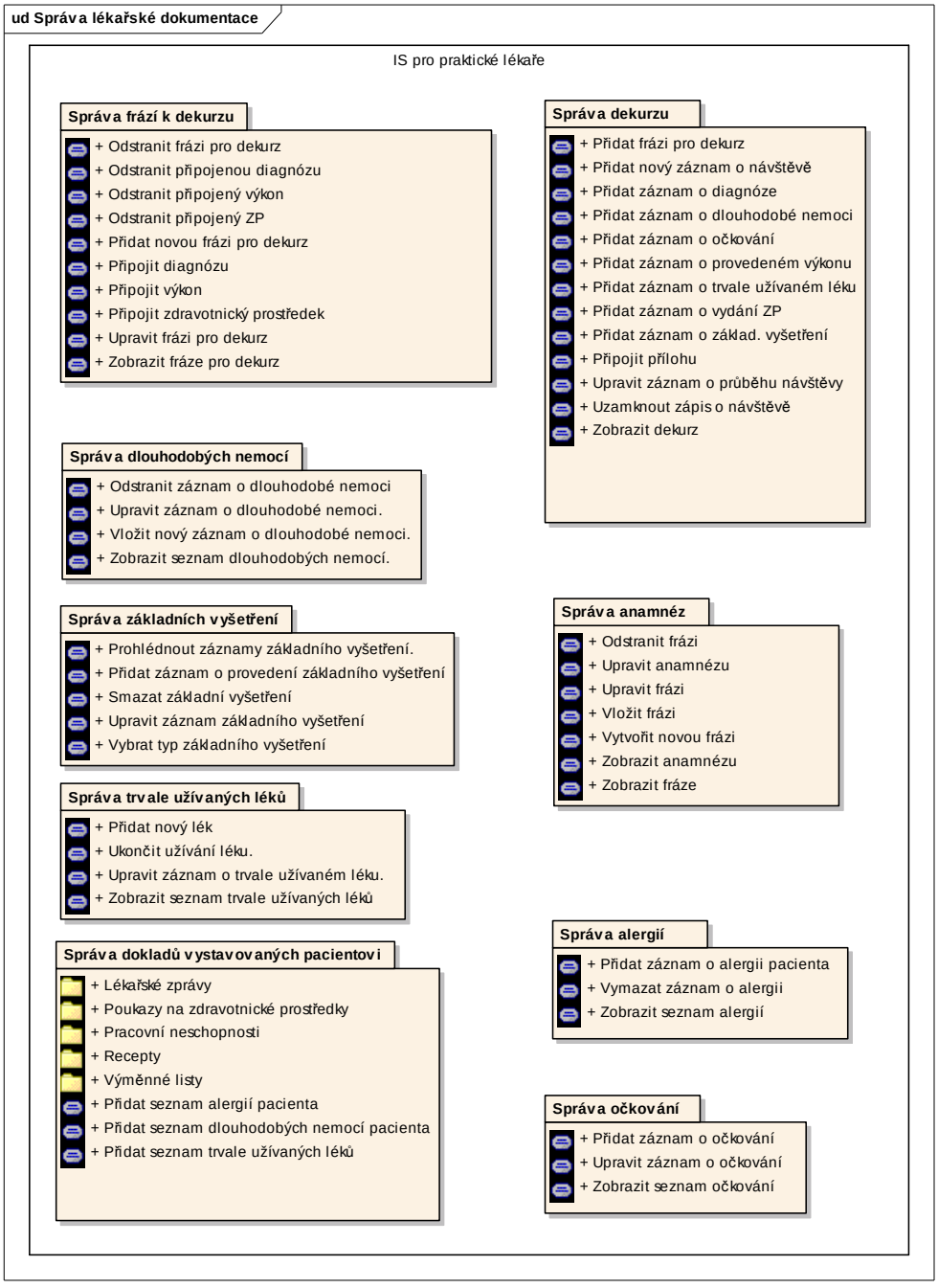
Vymazání záznamu z lokálního číselníku

1. Případ užití začíná, jestliže chce lékař odstranit záznam z lokálního číselníku.
2. Systém požádá lékaře o výběr záznamu, který chce odstranit.
3. INCLUDE (Prohlížet číselník).
4. Systém odstraní vybraný záznam.

5. Modul správa lékařské dokumentace

Balíček obsahuje případy užití související s vedením zdravotní dokumentace pacienta.

Správa lékařské dokumentace

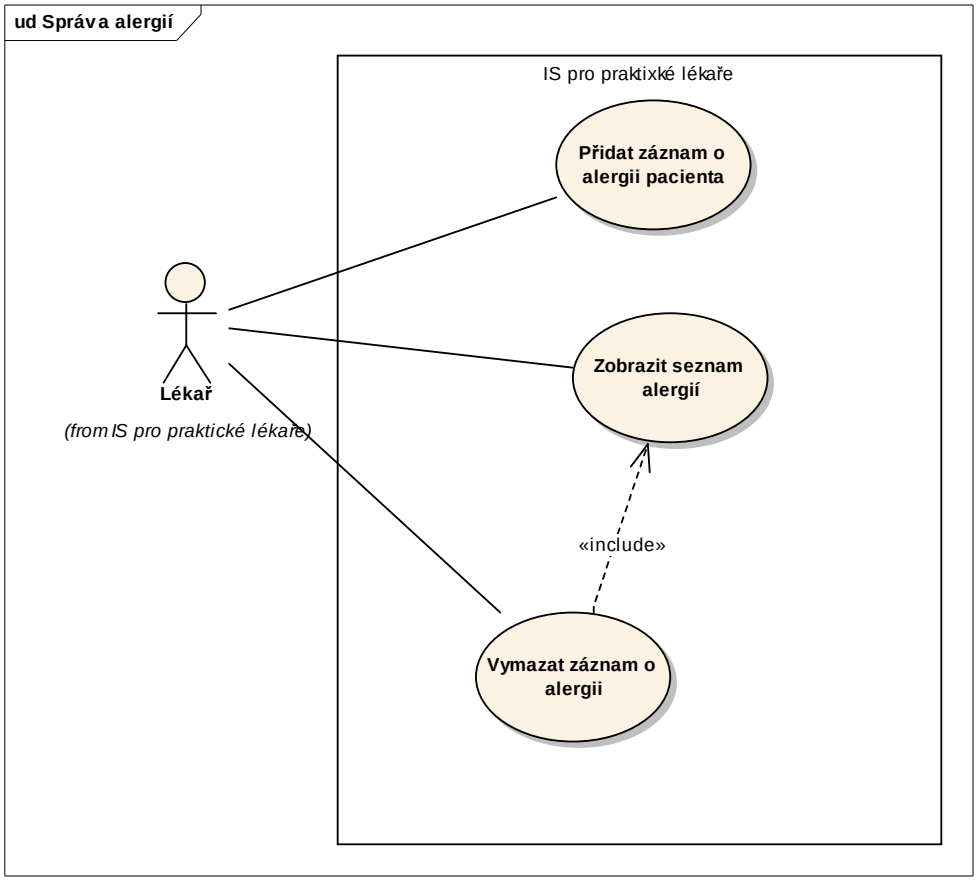


Obrázek:: Správa lékařské dokumentace

6. Správa alergií

Balíček obsahuje případy užití související s evidencí alergií pacienta.

Správa alergií



Obrázek:: Správa alergií

Přidat záznam o alergii pacienta

Přidá nový záznam o alergii pacienta na určitou látku. Záznam o alergii obsahuje název alergie a popis.

Tok událostí:

1.1.23 **Basic Path**

Přidání záznamu o alergii pacienta

1. Případ užití začíná, když chce lékař zadat do systému nový záznam o alergii pacienta.
2. Systém zobrazí formulář umožňující zadat: název a popis alergie.
3. Lékař vyplní požadované údaje.
4. Systém uloží nový záznam o alergii.

Vymazat záznam o alergii

Odstraní záznam o alergii.

Tok událostí:

1.1.24 **Basic Path**

Vymazání záznamu o alergii

1. Případ užití začíná, když chce lékař odstranit některý ze záznamů o alergii pacienta.
2. Systém požádá lékaře o výběr záznamu alergie, který chce odstranit.
3. INCLUDE (Zobrazit seznam alergií).
4. Systém odstraní vybraný záznam.

Zobrazit seznam alergií

Zobrazí seznam všech látek, na které je pacient alergický.

Tok událostí:

1.1.25 **Basic Path**

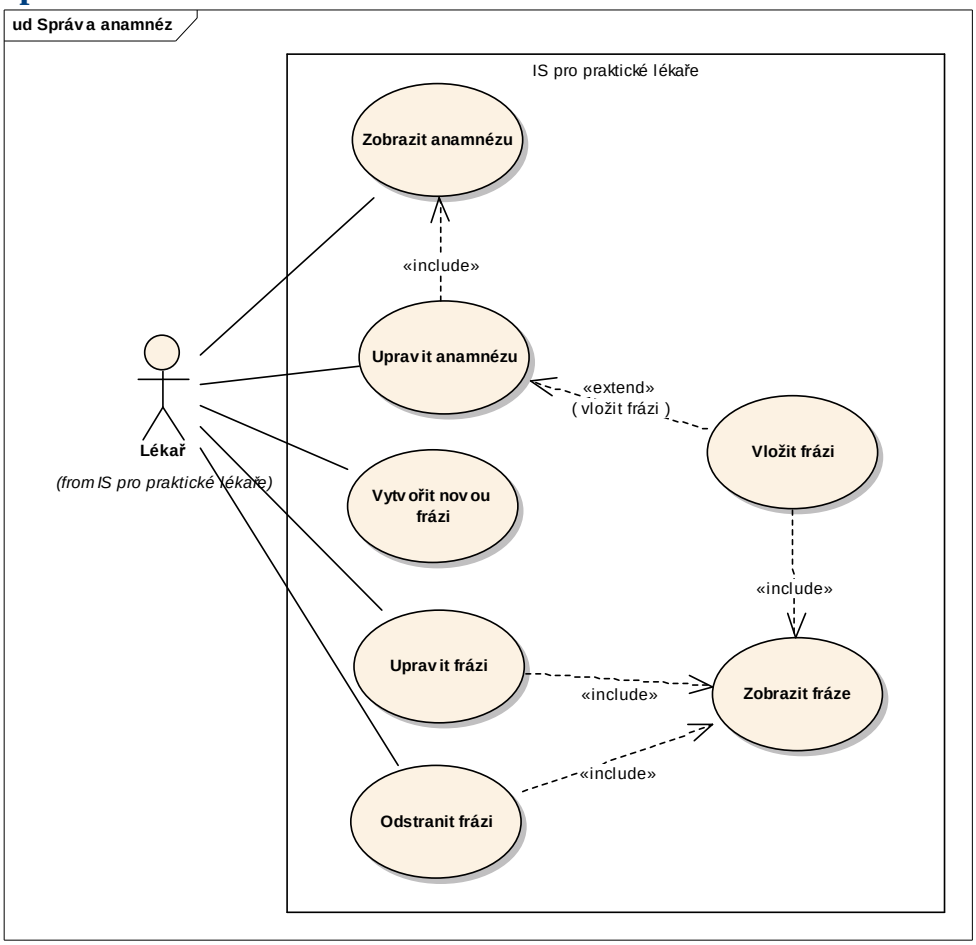
Zobrazení seznamu alergií

1. Případ užití začíná, když uživatel chce vybrat záznam ze seznamu alergií, které byly u vybraného pacienta diagnostikovány.
2. Systém zobrazí záznamy alergií, které byly u pacienta diagnostikovány.
3. Lékař vybere jeden ze zobrazených záznamů.

7. Správa anamnéz

Případy užití v tomto balíčku jsou shodné pro všechny typy evidovaných anamnéz. Evidované anamnézy jsou : Rodinná, Osobní, Pracovní, Sociální, Odborná. Každá anamnéza obsahuje položku umožňující zápis libovolného textu pro doplnění informací o pacientovi.

Správa anamnéz



Obrázek:: Správa anamnéz

Odstranit frázi

Vymaže ze seznamu frází pro anamnézu vybranou frázi.

Tok událostí:

1.1.26 **Basic Path**

Odstranění fráze

1. Případ užití začíná,jestliže chce lékař odstranit některou z frází pro anamnézu
2. INCLUDE (Zobrazit fráze pro anamnézu).
3.Systém vymaže vybranou frázi.

Upravit anamnézu

Umožňuje upravovat všechny údaje anamnézy, které byly do systému zadány. Na začátku jsou všechny anamnézy prázdné.

Tok událostí:

1.1.27 **Basic Path**

Upravení anamnézy

1. Případ užití začíná, když chce lékař doplnit či upravit anamnézu pacienta.
2. INCLUDE (Zobrazit anamnézu).
3. Lékař provede zápis požadovaných změn.
<vložit frázi>
4. Systém uloží provedené změny anamnézy.

Upravit frázi

Umožní upravit název nebo text fráze pro anamnézu.

Tok událostí:

1.1.28 **Basic Path**

Upravení fráze pro anamnézu

1. Případ užití začíná, když chce lékař změnit text některé z frází pro anamnézu.
2. INCLUDE (Zobrazit fráze pro anamnézu).
3. Systém zobrazí celý text vybrané fráze.
4. Lékař provede požadované změny.
5. Systém provedené změny uloží.

Vložit frázi

Vloží text fráze do zvolené anamnézy.

Tok událostí:

1.1.29 **Basic Path**

Vložení fráze do anamnézy

1. Případ užití začíná, když chce lékař vložit frázi do anamnézy.
2. INCLUDE (Zobrazit fráze pro anamnézu).
3. Systém vloží do anamnézy text vybrané fráze.

Vytvořit novou frázi

Vloží do seznamu frází pro anamnézu novou frázi obsahující název fráze a vlastní text.

Tok událostí:

1.1.30 **Basic Path**

Vytvoření fráze pro anamnézu

1. Případ užití začíná, když chce lékař vytvořit novou frázi pro anamnézu.
2. Systém zobrazí formulář umožňující zadat: název fráze a vlastní text.
3. Lékař vyplní požadované údaje.
4. Systém uloží frázi do systému.

Zobrazit anamnézu

Zobrazí všechny údaje anamnézy, které byly do systému zadány.

Tok událostí:

1.1.31 **Basic Path**

Zobrazení anamnézy

1. Případ užití začíná, když si chce lékař prohlédnout pacientovu anamnézu.
2. Systém zobrazí veškeré informace evidované v anamnéze.

Zobrazit fráze

Zobrazí názvy všech frází pro anamnézu. Umožňuje jednu ze zobrazených frází vybrat.

Tok událostí:

1.1.32 **Basic Path**

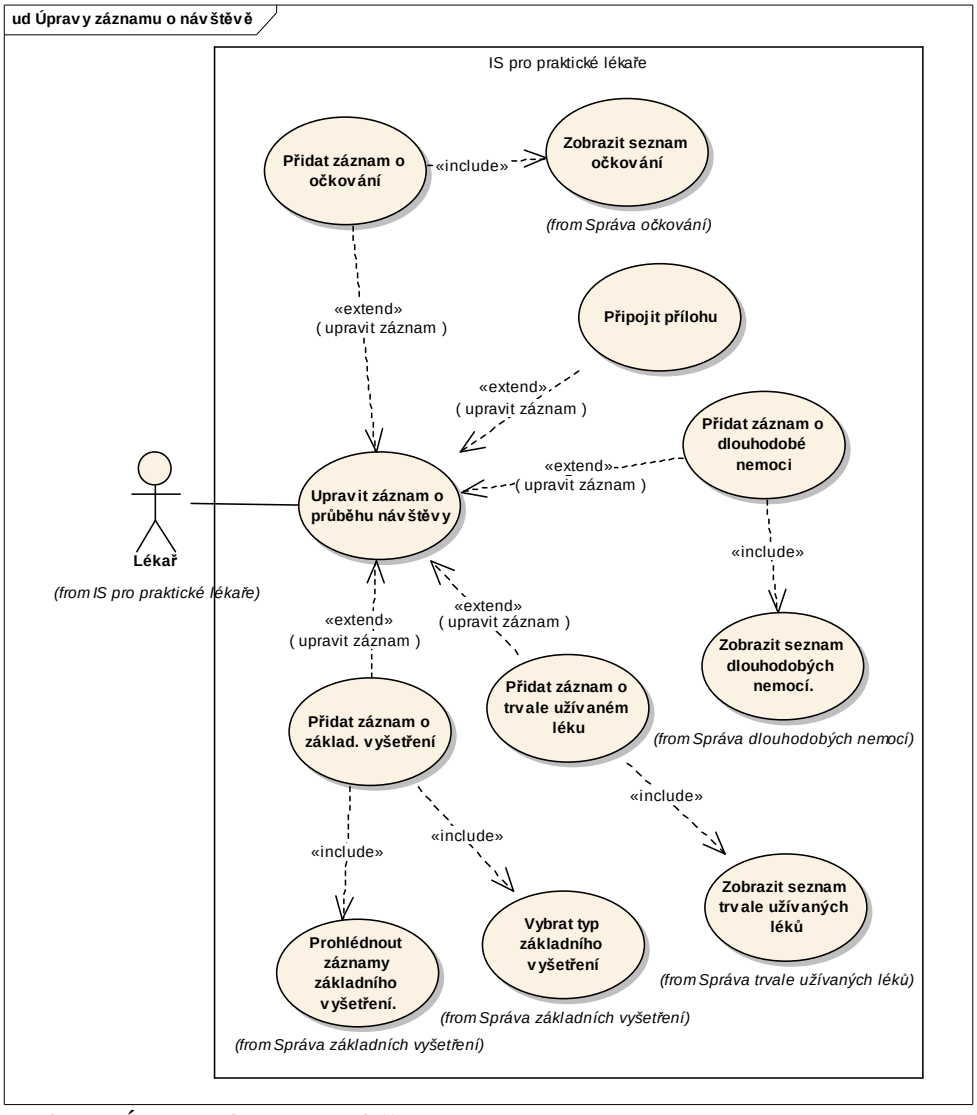
Zobrazení frází pro anamnézu

1. Případ užití začíná, když uživatel chce vybrat některou z frází pro anamnézu.
2. Systém zobrazí seznam názvů frází pro anamnézu.
3. Uživatel jednu z frází vybere.

8. Správa dekurzu

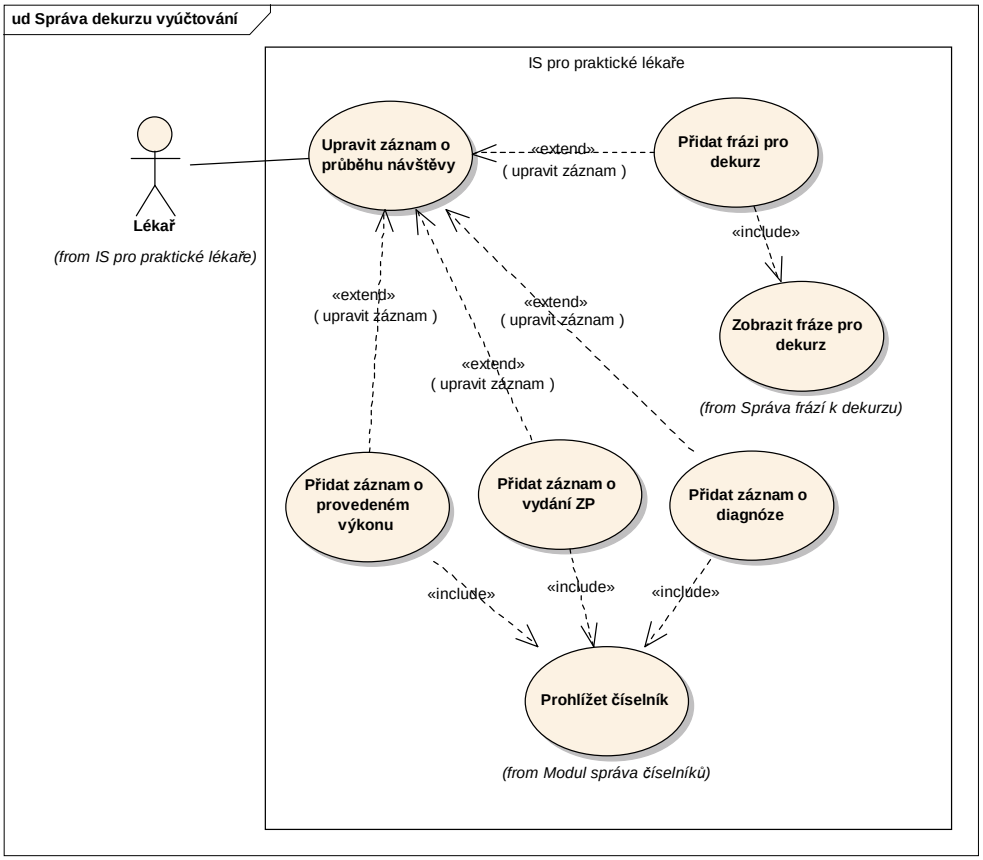
Balíček obsahuje případy užití související se zápisem o průběhu návštěvy pacienta u lékaře.

Úpravy záznamu o návštěvě



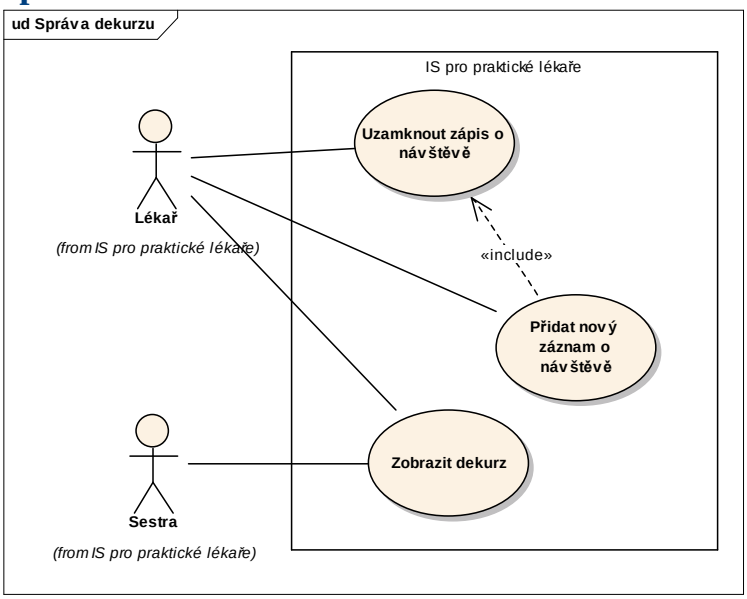
Obrázek:: Úpravy záznamu o návštěvě

Správa dekurzu vyúčtování



Obrázek:: Správa dekurzu vyúčtování

Správa dekurzu



Obrázek:: Správa dekurzu

Přidat frázi pro dekurz

Připojí k záznamu o průběhu návštěvy frázi pro dekurz.

Tok událostí:

1.1.33 **Basic Path**

Přidání fráze do dekurzu

1. Případ užití začíná, když chce lékař přidat do dekurzu frázi.
2. INCLUDE (Zobrazit seznam frází pro dekurz).
3. Systém přepíše text vybrané fráze do dekurzu.
4. Jsou-li k frázi připojeny další položky zapíše tyto položky do dekurzu také.

Přidat nový záznam o návštěvě

Přidá nový záznam o návštěvě do dekurzu pacienta. Automaticky k tomuto záznamu přidá aktuální datum a čas.

Tok událostí:

1.1.34 **Basic Path**

Přidání záznamu o návštěvě do dekurzu

1. Případ užití začíná, když lékař zadá příkaz Přidat záznam o návštěvě.
2. Jestliže předchozí záznam o průběhu návštěvy pacienta není uzamčen, pak:
   2.1 INCLUDE (Uzamknout zápis o návštěvě)
2. Systém přidá do dekurzu pacienta zápis o nové návštěvě. Text minulé návštěvy oddělí vodorovnou čarou.
3. K novému zápisu o návštěvě systém přidá datum a čas návštěvy.

Přidat záznam o diagnóze

Přidá záznam o diagnóze do zápisu o průběhu návštěvy.

Tok událostí:

1.1.35 **Basic Path**

Přidání záznamu o diagnóze

1. Případ užití začíná, když chce lékař zaznamenat diagnózu do dekurzu.
2. Systém požádá lékaře o zadání kódu diagnózy.
3. INCLUDE (Prohlížet číselník). Zobrazí číselník Diagnózy.
4. Lékař vybere požadovanou diagnózu.
5. Systém uloží diagnózu z důvodu pozdějšího vyúčtování výkonů a záznam o diagnóze přepíše do dekurzu.

Přidat záznam o dlouhodobé nemoci

Připojí k záznamu o průběhu návštěvy záznam o dlouhodobé nemoci, diagnostikované u pacienta, ze seznamu dlouhodobých nemocí.

Tok událostí:

1.1.36 **Basic Path**

Přidání záznamu o dlouhodobé nemoci

1. Případ užití začíná, když lékař zadá příkaz Připojit záznam o dlouhodobé nemoci.
2. INCLUDE (Zobrazit seznam dlouhodobých nemocí).
3. Systém přepíše záznam o vybrané dlouhodobé nemoci do dekurzu.

Přidat záznam o očkování

Připojí k záznamu o průběhu návštěvy zápis o provedeném očkování ze seznamu očkování.

Tok událostí:

1.1.37 **Basic Path**

Přidání záznamu o očkování

1. Případ užití začíná, když lékař zadá příkaz Připojit záznam o očkování.
2. INCLUDE (Zobrazit seznam očkování).
3. Systém přepíše záznam o vybraném očkování do dekurzu.

Přidat záznam o provedeném výkonu

Přidá záznam o provedení výkonu do zápisu o průběhu návštěvy.

Tok událostí:

1.1.38 **Basic Path**

Přidání záznamu o provedeném výkonu

1. Případ užití začíná, když chce lékař zaznamenat provedení výkonu.
2. Systém požádá lékaře o zadání kódu výkonu.
3. INCLUDE (Prohlížet číselník). Zobrazí číselník Výkony.
4. Lékař vybere výkon, který provedl.
5. Systém požádá lékaře o zadání počtu, kolikrát daný výkon provedl.
6. Lékař zadá počet výkonů.
7. Systém uloží provedené výkony a záznam o provedení přepíše do dekurzu.

Přidat záznam o trvale užívaném léku

Připojí k záznamu o průběhu návštěvy záznam o trvale užívaném léku pacientem ze seznamu trvale užívaných léků.

Tok událostí:

1.1.39 **Basic Path**

Přidání záznamu o trvale užívaném léku

1. Případ užití začíná, když lékař zadá příkaz Připojit záznam o trvale užívaném léku.
2. INCLUDE (Zobrazit seznam trvale užívaných léků).
3. Systém přepíše záznam o vybraném léku do dekurzu.

Přidat záznam o vydání ZP

Přidá záznam o vydání zdravotního prostředku (ZP) do zápisu o průběhu návštěvy.

Tok událostí:

1.1.40 **Basic Path**

Přidání záznamu o vydání ZP

1. Případ užití začíná, když chce lékař zaznamenat vydání zdravotnického prostředku pacientovi.
2. Systém požádá lékaře o zadání kódu ZP.
3. INCLUDE (Prohlížet číselník). Zobrazí číselník ZP.
4. Lékař vybere ZP, který pacientovi vydal.
5. Systém požádá o zadání množství ZP.
6. Lékař zadá vydané množství ZP.
7. Systém uloží vydané ZP a záznam o vydání přepíše do dekurzu.

Přidat záznam o základ. vyšetření

Připojí k záznamu o průběhu návštěvy zápis o provedeném základním vyšetření.

Tok událostí:

1.1.41 **Basic Path**

Přidat záznam o základním vyšetření

1. Případ užití začíná, když lékař zadá příkaz Připojit záznam o základním vyšetření.
2. Systém požádá a výběr typu základního vyšetření, které chce lékař přidat.
3. INCLUDE (Vybrat typ základního vyšetření).
4. INCLUDE (Prohlédnout záznamy základního vyšetření). Zobrazí vybraný typ základního vyšetření z předešlého kroku.
5. Systém zapíše záznam o vybraném základním vyšetření do dekurzu.

Připojit přílohu

Připojí k záznamu o průběhu návštěvy požadovaný soubor jako přílohu. (Obrazová dokumentace, zprávy z vyšetření u specialisty apod.)

Tok událostí:

1.1.42 **Basic Path**

Připojení přílohy

1. Případ užití začíná, když lékař zadá příkaz Připojit přílohu k dekurzu.
2. Systém zobrazí dialog umožňující procházet složky na disku či přenosném médiu a vybrat požadovaný soubor pro připojení.
3. Lékař vybere požadovaný soubor pro připojení.
4. Systém uloží požadovaný soubor.
5. Systém do dekurzu přidá zápis o připojení přílohy jako odkaz, aby bylo možné později kliknutím na tento odkaz dokument otevřít.

Upravit záznam o průběhu návštěvy

Umožňuje upravovat záznam o průběhu návštěvy pacienta u lékaře. Dále umožňuje automaticky doplnit do záznamu další údaje a připojit různé přílohy.

Tok událostí:

1.1.43 **Basic Path**

Upravení záznamu o návštěvě

1. Případ užití začíná, jestliže se lékař rozhodne upravit záznam o návštěvě.
2. Lékař upraví či doplní stávající záznam o průběhu návštěvy.
<upravit záznam>
3. Lékař potvrdí provedené změny.
4. Systém uloží provedené změny.

Uzamknout zápis o návštěvě

Umožní zápis o návštěvě uzamknout a zabezpečit elektronickým podpisem tak, aby nebylo možné zápis později změnit. Veškeré změny po uzamknutí zápisu musí být uloženy jako opravné zápisy, aby bylo možné kdykoliv zjistit původní zápis o průběhu návštěvy. Tato funkce je vyžadována v případě, že lékař nechce současně vést též papírovou formu dokumentace.

Tok událostí:

1.1.44 **Basic Path**

Uzamčení zápisu

1. Případ užití začíná, jestliže se lékař rozhodne uzamknout zápis o průběhu návštěvy.
2. Systém zápis opatří elektronickým podpisem a uzamkne tak, aby nebylo možné provádět později dodatečné změny v zápisu.

Zobrazit dekurz

Zobrazí dekurz pacienta. Jednotlivé návštěvy od sebe budou odděleny oddělovačem s uvedeným datumem, kdy se daná návštěva uskutečnila.

Tok událostí:

1.1.45 **Basic Path**

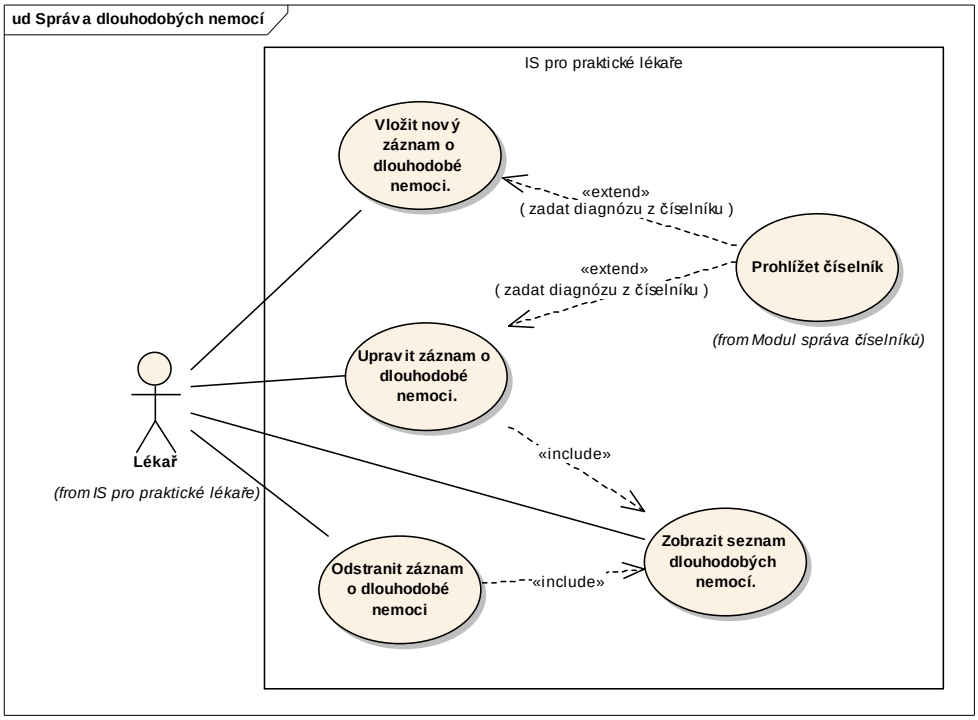
Zobrazení dekurzu

1. Případ užití začíná, jestliže si chce uživatel zobrazit dekurz pacienta.
2. Systém zobrazí jednotlivé zápisy o návštěvách uspořádané podle datumu. Jednotlivé návštěvy budou od sebe odděleny horizontální čárou, u které bude zobrazen datum návštěvy.

9. Správa dlouhodobých nemocí

Balíček obsahuje případy užití související s evidencí dlouhodobých nemocí u každého pacienta.

Správa dlouhodobých nemocí



Obrázek:: Správa dlouhodobých nemocí

Odstranit záznam o dlouhodobé nemoci

Vymaže záznam o dlouhodobé nemoci u vybraného pacienta.

Tok událostí:

1.1.46 **Basic Path**

Odstranění záznamu o dlouhodobé nemoci

1. Případ užití začíná, když chce lékař odstranit některý ze záznamů o dlouhodobé nemoci pacienta.
2. Systém požádá lékaře o výběr záznamu dlouhodobé nemoci, který chce odstranit.
3. INCLUDE (Zobrazit seznam dlouhodobých nemocí).
4. Systém odstraní vybraný záznam.

Upravit záznam o dlouhodobé nemoci.

Umožňuje opravit údaje uvedené u zvolené nemoci (kód, název, poznámku, datum od, datum do).

Tok událostí:

1.1.47 **Basic Path**

Upravení záznamu o dlouhodobé nemoci

1. Případ užití začíná, když chce lékař upravit některý ze záznamů o dlouhodobé nemoci pacienta.
2. Systém požádá lékaře o výběr záznamu dlouhodobé nemoci, který chce upravovat.
3. INCLUDE (Zobrazit seznam dlouhodobých nemocí).
4. Systém zobrazí formulář umožňující upravit veškeré údaje evidované u vybraného záznamu (kód diagnózy, název diagnózy, datum od, datum do a vlastní poznámku).
<zadat diagnózu z číselníku> Zobrazí číselník Diagnózy.
5. Lékař upraví požadované údaje.
6. Systém uloží do záznamu všechny změny provedené lékařem.

Vložit nový záznam o dlouhodobé nemoci.

Vloží do seznamu dlouhodobých nemocí nově diagnostikovanou nemoc a umožňuje k ní připojit vlastní poznámku.

Tok událostí:

1.1.48 **Basic Path**

Přidání záznamu o dlouhodobé nemoci

1. Případ užití začíná, když chce lékař zadat do systému nový záznam o dlouhodobé nemoci pacienta.
2. Systém zobrazí formulář umožňující zadat: kód diagnózy, název diagnózy, datum od, datum do a vlastní poznámku.
3. Lékař vyplní požadované údaje.
<zadat diagnózu z číselníku> Zobrazí číselník Diagnózy.
4. Systém uloží nový záznam o dlouhodobé nemoci.

Zobrazit seznam dlouhodobých nemocí.

Zobrazí seznam všech dlouhodobých nemocí, které byly u daného pacienta diagnostikovány. Umožňuje jeden ze zobrazených záznamů vybrat.

Tok událostí:

1.1.49 **Basic Path**

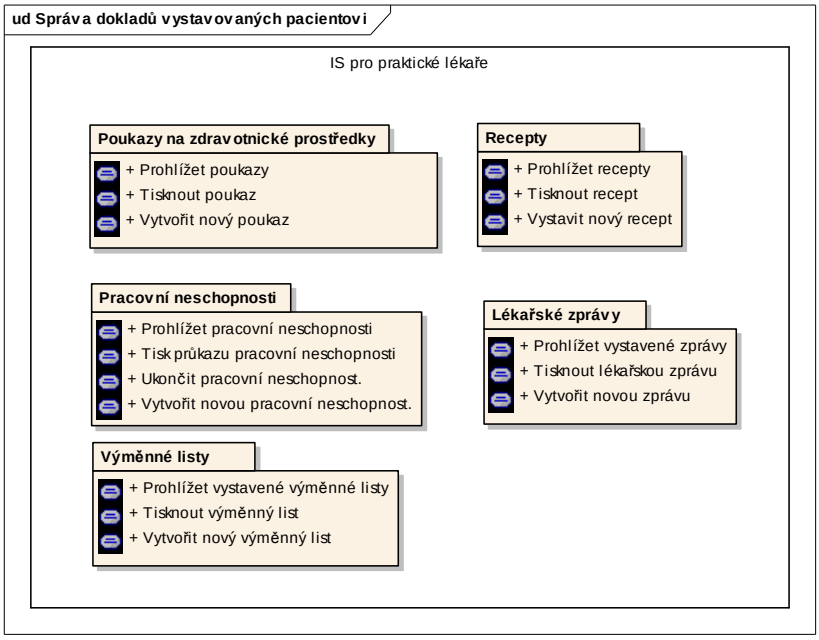
Zobrazení seznamu dlouhodobých nemocí.

1. Případ užití začíná, když uživatel chce vybrat záznam ze seznamu dlouhodobých nemocí, které byly u vybraného pacienta diagnostikovány.
2. Systém zobrazí seznam dlouhodobých nemocí, které byly u pacienta diagnostikovány.
3. Uživatel vybere jeden ze zobrazených záznamů.

10.Správa dokladů vystavovaných pacientovi

Balíček obsahuje případy užití související s evidencí dokladů, které byly pacientovi vystaveny.

Správa dokladů vystavovaných pacientovi



Obrázek:: Správa dokladů vystavovaných pacientovi

Přidat seznam alergií pacienta

Na zvolený dokument doplní seznam alergií pacienta.

Tok událostí:

1.1.50 **Basic Path**

Přidání seznamu alergií pacienta

1. Případ užití začíná, když chce uživatel na dokument vyplnit seznam alergií pacienta.
2. Systém přepíše seznam alergií pacienta na vybraný dokument.

Přidat seznam dlouhodobých nemocí pacienta

Na zvolený dokument doplní seznam dlouhodobých nemocí pacienta.

Tok událostí:

1.1.51 **Basic Path**

Přidání seznamu dlouhodobých nemocí

1. Případ užití začíná, když chce uživatel na dokument vyplnit seznam dlouhodobých nemocí.
2. Systém přepíše seznam dlouhodobých nemocí pacienta na vybraný dokument.

Přidat seznam trvale užívaných léků

Na zvolený dokument doplní seznam trvale užívaných léků pacienta.

Tok událostí:

1.1.52 **Basic Path**

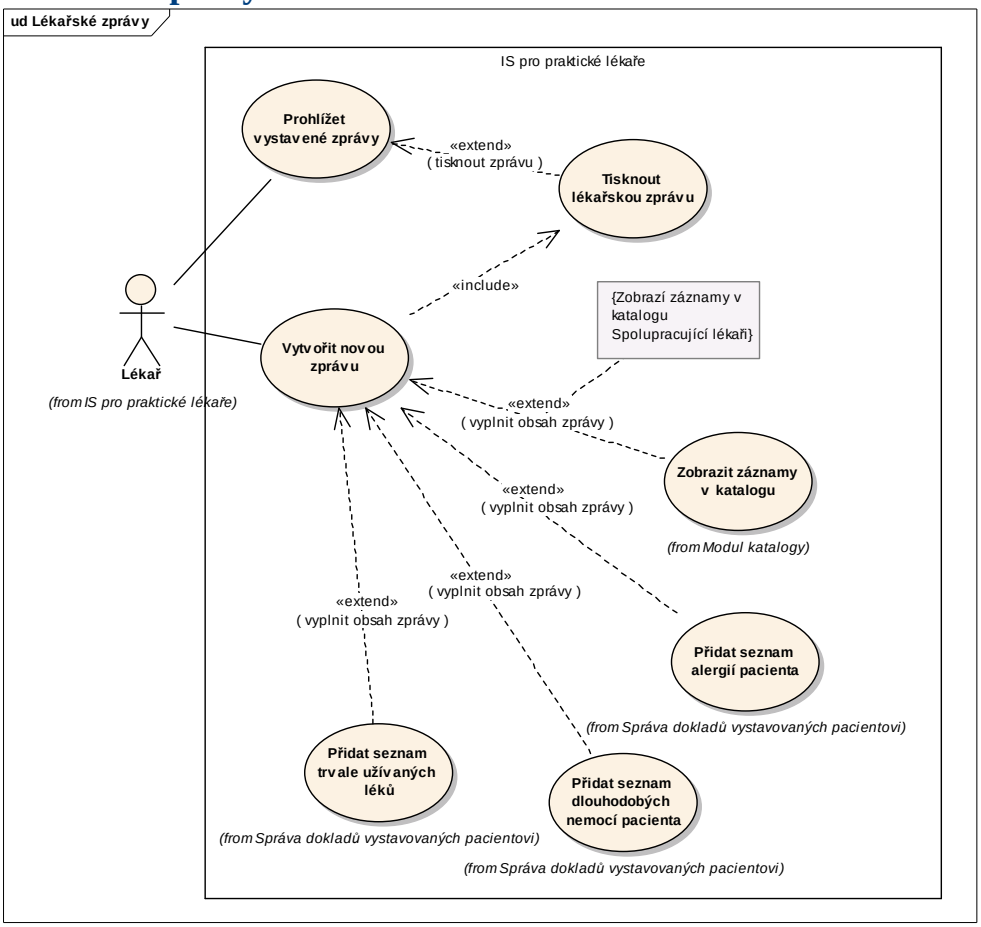
Přidání seznamu trvale užívaných léků

1. Případ užití začíná, když chce uživatel na dokument vyplnit seznam trvale užívaných léků pacientem.
2. Systém přepíše seznam trvale užívaných léků pacientem na vybraný dokument.

11.Lékařské zprávy

Balíček obsahuje případy užití související s vystavováním a evidencí již vystavených lékařských zpráv pacientovi.

Lékařské zprávy



Obrázek:: Lékařské zprávy

Prohlížet vystavené zprávy

Zobrazí seznam všech doposud vystavených lékařských zpráv, které byly zvolenému pacientovi vystaveny.

Tok událostí:

1.1.53 **Basic Path**

Prohlížení lékařských zpráv

1. Případ užití začíná, když si lékař chce prohlédnout vystavené zprávy pacientovi.
2. Systém zobrazí seznam všech vystavených zpráv.
3. Lékař vybere zprávu, kterou chce zobrazit.
4. Systém zobrazí vybranou lékařskou zprávu.

<tisknout zprávu>

Tisknout lékařskou zprávu

Vytiskne vybranou lékařskou zprávu.

Tok událostí:

1.1.54 **Basic Path**

Tisknutí lékařské zprávy

1. Případ užití začíná, když chce lékař vytisknout lékařskou zprávu.
2. Systém vytiskne lékařskou zprávu.

Vytvořit novou zprávu

Vytvoří novou lékařskou zprávu. Lékařská zpráva obsahuje: datum, adresu zdravotnického zařízení pro které je zpráva vystavována, jméno a rodné číslo pacienta a vlastní text zprávy.

Tok událostí:

1.1.55 **Basic Path**

Vytvoření nové zprávy

1. Případ užití začíná, když se lékař rozhodne vystavit pacientovi novou lékařskou zprávu.
2. Systém zobrazí formulář umožňující zadat datum, text zprávy a adresu zdravotnického zřízení pro které je zpráva určena.

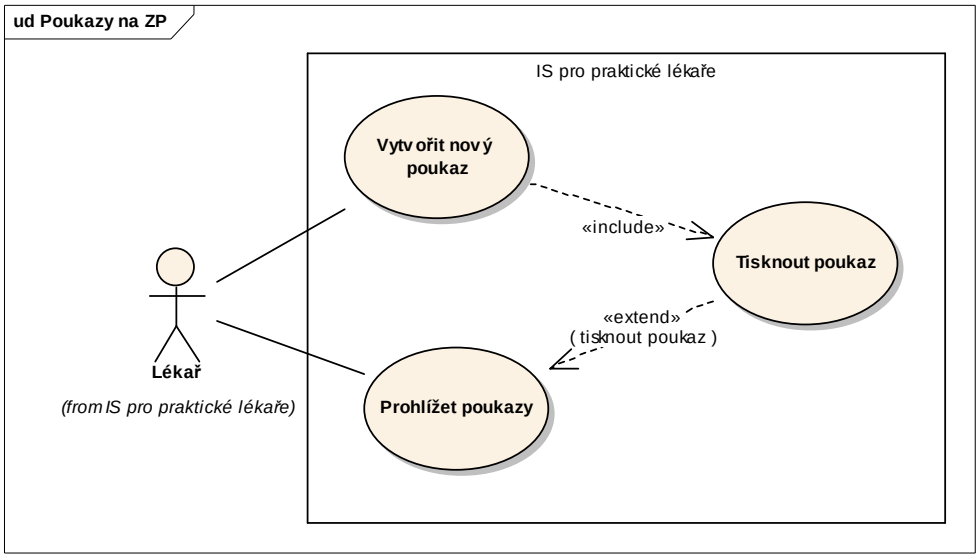
<vyplnit obsah zprávy>

3. Lékař vyplní požadované údaje.
4. INCLUDE (Tisknout lékařskou zprávu)
5. Systém uloží vytvořenou zprávu do seznamu zpráv.

12.Poukazy na zdravotnické prostředky

Balíček obsahuje případy užití související s vystavování a evidencí již vystavených poukazů na zdravotnické prostředky.

Poukazy na ZP



Obrázek:: Poukazy na ZP

Prohlížet poukazy

Zobrazí seznam všech poukazů, které byly pacientovi vystaveny.

Tok událostí:

1.1.56 **Basic Path**

Prohlížení poukazů

1. Případ užití začíná, když si lékař chce prohlédnout vystavené poukazy pacientovi.
2. Systém zobrazí seznam všech vystavených poukazů.
3. Lékař vybere poukaz, který chce zobrazit.
4. Systém zobrazí vybraný poukaz na zdravotnický prostředek.

<tisknout poukaz>

Tisknout poukaz

Vytiskne poukaz na zdravotní pomůcku na předepsaný formulář nebo na čistý papír.

Tok událostí:

1.1.57 **Basic Path**

Tisknutí poukazu

1. Případ užití začíná, když chce lékař vytisknout poukaz na zdravotnický prostředek.
2. Systém požádá lékaře o výběr, zda se má tisk provést na čistý papír nebo předtištěný formulář.
2. Systém poukaz vytiskne.

Vytvořit nový poukaz

Vytvoří nový záznam o vystavení poukazu pacientovi.

Tok událostí:

1.1.58 **Basic Path**

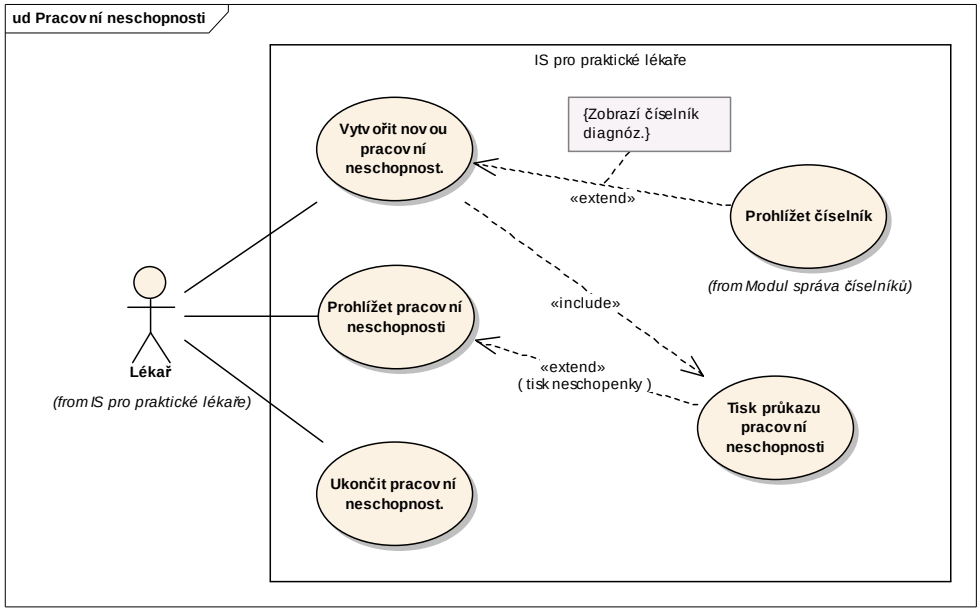
Vytvoření nového poukazu

1. Případ užití začíná, když se lékař rozhodne vystavit pacientovi nový poukaz.
2. Systém zobrazí formulář umožňující zadat: jméno a příjmení pacienta, bydliště, diagnózu, kód pomůcky, název pomůcky, množství, datum, vlastní text.
3. Lékař vyplní požadované údaje.
4. INCLUDE (Tisknout poukaz)
5. Systém uloží vytvořený poukaz do seznamu poukazů.

13.Pracovní neschopnosti

Balíček obsahuje případy užití související s vystavováním a evidencí již vystavených průkazů pracovní neschopnosti.

Pracovní neschopnosti



Obrázek:: Pracovní neschopnosti

Prohlížet pracovní neschopnosti

Zobrazí seznam všech pracovních neschopností vybraného pacienta.

Tok událostí:

1.1.59 **Basic Path**

Prohlížení pracovních neschopností

1. Případ užití začíná, jestliže si chce lékař prohlédnout všechny pracovní neschopnosti, které danému pacientovi vystavil.
2. Systém zobrazí seznam všech vystavených neschopností - zobrazí pouze datum od a datum do.
3. Lékař si výběrem jedné z neschopností může zobrazit detailní informace o neschopnosti.

<tisk neschopenky>

Tisk průkazu pracovní neschopnosti

Vytiskne průkaz o pracovní neschopnosti pacienta.

Tok událostí:

1.1.60 **Basic Path**

Tisknutí průkazu pracovní neschopnosti

1. Případ užití začíná, jestliže chce lékař vytisknout průkaz pracovní neschopnosti.
2. Systém vytiskne požadovaný průkaz.

Ukončit pracovní neschopnost.

Umožňuje zadat datum ukončení pracovní neschopnosti.

Tok událostí:

1.1.61 **Basic Path**

Ukončení pracovní neschopnosti

1. Případ užití začíná, jestliže se lékař rozhodne ukončit pracovní neschopnost pacienta.
2. Systém požádá lékaře o zadání datumu, ke kterému má být neschopnost ukončena.
3. Lékař zadá požadované datum. Implicitně aktuální datum.
4. Systém doplní do naposledy vystaveného průkazu neschopnosti do položky "datum do" zadané datum.

Vytvořit novou pracovní neschopnost.

Vytvoří novou pracovní neschopnost. Záznam o pracovní neschopnosti obsahuje údaje: datum začátku neschopnosti, datum ukončení neschopnosti, číslo průkazu a diagnózu.

Tok událostí:

1.1.62 **Basic Path**

Vytvoření průkazu pracovní neschopnosti

1. Případ užití začíná, jestliže se lékař rozhodne vystavit pacientovi průkaz pracovní neschopnosti.
2. Systém zobrazí formulář umožňující zadat: datum od, datum do, adresu pobytu během neschopnosti, diagnózu, datum vystavení a číslo zdravotního průkazu.

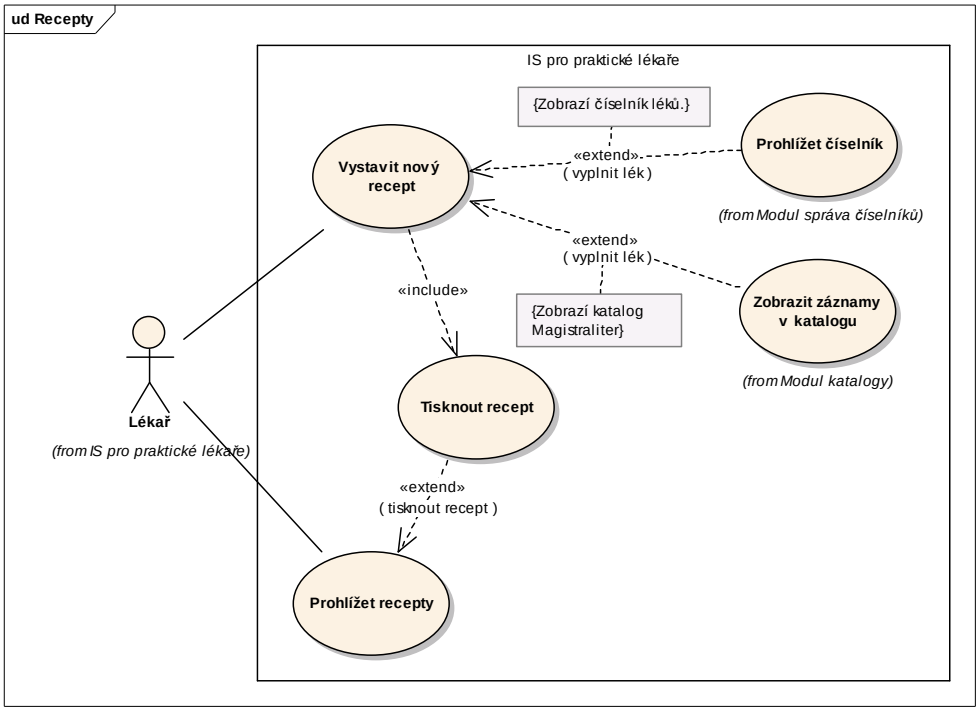
<zadání diagnózy>

3. Lékař vyplní požadované údaje.
4. INCLUDE (Tisk průkazu pracovní neschopnosti).
5. Systém uloží záznam o vydání průkazu pracovní neschopnosti.

14.Recepty

Balíček obsahuje případy užití související s vystavování a evidencí již vystavených receptů.

Recepty



Obrázek:: Recepty

Prohlížet recepty

Zobrazí seznam všech receptů, které byly pacientovi vystaveny.

Tok událostí:

1.1.63 **Basic Path**

Prohlížení vystavených receptů

1. Případ užití začíná, když si lékař chce prohlédnout vystavené recepty.
2. Systém zobrazí seznam všech vystavených receptů.
3. Lékař vybere recept, který chce zobrazit.
4. Systém zobrazí vybraný recept.

<tisknout recept>

Tisknout recept

Vytiskne zvolený recept na příslušný formulář. Tiskopis receptu vydávaného pojišťovnou je v příloze.

Tok událostí:

1.1.64 **Basic Path**

Tisknutí receptu

1. Případ užití začíná, jestliže chce lékař vytisknout pacientovi recept na léky.
2. Systém vytiskne na formulář dodávaný pojišťovnou informace o předepsaném léku.

Příloha:

D:\Dokumenty\Diplomka\VZP\Tiskopisy\Recept.pdf

Tiskopis receptu vydávaného VZP.

Vystavit nový recept

Vytvoří nový záznam o vydání receptu pacientovi. Recept obsahuje kód a název předepsaného léku, dávkování, popřípadě další poznámku.

Tok událostí:

1.1.65 **Basic Path**

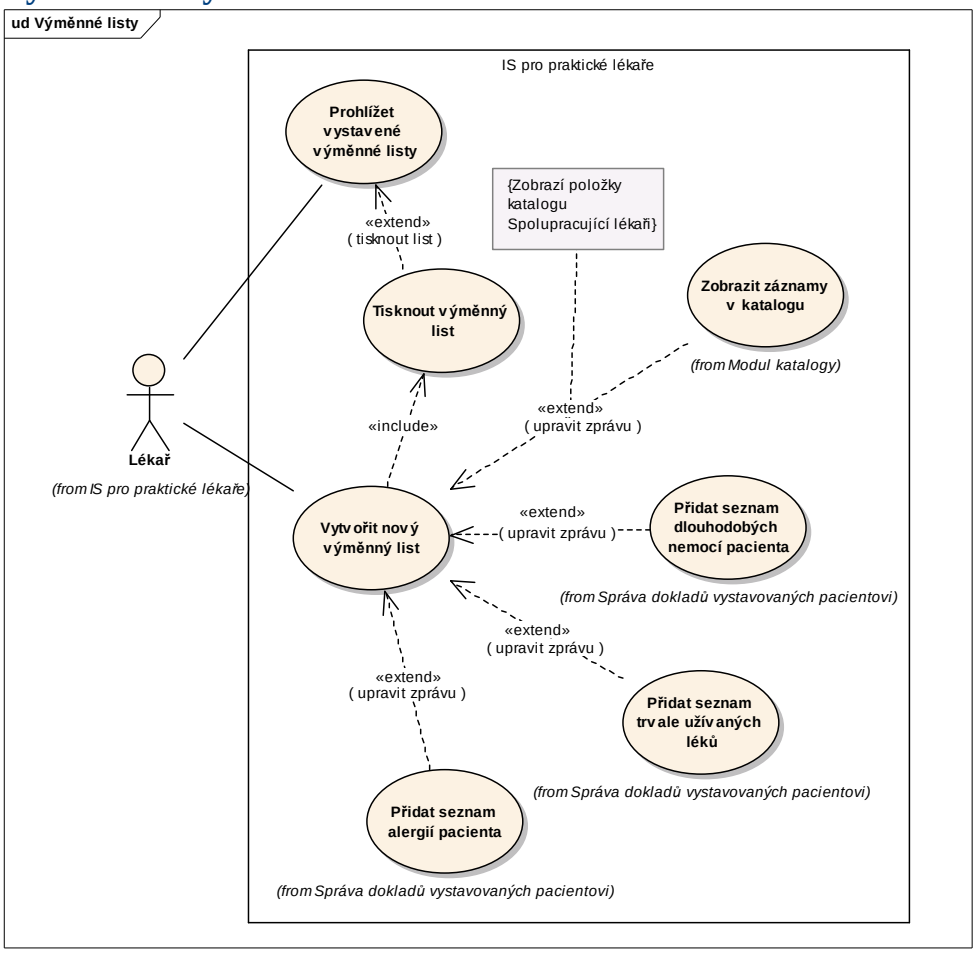
Vystavení nového receptu

1. Případ užití začíná, když se lékař rozhodne vystavit pacientovi nový recept na léky.
2. Systém zobrazí formulář umožňující zadat: kód léku, název léku, dávkování a poznámku. Na každém receptu můžou být maximálně dva léky.
3. Lékař vyplní požadované údaje.
<vyplnit lék>
4. INCLUDE (Tisknout recept)
5. Systém uloží vystavený recept do seznamu receptů.

15. Výměnné listy

Balíček obsahuje případy užití související s vystavováním a evidencí již vystavených výměnných listů.

Výměnné listy



Obrázek:: Výměnné listy

Prohlížet vystavené výměnné listy

Zobrazí seznam všech dosud vystavených výměnných listů, které byly pacientovi vystaveny.

Tok událostí:

1.1.66 **Basic Path**

Prohlížení výměnných listů

1. Případ užití začíná, když si chce lékař prohlédnout vystavené výměnné listy.
2. Systém zobrazí seznam všech výměnných listů, které byly pacientovi vystaveny.
3. Lékař vybere výměnný list, který chce zobrazit.
4. Systém zobrazí vybraný výměnný list.

<tisknout list>

Tisknout výměnný list

Vytiskne požadovaný výměnný list.

Tok událostí:

1.1.67 **Basic Path**

Tisknutí výměnného listu

1. Případ užití začíná, jestliže chce lékař vytisknout výměnný list pro pacienta.
2. Systém vytiskne výměnný list.

Vytvořit nový výměnný list

Vytvoří nový výměnný list. Výměnný list obsahuje: datum, adresu cílového zdravotnického zařízení, které je možno vybrat z katalogu, a dále vlastní text výměnného listu.

Tok událostí:

1.1.68 **Basic Path**

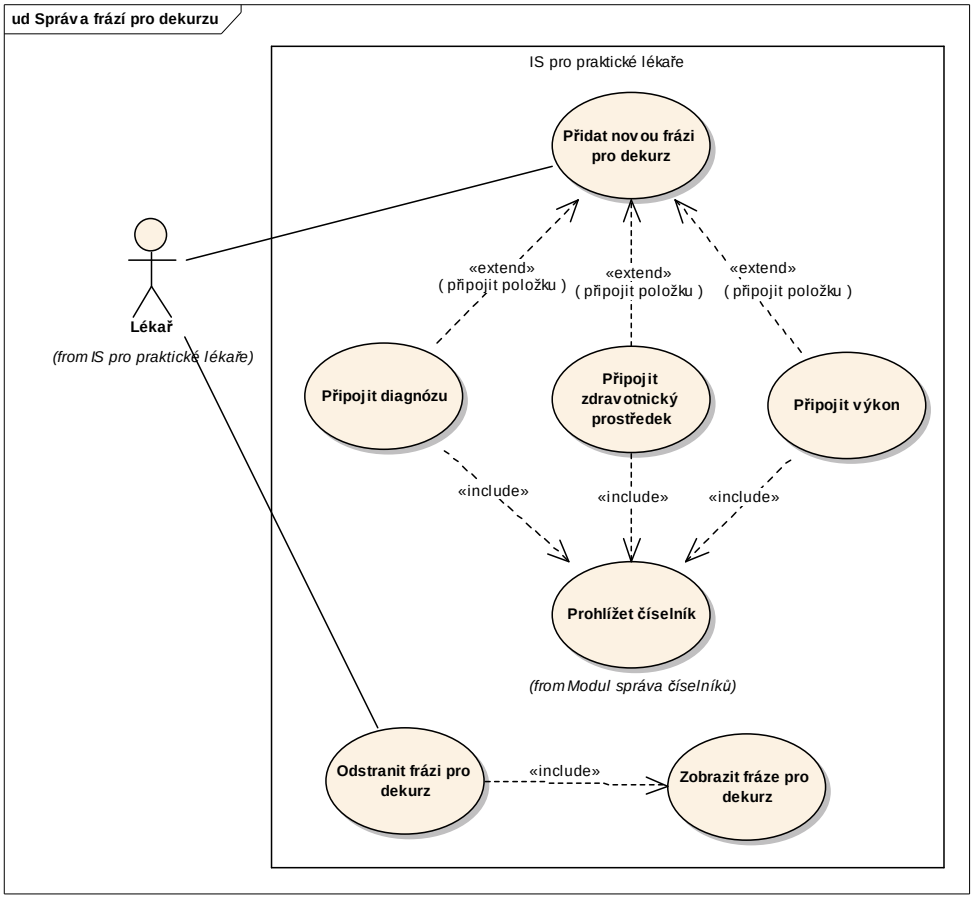
Vytvoření nového výměnného listu

1. Případ užití začíná, jestliže chce lékař pacientovi vystavit nový výměnný list.
2. Systém zobrazí formulář umožňující zadat: adresu lékaře pro kterého je výměnný list určen a vlastní text zprávy.
3. Lékař zadá požadované údaje.
<upravit zprávu>
4. Lékař potvrdí zadané údaje.
5. INCLUDE (Tisknout výměnný list).
6. Systém uloží vystavený výměnný list do seznamy výměnných listů.

16.Správa frází k dekurzu

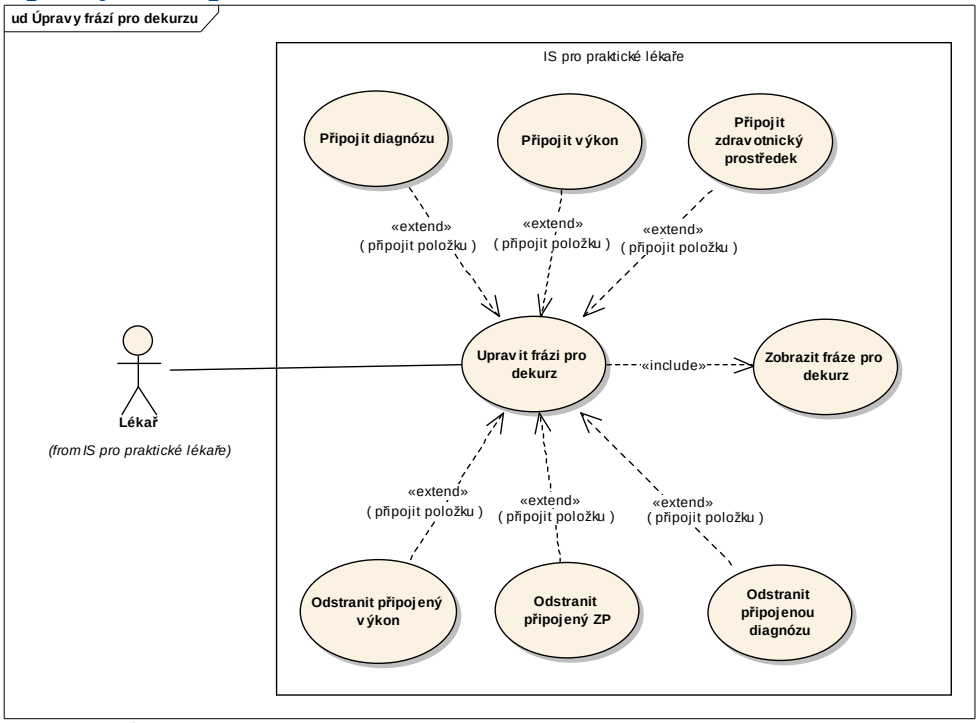
Balíček obsahuje případy užití související se správou frází pro dekurz, které může lékař využívat při zápisu průběhu návštěvy pacienta.

Správa frází pro dekurzu



Obrázek:: Správa frází pro dekurzu

Úpravy frází pro dekurzu



Obrázek:: Úpravy frází pro dekurzu

Odstranit frázi pro dekurz

Odstraní frázi pro dekurz.

Tok událostí:

1.1.69 **Basic Path**

Odstranění fráze pro dekurz

1. Případ užití začíná, jestliže se lékař rozhodne odstranit některou z frází pro dekurz.
2. INCLUDE (Zobrazit fráze pro dekurz).
3. Systém odstraní vybranou frázi pro dekurz.

Odstranit připojenou diagnózu

Odstraní připojenou diagnózu.

Tok událostí:

1.1.70 **Basic Path**

Odstranění připojené diagnózy

1. Případ užití začíná, jestliže chce lékař odstranit diagnózu připojenou k frázi pro dekurz.
2. Systém zobrazí seznam připojených diagnóz.
3. Lékař jednu ze zobrazených diagnóz vybere.
4. Systém odstraní vybranou diagnózu.

Odstranit připojený výkon

Odstraní výkon připojený k vybrané frázi pro dekurz.

Tok událostí:

1.1.71 **Basic Path**

Odstranění připojeného výkonu

1. Případ užití začíná, jestliže chce lékař odstranit výkon připojený k frázi pro dekurz.
2. Systém zobrazí seznam připojených výkonů.
3. Lékař jeden ze zobrazených výkonů vybere.
4. Systém odstraní vybraný výkon.

Odstranit připojený ZP

Odstraní připojený zdravotnický prostředek.

Tok událostí:

1.1.72 **Basic Path**

Odstranění připojeného ZP

1. Případ užití začíná, jestliže chce lékař odstranit zdravotnický prostředek připojený k frázi pro dekurz.
2. Systém zobrazí seznam připojených zdravotnických prostředků.
3. Lékař jeden ze zobrazených ZP vybere.
4. Systém odstraní vybraný ZP.

Přidat novou frázi pro dekurz

Přidá novou frázi pro dekurz.

Tok událostí:

1.1.73 **Basic Path**

Přidání nové fráze pro dekurz

1. Případ užití začíná, když chce lékař vytvořit novou frázi pro dekurz.
2. Systém zobrazí formulář umožňující zadat: název fráze a vlastní text.
3. Lékař vyplní požadované údaje.
<připojit položku>
4. Systém uloží frázi do systému.

Připojit diagnózu

Připojí k frázi pro dekurz diagnózu.

Tok událostí:

1.1.74 **Basic Path**

Připojení diagnózy

1. Případ užití začíná, jestliže se lékař rozhodne k frázi pro dekurz připojit diagnózu.
2. INCLUDE (Zobrazí číselník Diagnózy).
3. Systém připojí k frázi pro dekurz vybranou diagnózu.

Připojit výkon

Připojí k frázi pro dekurz výkon.

Tok událostí:

1.1.75 **Basic Path**

Připojení výkonu

1. Případ užití začíná, jestliže se lékař rozhodne k frázi pro dekurz připojit výkon.
2. INCLUDE (Zobrazí číselník Výkony).
3. Systém připojí k frázi pro dekurz vybraný výkon.

Připojit zdravotnický prostředek

Připojí k frázi pro dekurz zdravotnický prostředek.

Tok událostí:

1.1.76 **Basic Path**

Připojení zdravotnického prostředku

1. Případ užití začíná, jestliže se lékař rozhodne k frázi pro dekurz připojit ZP.
2. INCLUDE (Zobrazí číselník Zdravotnické prostředky).
3. Systém připojí k frázi pro dekurz vybraný ZP.

Upravit frázi pro dekurz

Upraví již existující frázi pro dekurz.

Tok událostí:

1.1.77 **Basic Path**

Upravení fráze pro dekurz

1. Případ užití začíná, když chce lékař změnit text některé z frází pro dekurz.
2. INCLUDE (Zobrazit fráze pro dekurz).
3. Systém zobrazí celý text vybrané fráze.
4. Lékař provede požadované změny.
<připojit položku>
5. Systém provedené změny uloží.

Zobrazit fráze pro dekurz

Zobrazí všechny fráze pro dekurz.

Tok událostí:

1.1.78 **Basic Path**

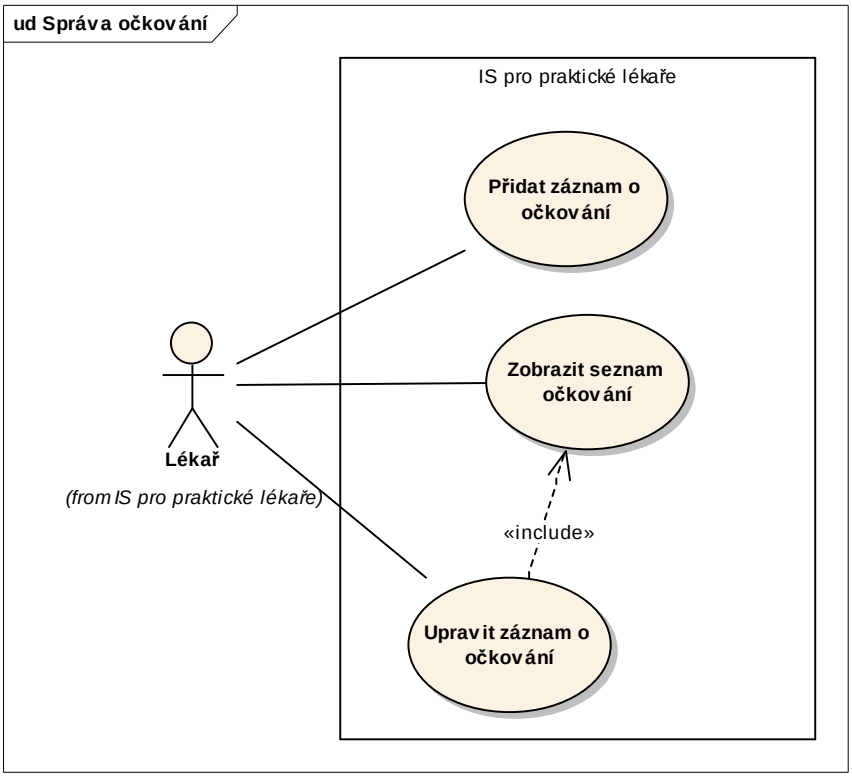
Zobrazení frází pro dekurz

1. Případ užití začíná, když uživatel chce vybrat frázi se seznamu frází pro dekurz.
2. Systém zobrazí seznam názvů frází pro dekurz.
3. Uživatel jednu z frází vybere.

17.Správa očkování

Balíček obsahuje případy užití související s evidencí provedených očkování.

Správa očkování



Obrázek:: Správa očkování

Přidat záznam o očkování

Přidá do seznamu očkování nový záznam.

Tok událostí:

1.1.79 **Basic Path**

Přidání záznamu o očkování

1. Případ užití začíná, když chce lékař zadat do systému nový záznam o provedeném očkování.
2. Systém zobrazí formulář umožňující zadat: datum, očkovací látku a poznámku.
3. Lékař vyplní požadované údaje.
4. Systém uloží nový záznam o očkování.

Upravit záznam o očkování

Umožňuje změnit údaje evidované v záznamu o očkování.

Tok událostí:

1.1.80 **Basic Path**

Upravení záznamu očkování

1. Případ užití začíná, když chce lékař upravit některý ze záznamů očkování pacienta.
2. Systém požádá lékaře o výběr záznamu očkování, který chce upravovat.
3. INCLUDE (Zobrazit seznam očkování).
4. Systém zobrazí formulář umožňující upravit veškeré údaje evidované u vybraného záznamu: datum, očkovací látku a poznámku.
5. Lékař upraví požadované údaje.
6. Systém uloží do záznamu všechny změny provedené lékařem.

Zobrazit seznam očkování

Zobrazí seznam všech očkování provedených u pacienta.

Tok událostí:

1.1.81 **Basic Path**

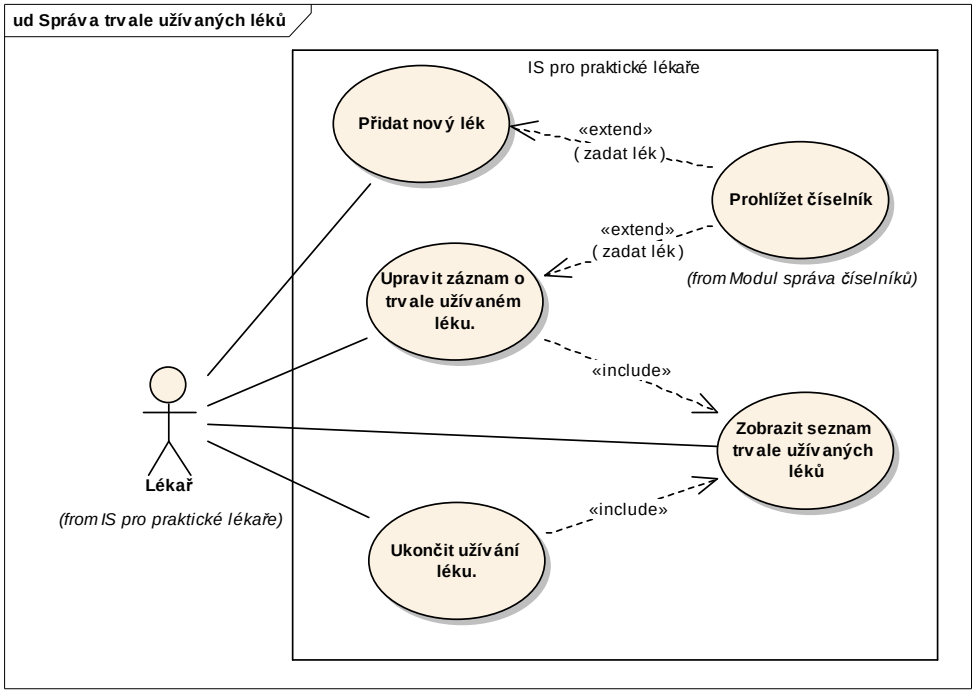
Zobrazení seznamu očkování

1. Případ užití začíná, když uživatel chce vybrat záznam o očkování, ze seznamu provedených očkování.
2. Systém zobrazí seznam provedených očkování. Každý záznam obsahuje: datum očkování, očkovací látku a poznámku.
3. Uživatel jeden ze záznamů vybere.

18.Správa trvale užívaných léků

Balíček obsahuje případy užití související s evidencí trvale užívaných léků pacientem.

Správa trvale užívaných léků



Obrázek:: Správa trvale užívaných léků

Přidat nový lék

Přidá na seznam trvale užívaných léků nový lék. U každého trvale užívaného léku je evidováno: kód léku, název léku, doplněk názvu, počet balení, dávkování, datum od, datum do a vlastní poznámku k danému léku.

Tok událostí:

1.1.82 **Basic Path**

Přidání nového léku

1. Případ užití začíná, když chce lékař zadat do systému nový záznam o trvale užívaném léku pacientem..
2. Systém zobrazí formulář umožňující zadat: kód léku, název léku, doplněk názvu, počet balení, dávkování, datum od, datum do a vlastní poznámku.
3. Lékař vyplní požadované údaje.
<zadat lék>Zobrazí číselník léků.
4. Systém uloží nový záznam o trvalém užívání léku pacientem.

Ukončit užívání léku.

Pro vybraný lék ukončí používání zaznamenáním aktuálního datumu do položky datum do. Lék zůstane v seznamu trvale užívaných léků, pouze je označen jako již neužívaný.

Tok událostí:

1.1.83 **Basic Path**

Ukončení užívání léku

1. Případ užití začíná, když chce lékař ukončit užívání léku pacientem.
2. Systém požádá o výběr záznamu léku, který má pacient přestat užívat.
3. INCLUDE (Zobrazit seznam trvale užívaných léků).
4. Systém požádá a vložení datumu ukončení užívání léku.
5. Systém uloží do vybraného záznamu trvale užívaného léku do položky "datum do" zadané datum.

Upravit záznam o trvale užívaném léku.

Umožňuje měnit veškeré údaje záznamu trvale užívaného léku.

Tok událostí:

1.1.84 **Basic Path**

Upravení záznamu o trvale užívaném léku

1. Případ užití začíná, když chce lékař upravit záznam o trvale užívaném léku pacientem.
2. Systém zobrazí formulář umožňující upravit všechny položky záznamu (kód léku, název léku, doplněk názvu, počet balení, dávkování, datum od, datum do a vlastní poznámku).
3. Lékař vyplní požadované údaje.
<zadat lék> Zobrazí číselník léků.
4. Systém uloží provedené změny záznamu.

Zobrazit seznam trvale užívaných léků

Zobrazí seznam všech trvale užívaných léků u vybraného pacienta. Léky, u kterých byla medikace ukončena zobrazí odlišným formátem popř. umožní skrytí takovýchto léků.

Tok událostí:

1.1.85 **Basic Path**

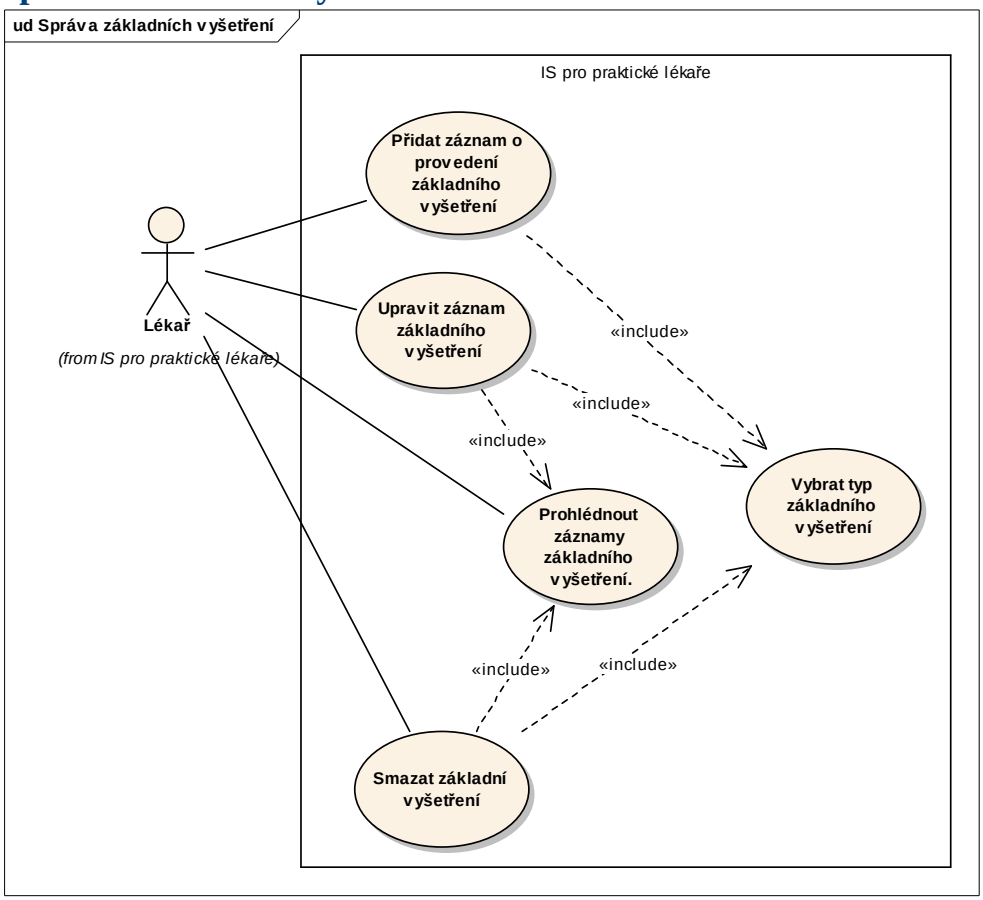
Zobrazení seznamu trvale užívaných léků

1. Případ užití začíná, když uživatel chce vybrat záznam ze seznamu trvale užívaných léků, které byly pacientovi předepsány.
2. Systém zobrazí seznam léků, které byly pacientovi předepsány.
3. Lékař vybere jeden ze zobrazených záznamů.

19.Správa základních vyšetření

Balíček obsahuje případy užití související s evidencí výsledků základních vyšetření. Jednotlivé typy základních vyšetření jsou: tlak a puls, hmotnost a výška, sedimentace.
Případy užití v tomto balíčku jsou shodné pro všechny typy základních vyšetření. Pouze se liší různými atributy, které jsou u jednotlivých základních vyšetření evidovány.

Správa základních vyšetření



Obrázek:: Správa základních vyšetření

Prohlédnout záznamy základního vyšetření.

Zobrazí seznam záznamů o provedení základního vyšetření. Umožňuje jeden ze zobrazených záznamů vybrat.

Tok událostí:

1.1.86 **Basic Path**

Prohlížení záznamů základního vyšetření.

1. Případ užití začíná, jestliže si lékař chce prohlédnout záznamy o provedených základních vyšetřeních.
2. Systém zobrazí všechny záznamy o provedení vybraného základního vyšetření.
3. Systém zobrazí graf zachycující vývoj hodnot základních vyšetření v čase.
4. Uživatel jeden ze záznamů vybere.

Přidat záznam o provedení základního vyšetření

Přidá záznam o provedení základního vyšetření pacienta. Zobrazí formulář umožňující zadat údaje evidované u zvoleného vyšetření.

Tok událostí:

1.1.87 **Basic Path**

Přidání záznamu o provedení vyšetření

1. Případ užití začíná, jestliže chce lékař zaznamenat výsledek provedeného základního vyšetření.
2. INCLUDE (Vybrat typ základního vyšetření).
3. Systém zobrazí formulář umožňující zadat všechny položky zvoleného základního vyšetření. Všechny typy vyšetření obsahují: datum a vlastní poznámku. Podle typu základního vyšetření dále obsahují:
Tlak a puls: diastolický tlak, systolický tlak, puls
Hmotnost a výška: hmotnost, výška
Sedimentace: hodnota sedimentace
4. Lékař vyplní požadované údaje.
5. Systém uloží zadané údaje.

Smazat základní vyšetření

Po potvrzení smaže záznam o provedení vybraného základního vyšetření.

Tok událostí:

1.1.88 **Basic Path**

Smazání záznamu o provedení základního vyšetření.

1. Případ užití začíná, jestliže se lékař rozhodne odstranit některý ze záznamů o provedení základního vyšetření.
2. INCLUDE (Vybrat typ základního vyšetření).
3. INCLUDE (Prohlédnout záznamy základního vyšetření).
4. Systém vymaže zvolený záznam o provedení základního vyšetření.

Upravit záznam základního vyšetření

Umožní upravit veškeré údaje evidovaného u zvoleného základního vyšetření.

Tok událostí:

1.1.89 **Basic Path**

Upravení záznamu základního vyšetření

1. Případ užití začíná, jestliže chce lékař změnit některý z údajů dříve provedeného základního vyšetření.
2. INCLUDE (Vybrat typ základního vyšetření).
3. INCLUDE (Prohlédnout záznamy základního vyšetření).
4. Systém zobrazí formulář umožňující změnit všechny údaje evidované u zvoleného základního vyšetření.
5. Lékař změní požadované údaje.
6. Systém uloží provedené změny.

Vybrat typ základního vyšetření

Umožňuje vybrat typ základního vyšetření (Tlak a puls, Hmotnost a výška, Sedimentace).

Tok událostí:

1.1.90 **Basic Path**

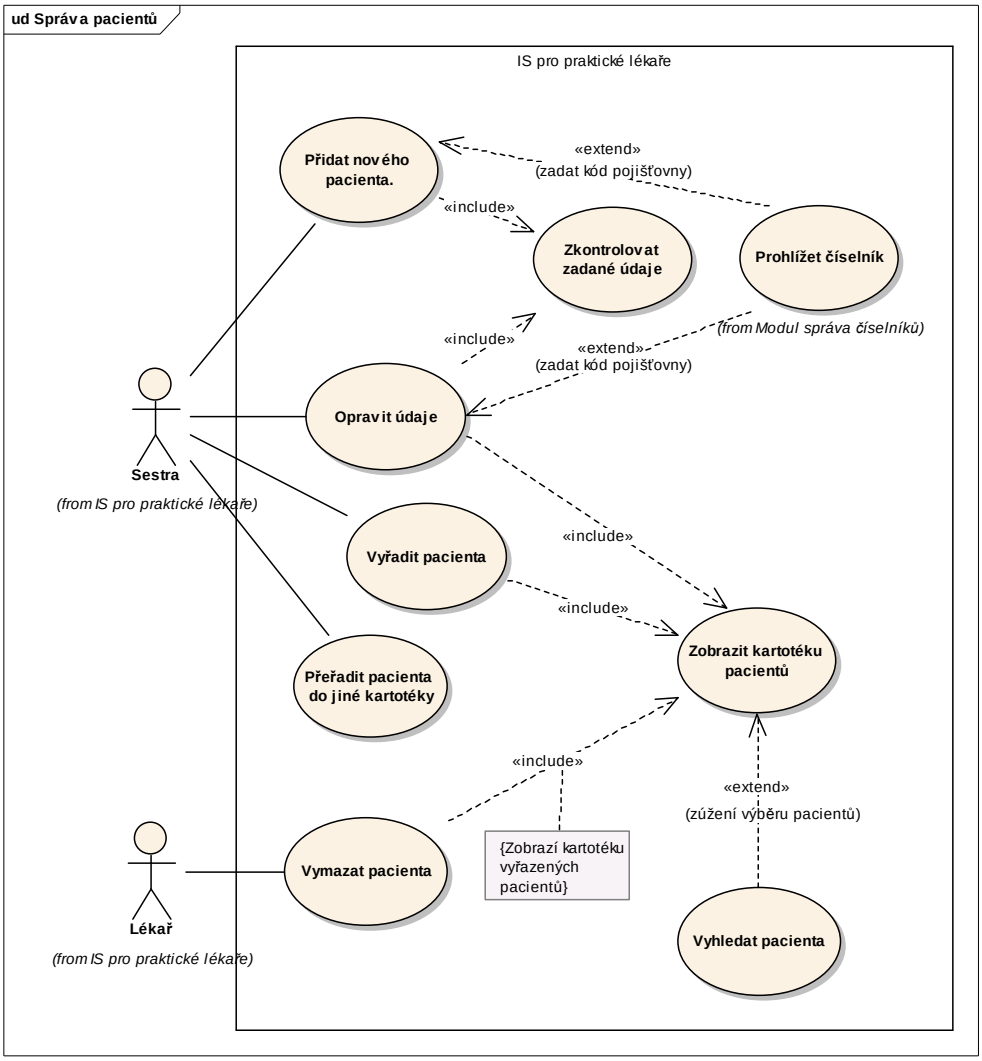
Výběr typu základního vyšetření

1. Případ užití začíná, jestliže uživatel chce zvolit typ základního vyšetření.
2. Systém zobrazí nabídku se seznamem typů základních vyšetření: Tlak a puls, Hmotnost a výška, Sedimentace.
3. Uživatel jeden z typů vybere.

20.Modul správa pacientů

Balíček obsahuje případy užití související s evidencí pacientů. Každý evidovaný pacient je zařazen právě v jedné kartotéce. Každý pacient může být členem několika skupin vytvářených lékařem.
Pro každou ordinaci evidovanou v rámci systému je vytvářena samostatná kartotéka pacientů. Dále existuje další samostatná kartotéka pacientů nepravidelné péče společná pro všechny ordinace v rámci systému.

Správa pacientů



Obrázek:: Správa pacientů

Opravit údaje

Umožňuje měnit veškeré údaje, které jsou u pacienta evidovány.

Tok událostí:

1.1.91 **Basic Path**

Oprava údajů o pacientovi

1. Případ užití začíná, jestliže uživatel chce změnit některé údaje o pacientovi.
2. Systém zobrazí formulář umožňující změnit veškeré údaje evidované u pacienta.
3. Uživatel provede změnu požadovaných údajů.
<zadat kód pojišťovny>
4. Uživatel potvrdí provedené změny.
5. INCLUDE (Zkontrolovat údaje)
5. Systém uloží nové údaje o pacientovi.

Přeřadit pacienta do jiné kartotéky

Umožní přesunout pacienta do jiné kartotéky evidované v rámci systému.

Tok událostí:

1.1.92 **Basic Path**

Přeřazení pacienta do jiné kartotéky

1. Případ užití začíná, když uživatel zadá příkaz Přeřadit pacienta do jiné kartotéky.
2. INCLUDE (Zobrazit kartotéku pacientů).
3. Systém zobrazí seznam kartoték. Zobrazené kartotéky budou: kartotéky jednotlivých ordinací evidovaných v rámci systému a kartotéka pacientů nepravidelné péče.
4. Uživatel vybere kartotéku, do které má být pacient přeřazen.
5. Systém přeřadí pacienta do nové kartotéky.

Přidat nového pacienta.

Přidá nového pacienta do kartotéky vybrané ordinace nebo do kartotéky nepravidelné péče.Kartotéka ordinace, do které má být pacient zařazen je dána přihlášením lékaře.

Tok událostí:

1.1.93 **Basic Path**

Přidání nového pacienta

1. Případ užití začíná, když uživatel zadá příkaz "přidat nového pacienta".
2. Systém zobrazí formulář umožňující zadat do systému informace o pacientovi. Povinně zadávané údaje jsou: jméno, příjmení, rodné číslo, číselný kód pojišťovny u které je pacient pojištěn. Nepovinné údaje jsou: titul, pohlaví, krevní skupina, státní občanství, stav, adresa bydliště, telefon1, telefon2, e-mail, povolání, zaměstnavatel, adresa zaměstnavatele, telefon zaměstnavatele, jméno otce, jméno matky, povolání otce, povolání matky, zaměstnavatel otce, zaměstnavatel matky, adresa bydliště otce, adresa bydliště matky, telefon otce, telefon matky, další poznámky.
3. Uživatel vyplní všechny povinné údaje.
<zadat kód pojišťovny>
4. Uživatel potvrdí zadané údaje.
5. INCLUDE (Zkontrolovat zadané údaje).
6. Systém požádá o výběr kartotéky, do které má být pacient zařazen (kartotéka aktuální ordinace nebo kartotéka nepravidelné péče).
7. Systém vloží nového pacienta do zvolené kartotéky a uloží k jeho záznamu všechny zadané údaje.

Tok událostí:

1.1.94 **Alternate**

Zrušení přidávání nového pacienta

1. Případ užití začíná, jestliže uživatel stiskne klávesu Esc nebo uzavře dialogové okno během vyplňování údajů o novém pacientovi.
2. Jestliže již byly nějaké údaje o pacientovi zadány, pak:
   2.1 Systém zobrazí upozornění, zda chce uživatel zadané údaje nejprve uložit nebo zda mají být zadané údaje zahozeny.
   2.2. Jestliže se rozhodne zadané údaje nejprve uložit,pak:
      2.2.1 Případ užití pokračuje ve 4. bodě případu užití Přidat nového pacienta.

Vyhledat pacienta

Umožňuje vyhledat pacienta podle rodného čísla nebo jména.

Tok událostí:

1.1.95 **Basic Path**

Vyhledání pacienta

1. Případ užití začíná, když chce uživatel vyhledat pacienta na základě jeho rodného čísla nebo jména.
2. Systém zobrazí formulář pro zadání rodného čísla nebo jména.
3. Uživatel zadá údaje pro vyhledávání.
4. Systém zobrazí v kartotéce pouze pacienty, kteří vyhovují zadaným kritériím.

Tok událostí:

1.1.96 **Alternate**

Pacient nenalezen

1. Případ užití začíná ve 4. kroku případu užití Vyhledat pacienta, jestliže zadaným kritériím neodpovídá žádný pacient.
2. Systém upozorní lékaře, že žádný pacient neodpovídá vyhledávacím kritériím.

Vymazat pacienta

Vyprší-li zákonem daná lhůta pro evidenci lékařské dokumentace vyřazeného pacienta, lze ho odstranit trvale i z kartotéky vyřazených pacientů.

Tok událostí:

1.1.97 **Basic Path**

Vymazání pacienta z evidence

1. Případ užití začíná, když lékař zadá příkaz Vymazat pacienta z evidence.
2. INCLUDE (Zobrazit kartotéku pacientů). Zobrazí kartotéku vyřazených pacientů.
3. Systém zkontroluje, zda již vypršela zákonem daná doba pro uchování lékařské dokumentace vyřazeného pacienta.
4. Jestliže tato doba již vypršela, pak:
   4.1 Systém odstraní veškerou dokumentaci vztahující se k vybranému pacientovi.
5. Jinak systém zobrazí informaci o nutnosti nadále uchovávat dokumentaci vybraného pacienta.

Vyřadit pacienta

Přesune pacienta do kartotéky vyřazených pacientů. Pacient zůstává dále evidován z důvodu uchování jeho zdravotní dokumentace.

Tok událostí:

1.1.98 **Basic Path**

Vyřazení pacienta

1. Případ užití začíná, když uživatel zadá příkaz Vyřadit pacienta.
2. INCLUDE (Zobrazit kartotéku pacientů).
3. Systém vybraného pacienta přesune do kartotéky vyřazených pacientů.
4. Systém zaznamená datum vyřazení pacienta z evidence.

Zkontrolovat zadané údaje

Provede kontrolu, zda jsou zadány všechny povinné údaje.

Tok událostí:

1.1.99 **Basic Path**

Kontrola zadaných údajů

1. Případ užití začíná při registraci nového pacienta nebo při úpravách základních informací o pacientovi.
2. Systém zkontroluje, zda byly zadány všechny povinné údaje (jméno, příjmení, rodné číslo, kód pojišťovny).
3. Systém zkontroluje, zda zadané rodné číslo je platné.
4. Jestliže nejsou všechny povinné údaje zadány, pak:
   4.1 Systém zobrazí informaci se seznamem údajů, které lékař nevyplnil a jsou povinné pro přidání nového pacienta.
5. Jestliže zadané rodné číslo není platné, pak:
   5.1 Systém upozorní lékaře na neplatnost zadaného rodného čísla.

Zobrazit kartotéku pacientů

Zobrazí kartotéku pacientů. Umožňuje výběr způsobu řazení pacientů (vzestupně, sestupně, podle rodného čísla, podle jména). Umožňuje určit skupiny pacientů, které mají být zobrazeny (pouze z této ordinace, nepravidelná péče, vyřazení pacienti, všichni pacienti, či jinou zvolenou skupinu, kterou si lékař sám vytvořil).

Tok událostí:

1.1.100 **Basic Path**

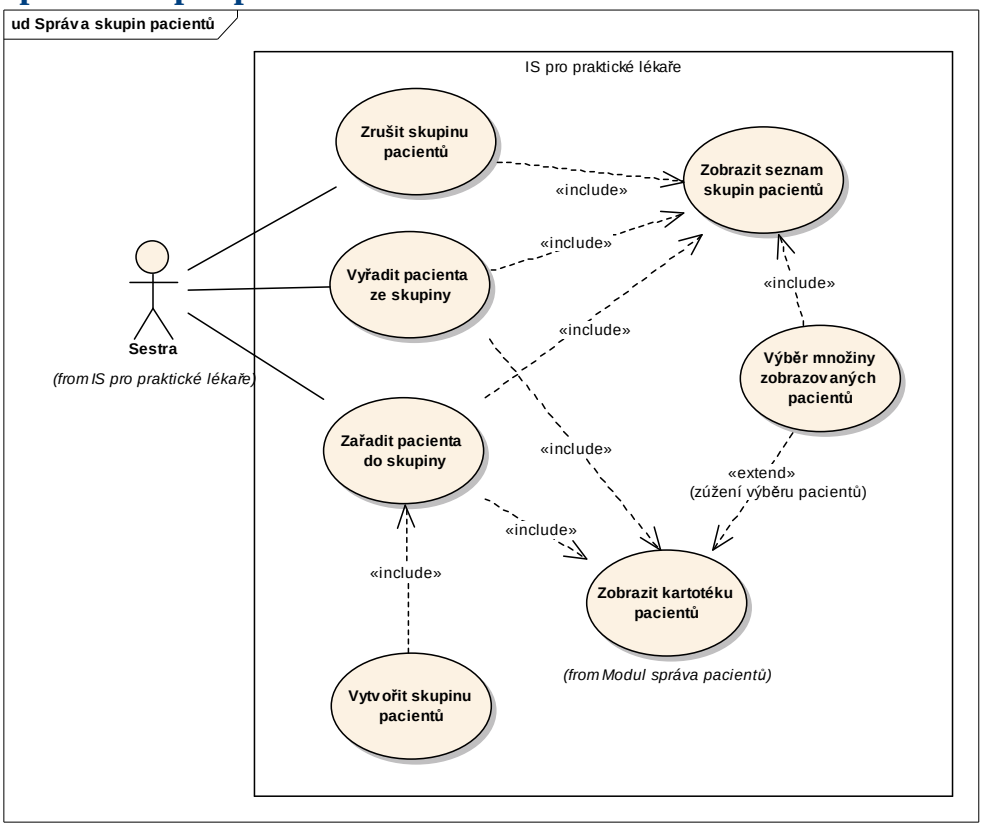
Zobrazení kartotéky pacientů

1. Případ užití začíná, když lékař zadá příkaz "Zobrazit kartotéku pacientů".
2. Systém zobrazí seznam všech pacientů ve vybrané skupině abecedně seřazený. Zobrazí jejich rodná čísla a jména.
<zúžení výběru pacientů>
3. Lékař vybere ze seznamu pacientů jednoho pacienta.

21.Správa skupin pacientů

Balíček obsahuje případy užití související s vytvářením skupin pacientů tak, aby bylo pro lékaře snazší vyhledávat pacienty, kteří mají nějaké vlastnosti společné (patří do stejné skupiny).

Správa skupin pacientů



Obrázek:: Správa skupin pacientů

Výběr množiny zobrazovaných pacientů

Umožní vybrat množinu pacientů, která má být v kartotéce zobrazena. Zobrazit lze: všechny pacienty, pouze pacienty z kartotéky aktivní ordinace, pacienty z kartotéky nepravidelná péče, vyřazené pacienty. Dále je možné zobrazovat pouze pacienty, kteří jsou členy vybrané skupiny nadefinované lékařem.

Tok událostí:

1.1.101 **Basic Path**

Výběr množiny zobrazených pacientů

1. Případ užití začíná, jestliže chce uživatel omezit množství zobrazovaných pacientů.
2. Systém zobrazí předvolené množiny pacientů, které lze vybrat: všichni pacienti, pacienti z kartotéky aktivní ordinace, pacienti z kartotéky nepravidelné péče, vyřazení pacienti a pacienti ve skupině.
3. Lékař vybere množinu pacientů, kterou chce v kartotéce zobrazit.
4. Vybere-li možnost zobrazit pacienty ve skupině, pak:
   4.1 Systém požádá o výběr skupiny, jejíž členové mají být zobrazeni.
   4.2 INCLUDE (Zobrazit seznam skupin pacientů)
5. Systém zobrazí v kartotéce pouze vybranou množinu pacientů.

Vytvořit skupinu pacientů

Vytvoří novou skupinu pacientů, kterou lékař může využít pro rozčlenění pacientů.

Tok událostí:

1.1.102 **Basic Path**

Vytvoření nové skupiny

1. Případ užití začíná, když uživatel zadá příkaz Vytvořit novou skupinu.
2. Systém zobrazí formulář pro zadání jména nové skupiny.
3. Uživatel zadá název nové skupiny.
4. Systém vytvoří novou skupinu.
5. INCLUDE (Zařadit pacienta do skupiny).

Vyřadit pacienta ze skupiny

Vyřadí vybrané pacienty ze skupiny.

Tok událostí:

1.1.103 **Basic Path**

Vyřazení pacienta ze skupiny

1. Případ užití zažíná, když uživatel zadá příkaz Vyřadit pacienta ze skupiny.
2. Systém požádá o výběr skupiny, ze které mají být pacienti vyřazeni.
3. INCLUDE (Zobrazit seznam skupin pacientů).
4. Systém požádá o výběr pacientů, kterým má členství ve skupině zrušit.
3. INCLUDE (Zobrazit kartotéku pacientů). Zobrazí pacienty, kteří jsou zařazeni ve vybrané skupině. Umožňuje vybrat více pacientů současně.
4. Systém zruší členství vybraných pacientů ve zvolené skupině.

Zařadit pacienta do skupiny

Zařadí vybrané pacienty do zvolené skupiny.

Tok událostí:

1.1.104 **Basic Path**

Zařazení pacientů do skupiny

1. Případ užití začíná, jestliže chce uživatel zařadit pacienta do skupiny.
2. Systém požádá o výběr skupiny, do které chce uživatel přidat nové pacienty.
3. INCLUDE (Zobrazit seznam skupin pacientů).
4. Systém požádá o výběr pacientů, kteří mají být do kartotéky přidány.
5. INCLUDE (Zobrazit kartotéku pacientů). Umožňuje vybrat více pacientů současně.
5. Systém zařadí vybrané pacienty do zvolené skupiny.

Zobrazit seznam skupin pacientů

Zobrazí všechny skupiny vytvořené uživatelem. Umožní jednu z nich vybrat.

Tok událostí:

1.1.105 **Basic Path**

Zobrazení seznamu skupin pacientů

1. Případ užití začíná, když uživatel chce vybrat skupinu pacientů ze seznamu skupin evidovaných v systému.
2. Systém zobrazí seznam skupin pacientů, které jsou v systému evidovány.
3. Uživatel jednu ze skupin vybere.

Zrušit skupinu pacientů

Zruší skupinu pacientů - pacienti zůstávají registrováni pouze se ruší jejich členství v této skupině.

Tok událostí:

1.1.106 **Basic Path**

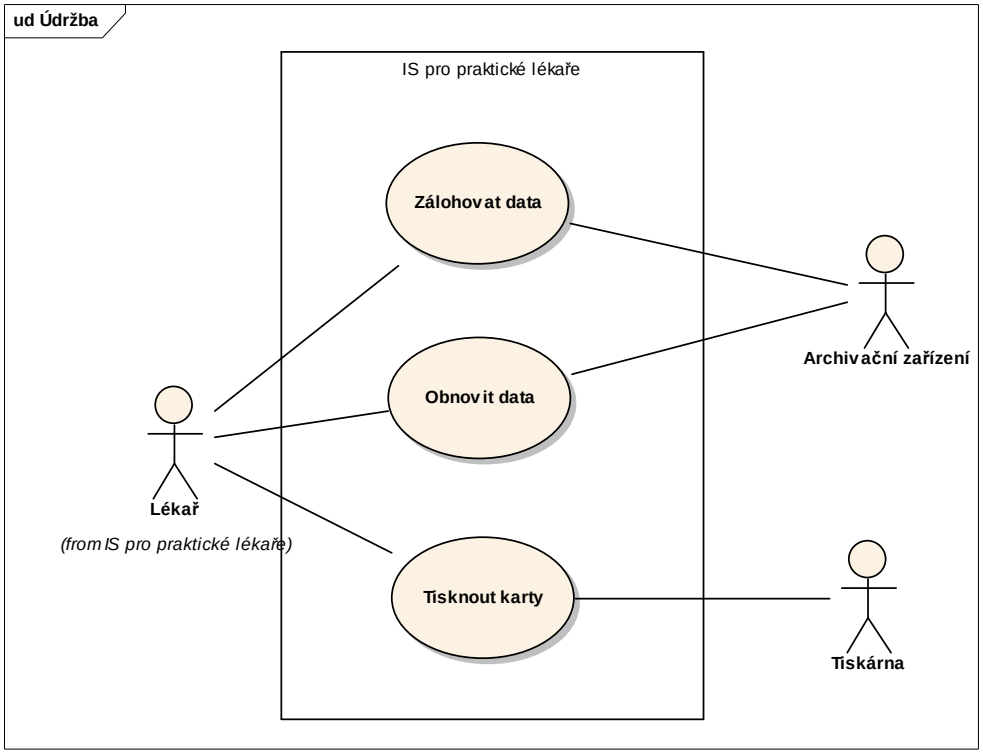
Zrušení skupiny pacientů

1. Případ užití začíná, když uživatel chce zrušit skupinu pacientů.
2. INCLUDE (Zobrazit seznam skupin pacientů).
3. Systém smaže vybranou skupinu. Všem pacientům, kteří jsou ve vybrané skupině, zruší jejich členství v této skupině.

22.Modul údržba dat

Balíček obsahuje případy užití související s archivací pořízených dat v elektronické a papírové podobě.

Údržba



Obrázek:: Údržba

Obnovit data

Umožní obnovit pořízená data ze zálohy.

Tok událostí:

1.1.107 **Basic Path**

Obnovení dat

1. Případ užití začíná, když se lékař rozhodne obnovit pořízená data ze zálohy.
2. Systém požádá lékaře o zadání souboru se zálohou dat.
3. Systém načte všechna pořízená data ze zálohy.

Tisknout karty

Umožňuje převést elektronicky pořízená data do tištěné podoby. Systém si musí uchovávat informaci o poloze naposledy vytištěného záznamu v kartě každého pacienta, aby bylo možné pokračovat s tiskem na stejném listu.

Tok událostí:

1.1.108 **Basic Path**

Tisknutí karet

1. Případ užití začíná, když se lékař rozhodne převést pořízená data do tištěné podoby.
2. Systém požádá lékaře o výběr způsobu tisku: Navázat na předchozí tisk, vytisknout od určitého datumu nebo tisk celé dokumentace.

Zálohovat data

Umožní zálohovat veškerá pořízená data.

Tok událostí:

1.1.109 **Basic Path**

Zálohování dat

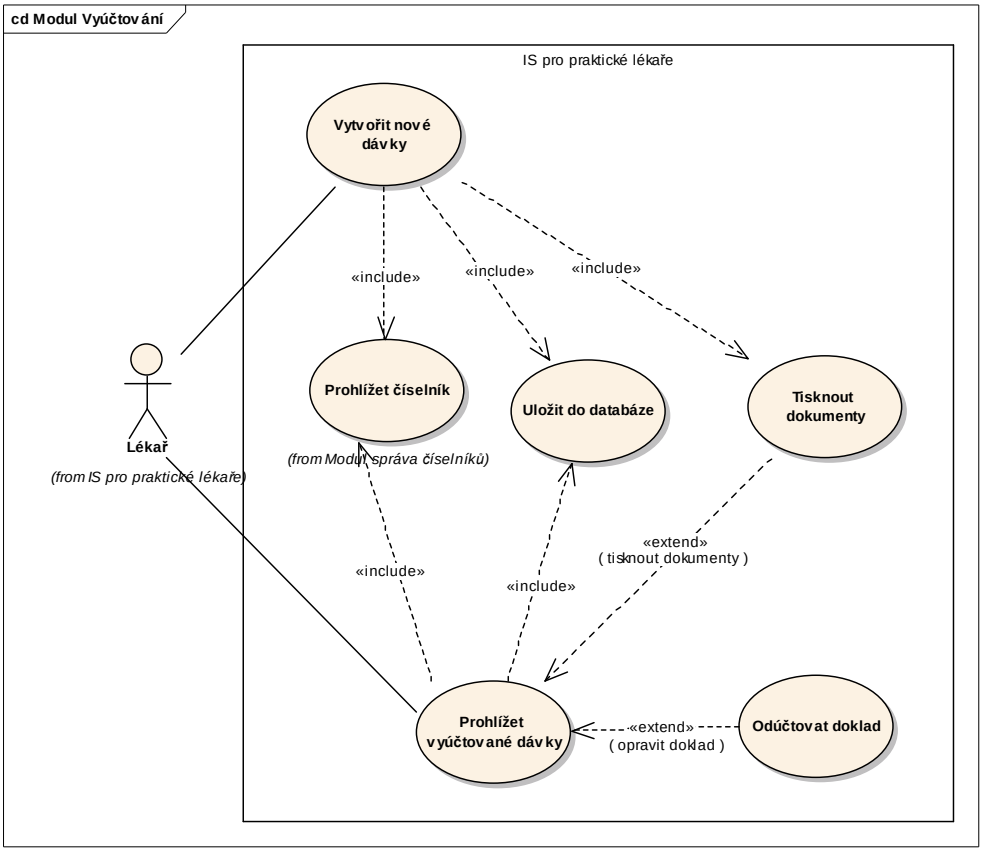
1. Případ užití začíná, když se lékař rozhodne zálohovat pořízená data.
2. Systém požádá lékaře o zadání místa, kam má být záloha umístěna.
3. Systém provede zálohu všech pořízených dat na vybrané místo.

23.Modul vyúčtování

V tomto balíčku jsou zahrnuty případy užití související s vykazováním provedené péče pojišťovnám a vystavováním potřebných dokladů. Dávky dokladů, které bude systém generovat:

05 - Dávka vyúčtování výkonů nepravidelné péče a LSPP
98 - Dávka ambulantní smíšená
80 - Dávka pro registraci pojištěnců

Modul Vyúčtování



Obrázek:: Modul Vyúčtování

Odúčtovat doklad

Umožní odúčtovat záznamy, na které byl doklad vystaven tak, aby byly příště opět zahrnuty do vyúčtování.

Tok událostí:

1.1.110 **Basic Path**

Odúčtování dokladu

1. Případ užití začíná, když se lékař rozhodne odúčtovat některý dokladu.
2. Systém označí záznamy, na které byl doklad vystaven, jako odmítnuté tak, aby byly při příštím vyúčtování zahrnuty do opravné dávky.
3. Původní verzi dokladu systém nechá v dávce uloženou.

Prohlížet vyúčtované dávky

Zobrazí seznam všech vyúčtovaných dávek pro vybranou pojišťovnu.

Tok událostí:

1.1.111 **Basic Path**

Prohlížení vyúčtování dávek

1. Případ užití začíná, když si chce lékař prohlédnout vyúčtované dávky.
2. INCLUDE (Prohlížet číselník) Zobrazí číselník pojišťoven. Umožní vybrat množinu pojišťoven.
3. Systém zobrazí seznam generovaných dávek pro vybrané pojišťovny.
4. Lékař vybere dávku, kterou chce prohlédnout.
<tisknout dokumenty>
5. Systém zobrazí seznam dokladů obsažených ve vybrané dávce. Umožní lékaři jeden z dokladů vybrat a zobrazit jeho jednotlivé položky.
<opravit doklad>
6. Lékař může označit vybranou dávku jako proplacenou nebo neproplacenou.
7. Jestliže lékař změní stav dávky (proplacená, neproplacená) pak:
   7.1 INCLUDE (Uložit do databáze)

Tisknout dokumenty

Umožňuje vytisknout na tiskárně potřebné dokumenty. Faktura za dávky se nejčastěji používá při opravném účtování dokladů, které byly při předchozím zpracování pojišťovnou odmítnuty.

Tok událostí:

1.1.112 **Basic Path**

Tisknutí dokumentů k dávkám

1. Případ užití začíná, jestliže chce lékař provést vytištění všech potřebných dokumentů k vygenerovaným dávkám.
2. Systém vytiskne průvodní list ke každé dávce.
3. Systém vytiskne průvodní list ke každé disketě.
4. Systém požádá lékaře o výběr faktury, která má být vytištěna (Faktura za období nebo Faktura za dávky).
5. Lékař vybere požadovaný typ faktury.
6. Systém vytiskne požadovaný typ faktury.

Příloha:
D:\Dokumenty\Diplomka\VZP\Tiskopisy\PruvodniListDavky.pdf
Ukázka tiskopisu průvodního listu dávky dodávaného VZP.
Příloha:
D:\Dokumenty\Diplomka\VZP\Tiskopisy\PruvodniListDiskety.pdf
Ukázka tiskopisu průvodního listu diskety dodávaného VZP.
Příloha:
D:\Dokumenty\Diplomka\VZP\Tiskopisy\FakturaZaDavky.pdf
Ukázka tiskopisu Faktura za dávky dodávaného VZP.
Příloha:
D:\Dokumenty\Diplomka\VZP\Tiskopisy\FakturaZaObdobi.pdf
Ukázka tiskopisu Faktura za období dodávaného VZP.

Uložit do databáze

Uloží generované dávky do databáze.

Tok událostí:

1.1.113 **Basic Path**

Uložení generovaných dávek do databáze

1. Případ užití začíná, jestliže chce lékař uložit do databáze vygenerované dávky.
2. Systém uloží vygenerované dávky do databáze.
3. Systém uloží veškeré vytištěné dokumenty do systému tak, aby bylo možné tyto dokumenty opětovně vytisknout.

Vytvořit nové dávky

Generovaný soubor má standardní název KDAVKA.XXX, kde XXX představuje kód pojišťovny z číselníku Zdravotní pojišťovny. Soubor obsahuje buď jednu dávku nebo několik dávek řazených za sebou. Na každé disketě předávané pojišťovně smí být vždy pouze jeden soubor.

Tok událostí:

1.1.114 **Basic Path**

Vyúčtování dávek

1. Případ užití začíná, když lékař zadá příkaz Spustit vyúčtování pro pojišťovny.
2. Systém požádá lékaře o zadání pojišťoven, pro které má být vyúčtování provedeno.
3. INCLUDE (Prohlížet číselník) Zobrazí číselník pojišťoven umožní vybrat množinu pojišťoven, pro které má být vyúčtování provedeno.
4. Systém požádá o zadání období, za které má být vyúčtování provedeno. Systém automaticky před vyplní datum začátku a konce posledního ukončeného měsíce.
6. Systém vytvoří soubory dávek obsahující jednotlivé doklady podle požadavků pojišťovny. Přesný popis datového rozhraní je v příloze.
7. Systém požádá uživatele o zadání umístění, kam mají být soubory uloženy. (Implicitně disketa).
8. Systém uloží soubory na zadané místo.
9. INCLUDE (Tisknout dokument)
10. INCLUDE (Uložit do databáze)

Příloha:
D:\Dokumenty\Diplomka\VZP\DRDokladu.pdf
Popis datového rozhraní dávek dokladů pro pojišťovnu.

Příloha:
D:\Dokumenty\Diplomka\VZP\DRDokladuZmeny.pdf
Popis změn datového rozhraní dávek dokladů pro pojišťovnu.

Příloha:
D:\Dokumenty\Diplomka\VZP\DRDokladuPripravovaneZmeny.pdf
Popis připravovaných změn v datovém rozhraní dávek dokladů pro pojišťovnu.

Příloha:
D:\Dokumenty\Diplomka\VZP\Metodika.pdf
Metodika pro pořizování a předávání dokladů VZP.

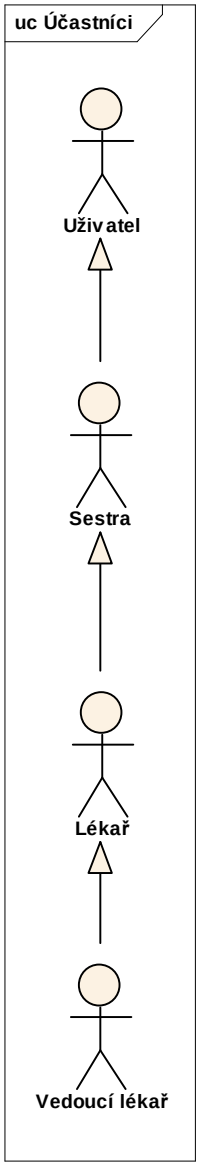
Příloha:
D:\Dokumenty\Diplomka\VZP\MetodikaZmeny.pdf
Změny v metodice v pořizování a předávání dokladů VZP.

Ukázka stručného popisu případů použití a účastníků - IS pro praktické lékaře

aneb pomocí "PACKAGE" strukturované případy použití(bez popisu toku událostí) systému.

Vytvořil J. Mlejnek. Pro potřeby předmětu X36SIN upravil M. Komárek.

Účastníci



Lékař

public Actor: Uživatel s rolí lékař může:

1. Provádět veškeré záznamy související s vyšetřením pacienta.
2. Měnit veškeré údaje evidované u pacienta.
3. Vystavovat pacientovi libovolné doklady.
4. Dále může využívat veškeré funkce, které využívá uživatel v roli sestra.

Sestra

public Actor: Uživatel s touto rolí může:

1. Registrovat nové pacienty.
2. Provádět veškeré úpravy související s evidencí pacienta v ordinaci nikoliv však s jeho zdravotní dokumetací.
3. Prohlížet zdravotní dokumentaci pacienta.

Uživatel

public Actor: Uživatel s touto rolí se může pouze využít službu přihlášení do systému, která mu při zadání správného jména a hesla povolí přístup do IS s novou rolí odpovídající zadanému uživatelskému jménu.

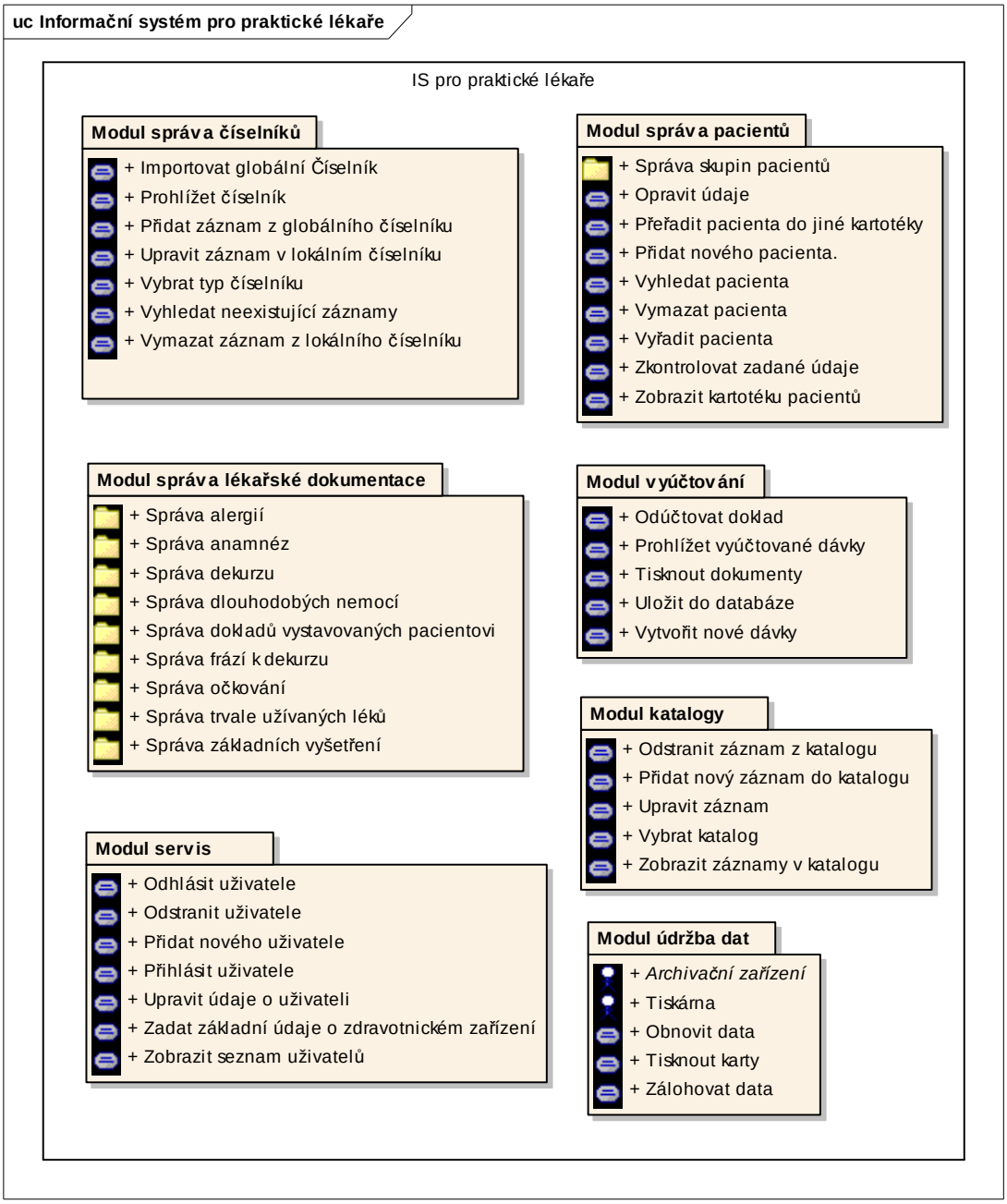
Vedoucí lékař

public Actor: Uživatel s rolí vedoucí lékař může:

1. Nastavovat základní informace o zdravotnickém zařízení, které systém využívá.
2. Přidávat či odstraňovat uživatele, kteří mohou IS využívat.
3. Nastavovat role uživatelů v jakých budou při práci s IS vystupovat.
4. Provádět vyúčtování lékařské péče pojišťovně.

Případy použití

Hlavní balíček obsahující všechny případy užití informačního systému.



Obrázek 1 : Informační systém pro praktické lékaře

Lékař

public Actor: Uživatel s rolí lékař může:

1. Provádět veškeré záznamy související s vyšetřením pacienta.
2. Měnit veškeré údaje evidované u pacienta.
3. Vystavovat pacientovi libovolné doklady.
4. Dále může využívat veškeré funkce, které využívá uživatel v roli sestra.

Sestra

public Actor: Uživatel s touto rolí může:

1. Registrovat nové pacienty.
2. Provádět veškeré úpravy související s evidencí pacienta v ordinaci nikoliv však s jeho zdravotní dokumetací.
3. Prohlížet zdravotní dokumentaci pacienta.

Uživatel

public Actor: Uživatel s touto rolí se může pouze využít službu přihlášení do systému, která mu při zadání správného jména a hesla povolí přístup do IS s novou rolí odpovídající zadanému uživatelskému jménu.

Vedoucí lékař

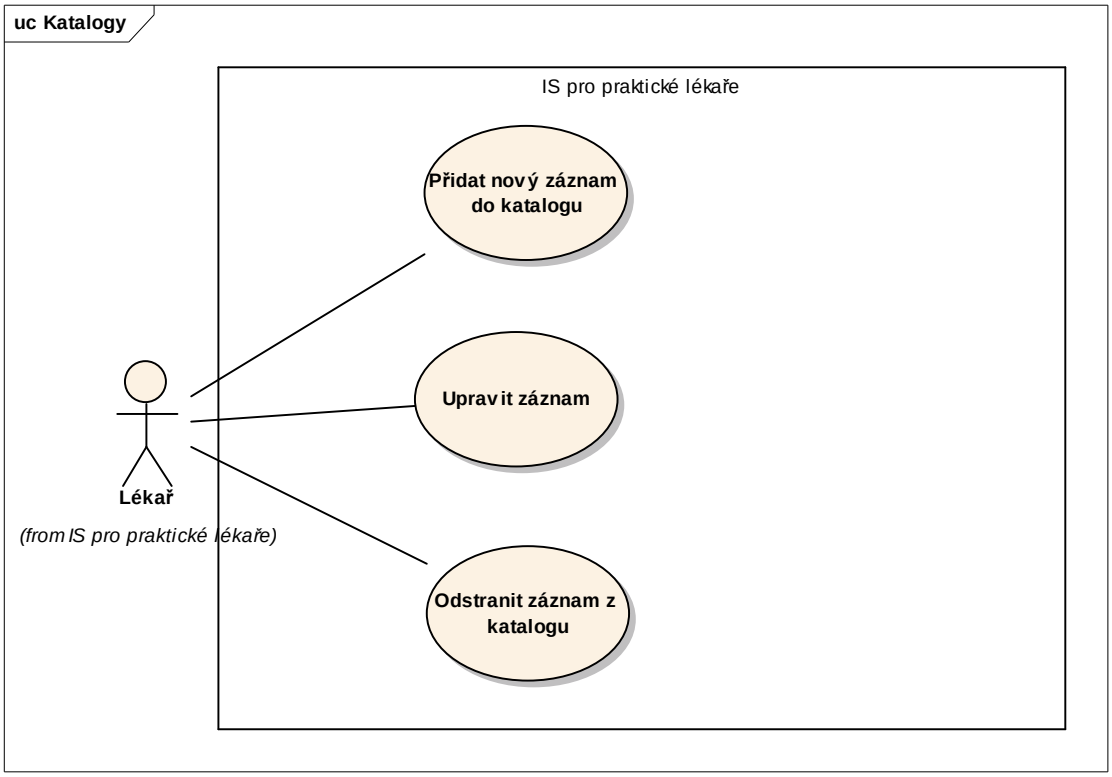
public Actor: Uživatel s rolí vedoucí lékař může:

1. Nastavovat základní informace o zdravotnickém zařízení, které systém využívá.
2. Přidávat či odstraňovat uživatele, kteří mohou IS využívat.
3. Nastavovat role uživatelů v jakých budou při práci s IS vystupovat.
4. Provádět vyúčtování lékařské péče pojišťovně.

Modul katalogy

Případy užití v tomto balíčku jsou shodné pro všechny katalogy v systému a slouží pro rychlejší zadávání opakujících se údajů. V systému budou evidovány katalogy: Adresy, Spolupracující lékaři, Magistraliter. Katalogy mohou být prázdné, veškeré záznamy v katalogu si musí uživatel sám nadefinovat.
Záznam v katalogu bude obsahovat následující položky:

1. Magistraliter - kód, název, doplatek, počet dávek, popis.
2. Spolupracující lékaři - příjmení, jméno, odbornost, IČZ.
3 . Adresy - jméno, ulice, číslo popisné, obec, PSČ.



Obrázek 2 : Katalogy

Odstranit záznam z katalogu

public Případ užití: Odstraní vybraný záznam z katalogu.

Přidat nový záznam do katalogu

public Případ užití: Zobrazí formulář umožňující zadat všechny položky záznamu daného katalogu. Přidá nový záznam do katalogu.

Upravit záznam

public Případ užití: Umožňuje upravit jednotlivé položky u vybraného záznamu v katalogu. Seznam položek jednotlivých záznamů v katalogu je uveden v popisu tohoto balíčku.

Vybrat katalog

public Případ užití: Umožňuje lékaři vybrat katalog, se kterým chce pracovat.

Zobrazit záznamy v katalogu

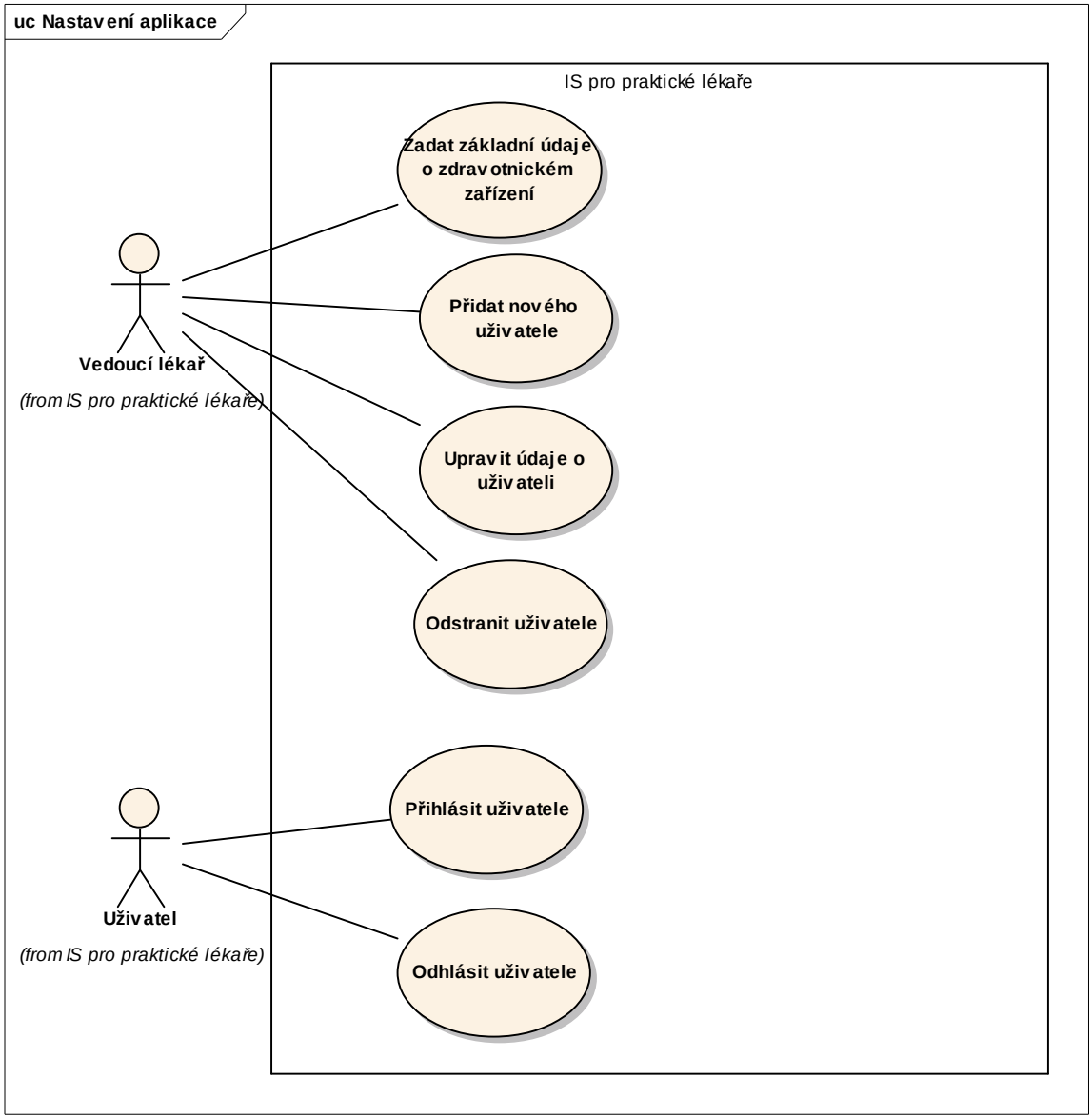
public Případ užití: Zobrazí jednotlivé záznamy v katalogu. Umožňuje je seřadit podle vybrané položky. Dále umožňuje jeden ze zobrazených záznamů vybrat. Položky, které se budou u jednotlivých katalogů zobrazovat:

Magistraliter - kód, název,

Spolupracující lékaři - IČZ, příjmení, jméno,
Adresy - jméno, obec.

Modul servis

Balíček obsahuje případy užití související s nastavením aplikace a správou uživatelů.



Obrázek 3 : Nastavení aplikace

Odhlásit uživatele

public Případ užití: Odhlásí uživatele ze systému.

Odstranit uživatele

public Případ užití: Odstraní ze systému zvoleného uživatele. Odstraněný uživatel již nebude moci využívat služeb IS.

Přidat nového uživatele

public Případ užití: Umožní přidat do IS nového uživatele, který bude moci IS využívat. Při vytváření nového uživatele je nutné zadat uživatelské jméno a heslo, kterým se bude do IS přihlašovat. Dále je nutné zadat zda se jedná o vedoucího lékaře, lékaře nebo sestru.

Přihlásit uživatele

public Případ užití: Umožní uživateli, po zadání hesla, se přihlásit do systému v jedné z následujících rolí: Vedoucí lékař, Lékař, Sestra.

Upravit údaje o uživateli

public Případ užití: Umožňuje změnit veškeré údaje evidované u uživatele.

Zadat základní údaje o zdravotnickém zařízení

public Případ užití: Umožňuje zadat všechny údaje týkající se zdravotního zařízení, které IS využívá. Údaje, které je možné nastavit: IČO, DIČ, identifikační číslo zařízení IČZ, název zařízení, adresa zařízení, banka, číslo účtu, odbornost.

Zobrazit seznam uživatelů

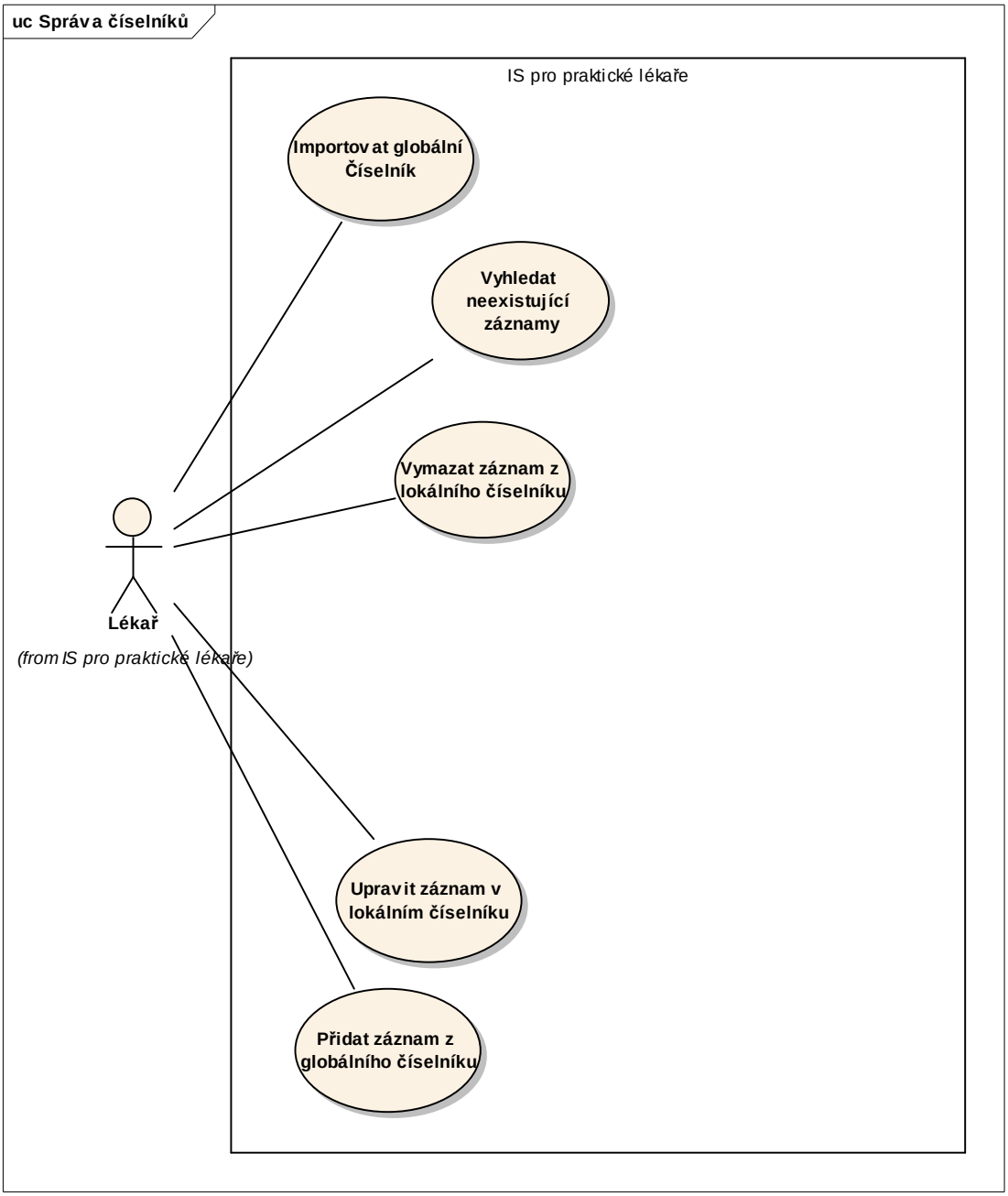
public Případ užití: Zobrazí seznam všech uživatelů, kteří mohou IS využívat.

Modul správa číselníků

Případy užití v tomto balíčku jsou shodné pro všechny číselníky vydávané zdravotní pojišťovnou, a které tento systém využívá. Seznam číselníků, se kterými bude tento IS pracovat:

1. Mezinárodní klasifikace nemocí verze 10 - JDG4.XXX (Diagnózy)
2. Zdravotní výkony - VYKONY.XXX (Výkony)
3. Hromadně vyráběné léčivé přípravky a potraviny pro zvláštní lékařské účely - LEKY.XXX (Léky)
4. Zdravotnické prostředky - PZT.XXX (Zdravotnické prostředky)
5. Smluvní odbornosti pracovišť - ODBORN.XXX (Odbornosti)
6. Kapitační koeficienty
7. Pojišťovny.

Za úplným názvem číselníku je jméno souboru, ve kterém pojišťovna daný číselník distribuuje. V závorce je potom uveden zkrácený název číselníku využívaný v této analýze.



Obrázek 4 : Správa číselníků

Importovat globální Číselník

public Případ užití: Provede načtení položek číselníku ze souboru dodávaného pojišťovnou do databáze.

Associated Files

D:\Dokumenty\Diplomka\VZP\DRCiselniky.pdf (Local File) .
Popis datového rozhraní číselníků vydávaných VZP.

D:\Dokumenty\Diplomka\VZP\DRCiselnikyZmeny.pdf (Local File) .
Popis změn v datovém rozhraní číselníků vydávaných VZP.

D:\Dokumenty\Diplomka\VZP\NoveDRCiselnikuLeky.pdf (Local File) .
Nové datové rozhraní číselníku Léky.

Prohlížet číselník

public Případ užití: Zobrazí záznamy v lokálním číselníku u každého záznamu zobrazí pouze kód a název. Umožňuje výběr způsobu řazení zobrazených záznamů podle kódu nebo názvu. Umožní vybrat jeden ze zobrazených záznamů.

Přidat záznam z globálního číselníku

public Případ užití: Zobrazí seznam záznamů ve vybraném globálním číselníku. Umožňuje přidání vybraných záznamů z globálního číselníku do číselníku lokálního.

Upravit záznam v lokálním číselníku

public Případ užití: Umožňuje upravit záznam v lokálním číselníku.

Vybrat typ číselníku

public Případ užití: Umožní vybrat číselník, s kterým chce uživatel pracovat.

Vyhledat neexistující záznamy

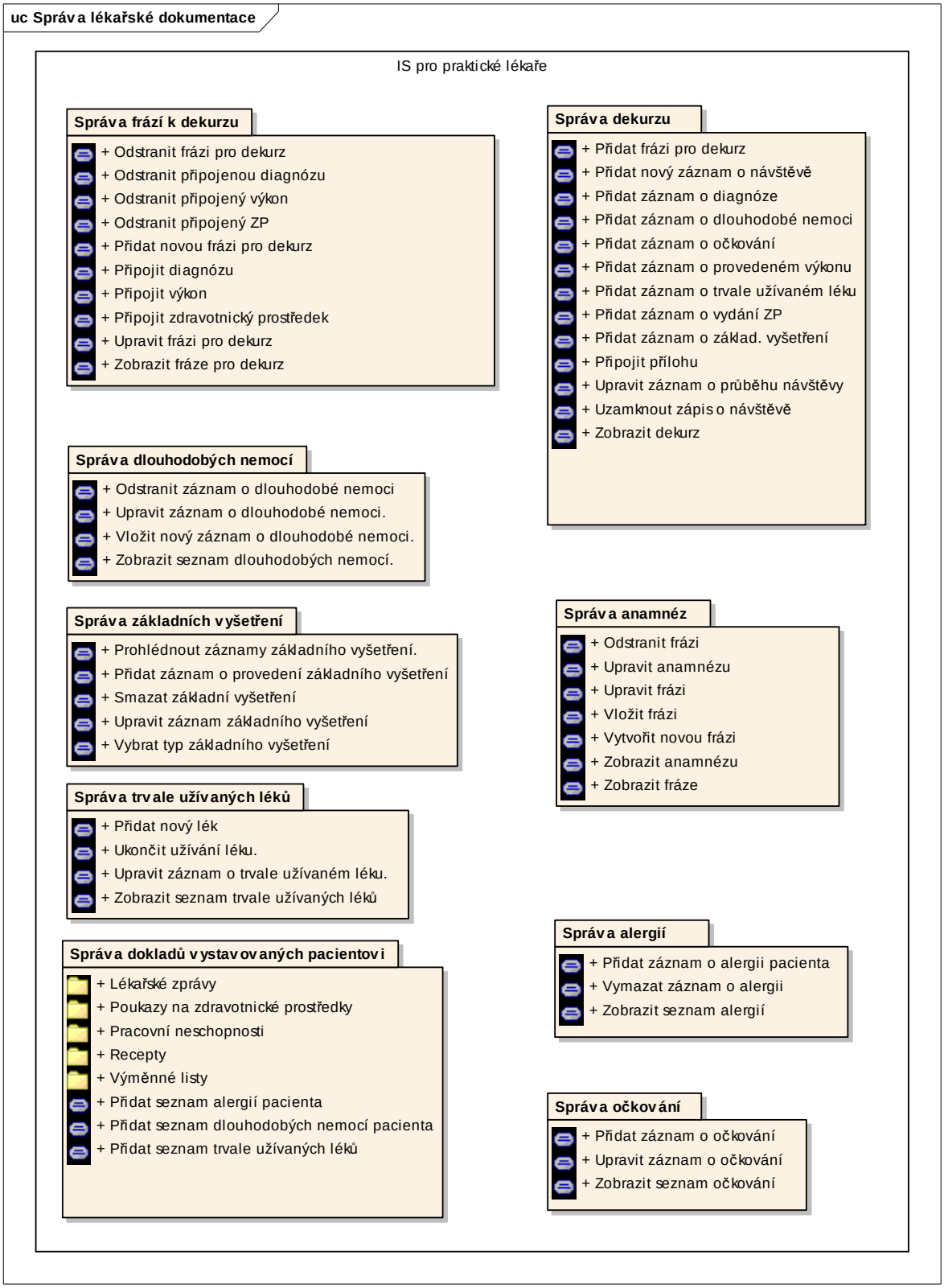
public Případ užití: Zobrazí záznamy z lokálního číselníku, které byly zrušeny v číselníku globálním. Umožňuje tyto záznamy z lokálních číselníků vymazat. Používá se při stažení nových globálních číselníků.

Vymazat záznam z lokálního číselníku

public Případ užití: Odstraní vybraný záznam z lokálního číselníku.

Modul správa lékařské dokumentace

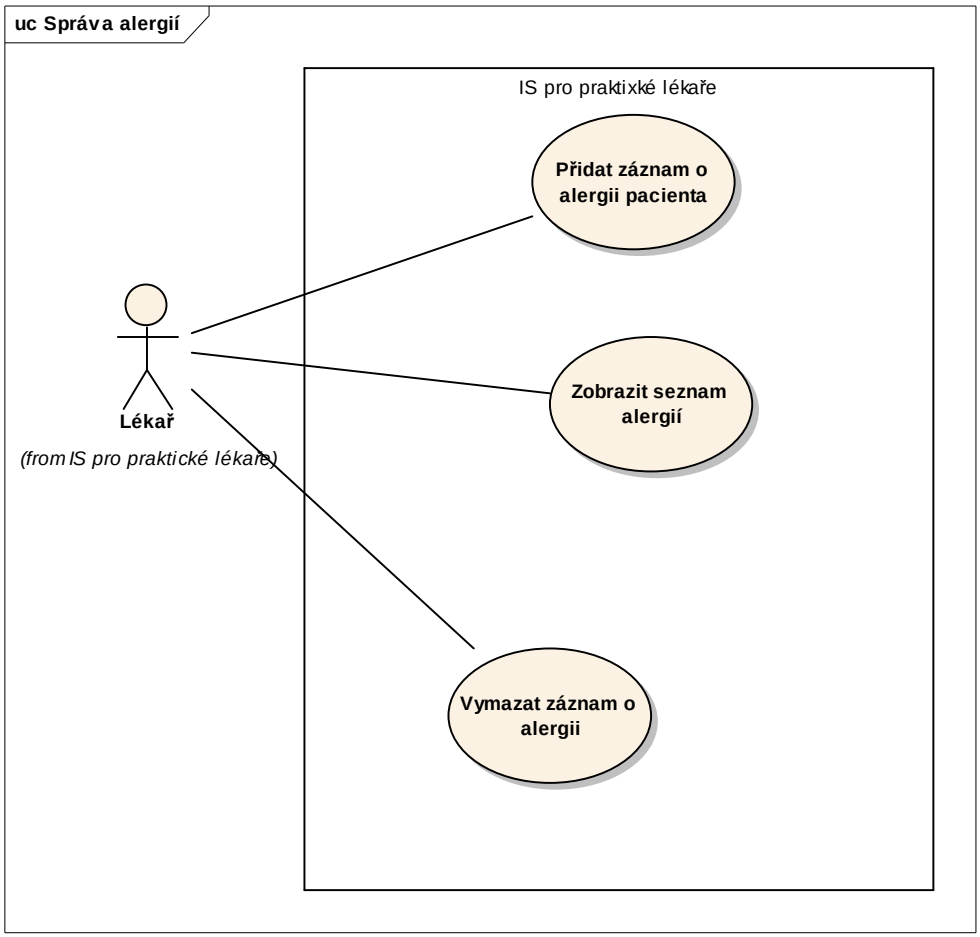
Balíček obsahuje případy užití související s vedením zdravotní dokumentace pacienta.



Obrázek 5 : Správa lékařské dokumentace

Správa alergií

Balíček obsahuje případy užití související s evidencí alergií pacienta.



Obrázek 6 : Správa alergií

Přidat záznam o alergii pacienta

public Případ užití: Přidá nový záznam o alergii pacienta na určitou látku. Záznam o alergii obsahuje název alergie a popis.

Vymazat záznam o alergii

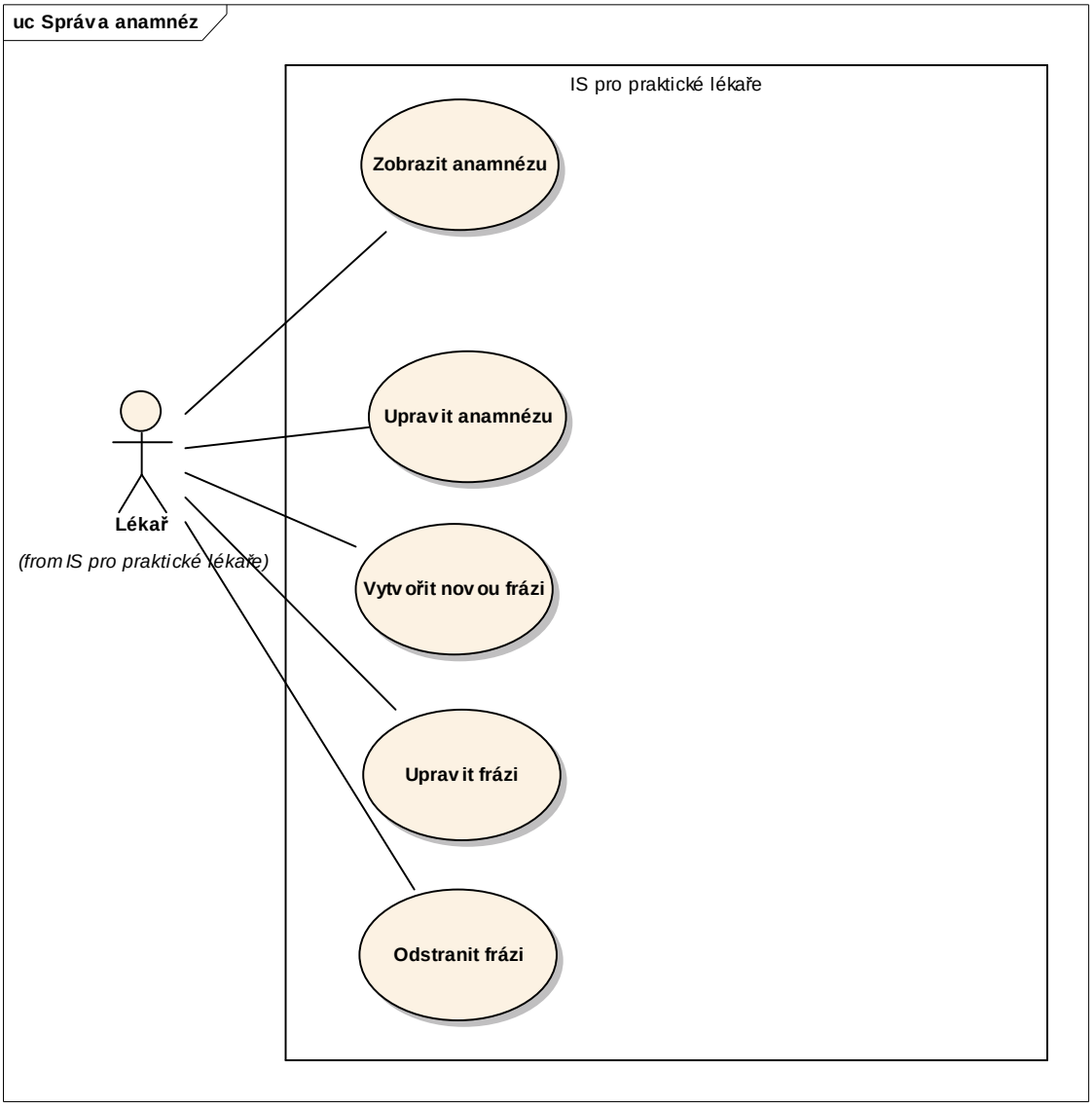
public Případ užití: Odstraní záznam o alergii.

Zobrazit seznam alergií

public Případ užití: Zobrazí seznam všech látek, na které je pacient alergický.

Správa anamnéz

Případy užití v tomto balíčku jsou shodné pro všechny typy evidovaných anamnéz. Evidované anamnézy jsou : Rodinná, Osobní, Pracovní, Sociální, Odborná. Každá anamnéza obsahuje položku umožňující zápis libovolného textu pro doplnění informací o pacientovi.



Obrázek 7 : Správa anamnéz

Odstranit frázi

public Případ užití: Vymaže ze seznamu frází pro anamnézu vybranou frázi.

Upravit anamnézu

public Případ užití: Umožňuje upravovat všechny údaje anamnézy, které byly do systému zadány. Na začátku jsou všechny anamnézy prázdné.

Upravit frázi

public Případ užití: Umožní upravit název nebo text fráze pro anamnézu.

Vložit frázi

public Případ užití: Vloží text fráze do zvolené anamnézy.

Vytvořit novou frázi

public Případ užití: Vloží do seznamu frází pro anamnézu novou frázi obsahující název fráze a vlastní text.

Zobrazit anamnézu

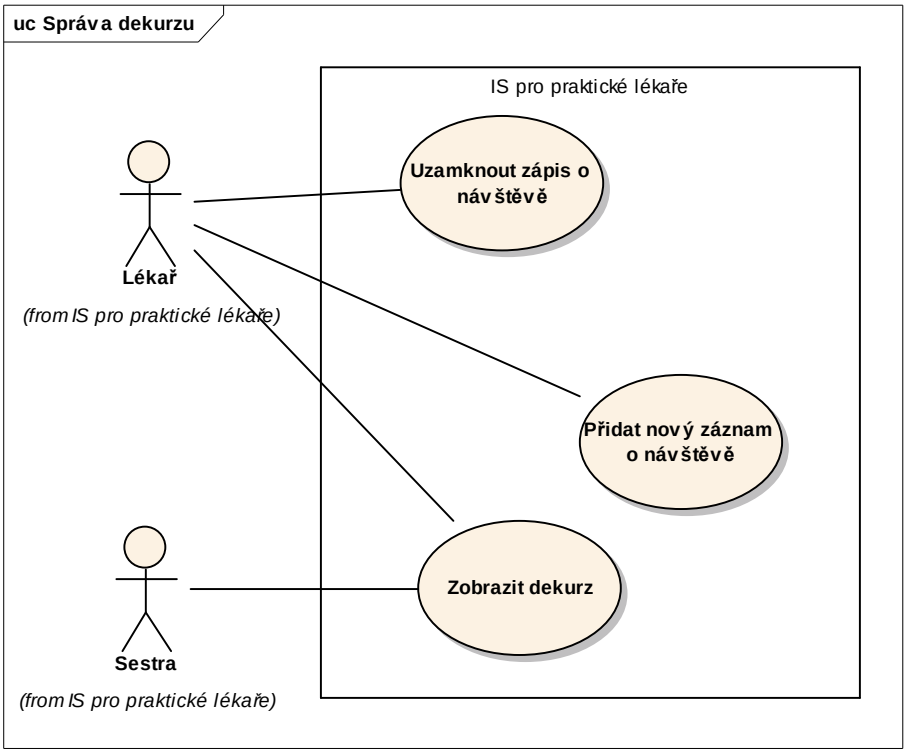
public Případ užití: Zobrazí všechny údaje anamnézy, které byly do systému zadány.

Zobrazit fráze

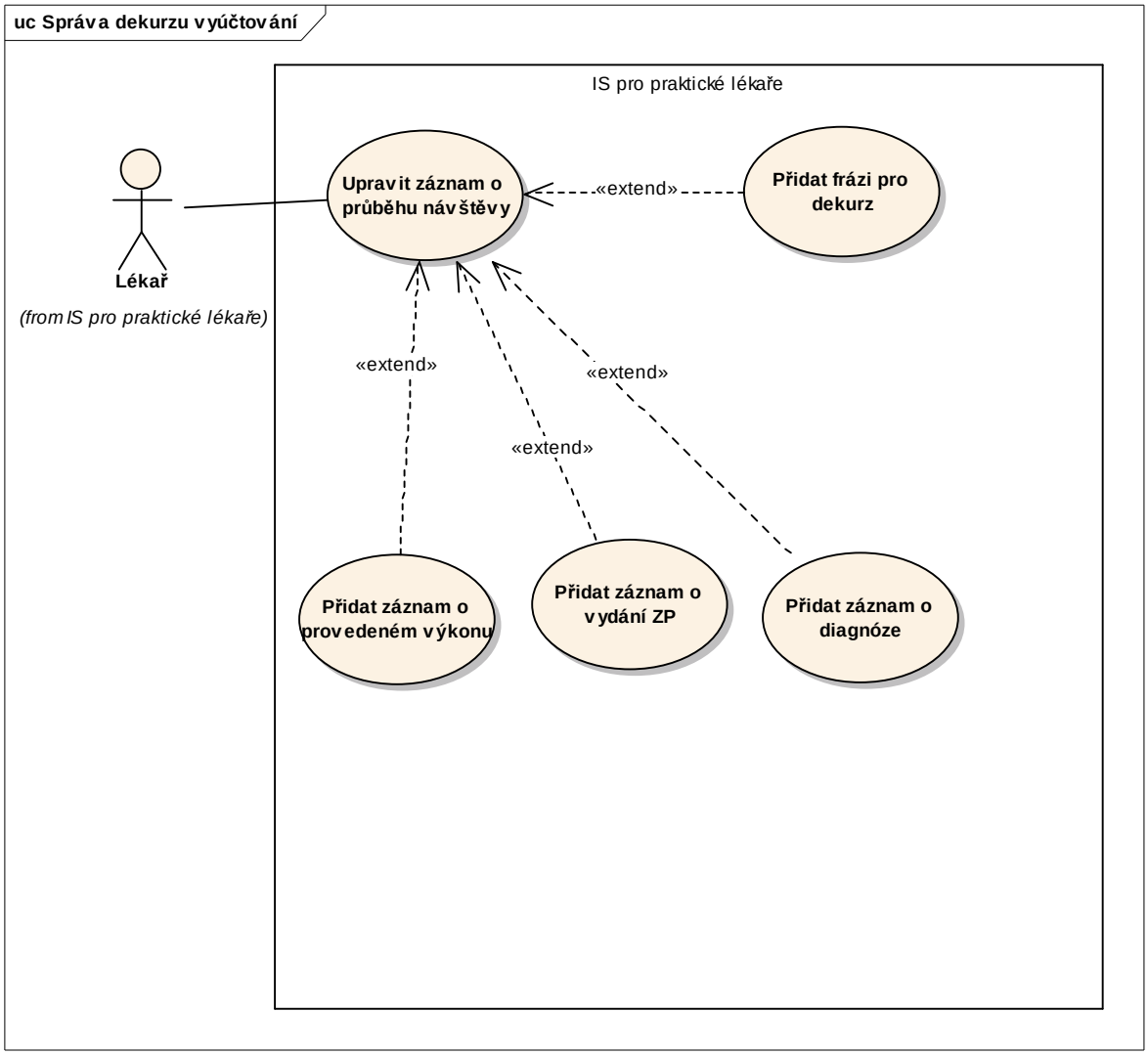
public Případ užití: Zobrazí názvy všech frází pro anamnézu. Umožňuje jednu ze zobrazených frází vybrat.

Správa dekurzu

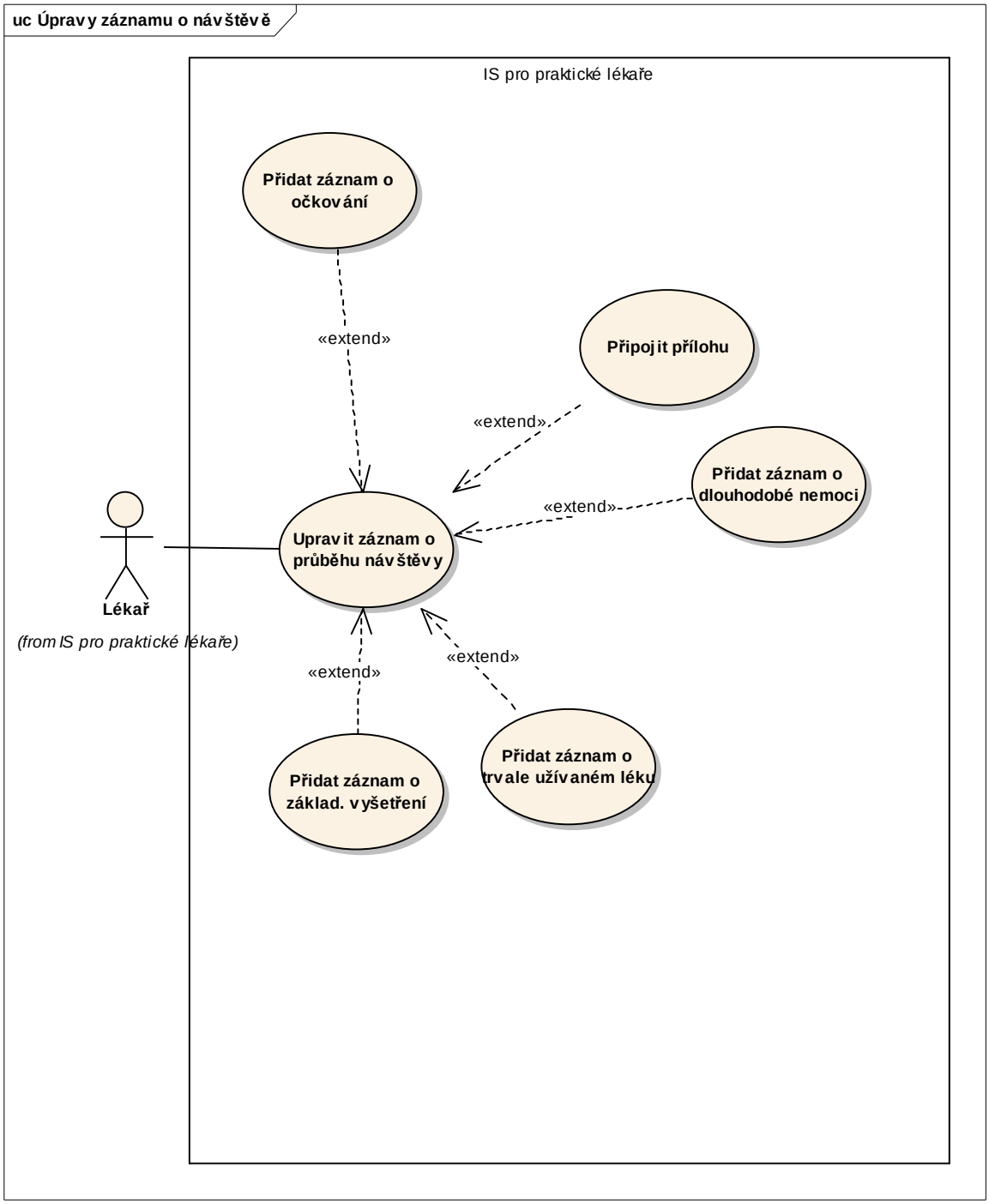
Balíček obsahuje případy užití související se zápisem o průběhu návštěvy pacienta u lékaře.



Obrázek 8 : Správa dekurzu



Obrázek 9 : Správa dekurzu vyúčtování



Obrázek 10 : Úpravy záznamu o návštěvě

Přidat frázi pro dekurz

public Případ užití: Připojí k záznamu o průběhu návštěvy frázi pro dekurz.

Přidat nový záznam o návštěvě

public Případ užití: Přidá nový záznam o návštěvě do dekurzu pacienta. Automaticky k tomuto záznamu přidá aktuální datum a čas.

Přidat záznam o diagnóze

public Případ užití: Přidá záznam o diagnóze do zápisu o průběhu návštěvy.

Přidat záznam o dlouhodobé nemoci

public Případ užití: Připojí k záznamu o průběhu návštěvy záznam o dlouhodobé nemoci, diagnostikované u pacienta, ze seznamu dlouhodobých nemocí.

Přidat záznam o očkování

public Případ užití: Připojí k záznamu o průběhu návštěvy zápis o provedeném očkování ze seznamu očkování.

Přidat záznam o provedeném výkonu

public Případ užití: Přidá záznam o provedení výkonu do zápisu o průběhu návštěvy.

Přidat záznam o trvale užívaném léku

public Případ užití: Připojí k záznamu o průběhu návštěvy záznam o trvale užívaném léku pacientem ze seznamu trvale užívaných léků.

Přidat záznam o vydání ZP

public Případ užití: Přidá záznam o vydání zdravotního prostředku (ZP) do zápisu o průběhu návštěvy.

Přidat záznam o základ. vyšetření

public Případ užití: Připojí k záznamu o průběhu návštěvy zápis o provedeném základním vyšetření.

Připojit přílohu

public Případ užití: Připojí k záznamu o průběhu návštěvy požadovaný soubor jako přílohu. (Obrazová dokumentace, zprávy z vyšetření u specialisty apod.)

Upravit záznam o průběhu návštěvy

public Případ užití: Umožňuje upravovat záznam o průběhu návštěvy pacienta u lékaře. Dále umožňuje automaticky doplnit do záznamu další údaje a připojit různé přílohy.

Uzamknout zápis o návštěvě

public Případ užití: Umožní zápis o návštěvě uzamknout a zabezpečit elektronickým podpisem tak, aby nebylo možné zápis později změnit. Veškeré změny po uzamknutí zápisu musí být uloženy jako opravné zápisy, aby bylo možné kdykoliv zjistit

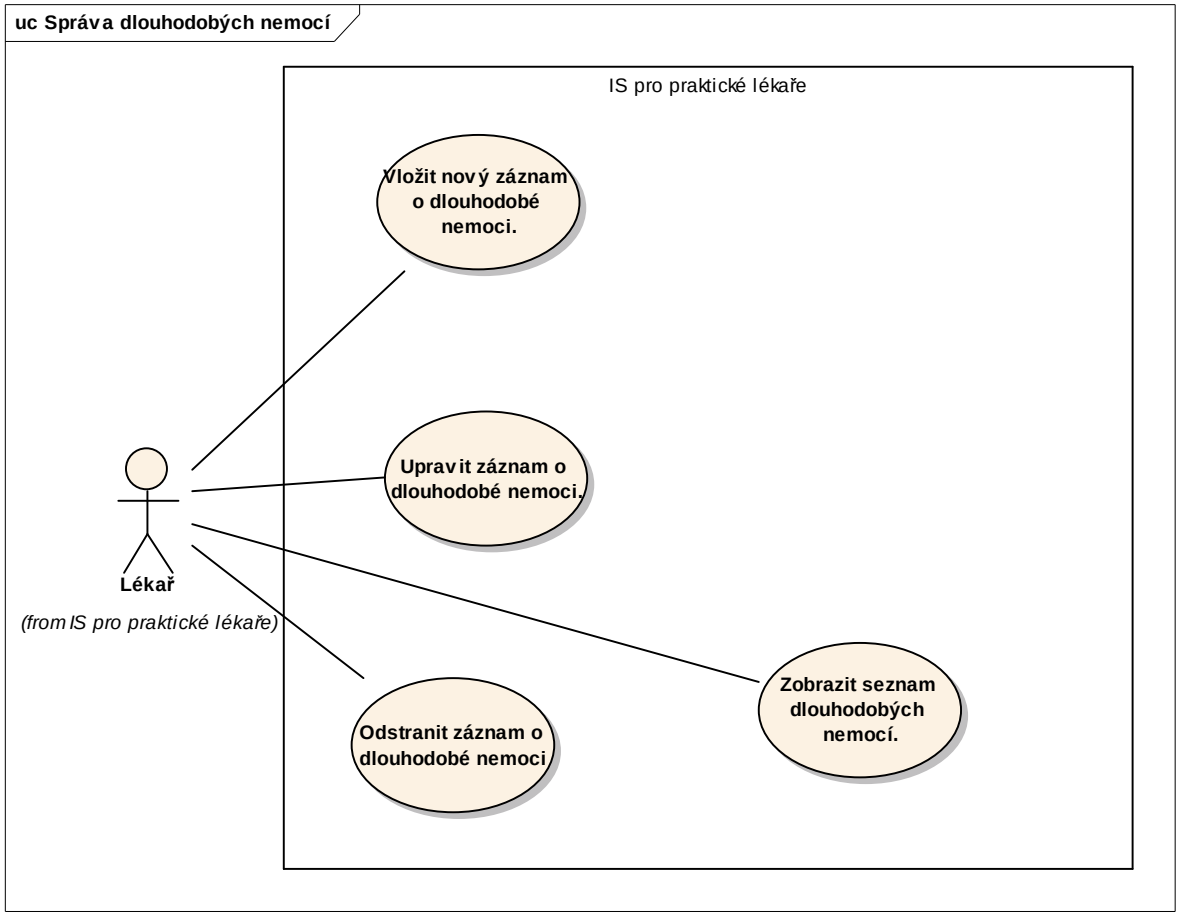
původní zápis o průběhu návštěvy. Tato funkce je vyžadována v případě, že lékař nechce současně vést též papírovou formu dokumentace.

Zobrazit dekurz

public Případ užití: Zobrazí dekurz pacienta. Jednotlivé návštěvy od sebe budou odděleny oddělovačem s uvedeným datumem, kdy se daná návštěva uskutečnila.

Správa dlouhodobých nemocí

Balíček obsahuje případy užití související s evidencí dlouhodobých nemocí u každého pacienta.



Obrázek 11 : Správa dlouhodobých nemocí

Odstranit záznam o dlouhodobé nemoci

public Případ užití: Vymaže záznam o dlouhodobé nemoci u vybraného pacienta.

Upravit záznam o dlouhodobé nemoci.

public Případ užití: Umožňuje opravit údaje uvedené u zvolené nemoci (kód, název, poznámku, datum od, datum do).

Vložit nový záznam o dlouhodobé nemoci.

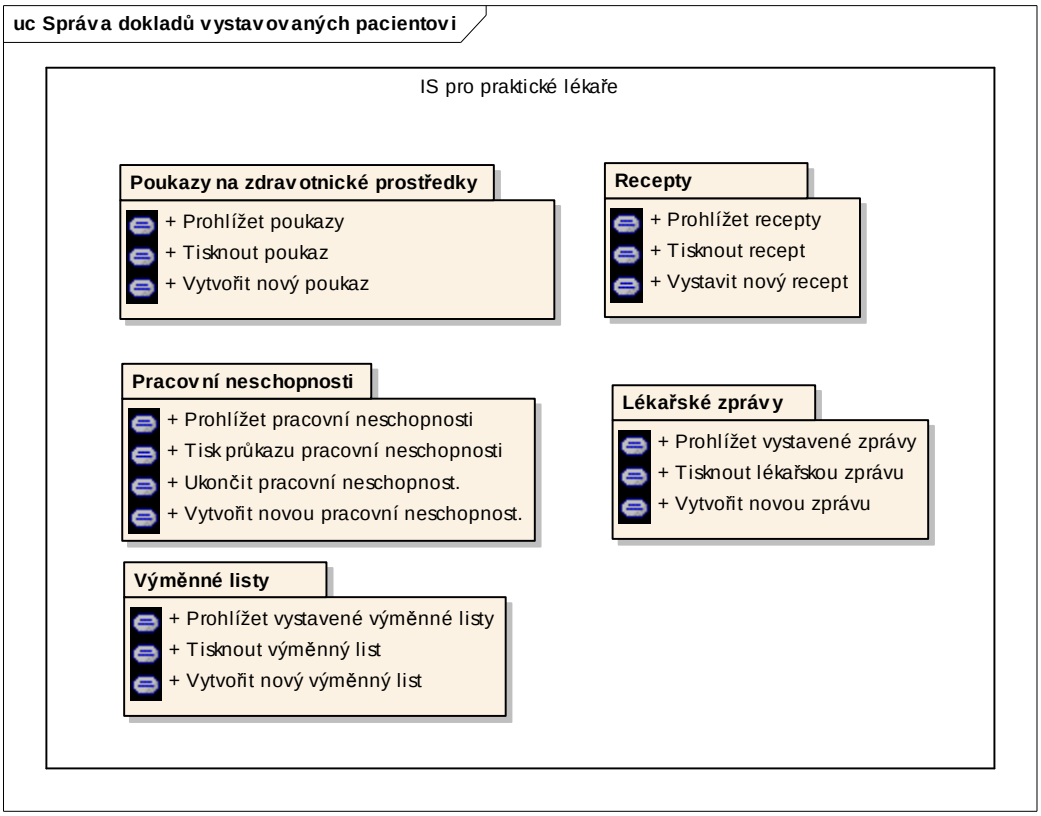
public Případ užití: Vloží do seznamu dlouhodobých nemocí nově diagnostikovanou nemoc a umožňuje k ní připojit vlastní poznámku.

Zobrazit seznam dlouhodobých nemocí.

public Případ užití: Zobrazí seznam všech dlouhodobých nemocí, které byly u daného pacienta diagnostikovány. Umožňuje jeden ze zobrazených záznamů vybrat.

Správa dokladů vystavovaných pacientovi

Balíček obsahuje případy užití související s evidencí dokladů, které byly pacientovi vystaveny.



Obrázek 12 : Správa dokladů vystavovaných pacientovi

Přidat seznam alergií pacienta

public Případ užití: Na zvolený dokument doplní seznam alergií pacienta.

Přidat seznam dlouhodobých nemocí pacienta

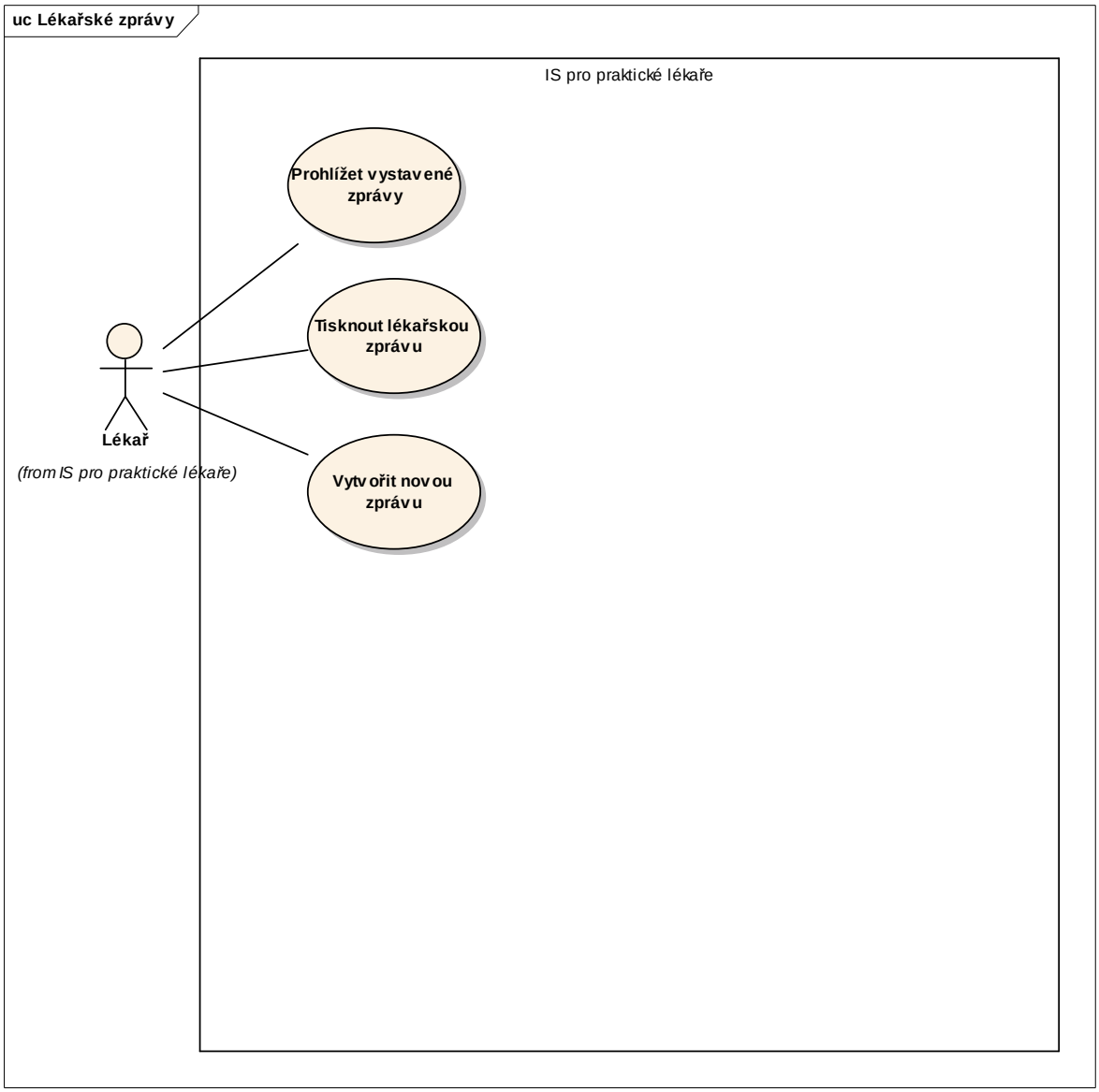
public Případ užití: Na zvolený dokument doplní seznam dlouhodobých nemocí pacienta.

Přidat seznam trvale užívaných léků

public Případ užití: Na zvolený dokument doplní seznam trvale užívaných léků pacienta.

Lékařské zprávy

Balíček obsahuje případy užití související s vystavováním a evidencí již vystavených lékařských zpráv pacientovi.



Obrázek 13 : Lékařské zprávy

Prohlížet vystavené zprávy

public Případ užití: Zobrazí seznam všech doposud vystavených lékařských zpráv, které byly zvolenému pacientovi vystaveny.

Tisknout lékařskou zprávu

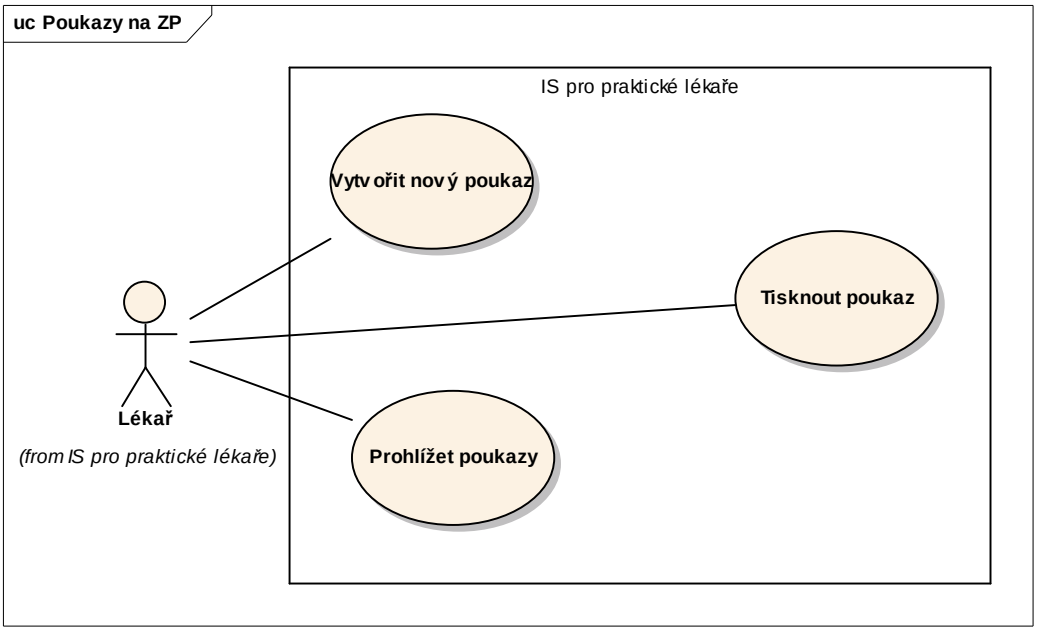
public Případ užití: Vytiskne vybranou lékařskou zprávu.

Vytvořit novou zprávu

public Případ užití: Vytvoří novou lékařskou zprávu. Lékařská zpráva obsahuje: datum, adresu zdravotnického zařízení pro které je zpráva vystavována, jméno a rodné číslo pacienta a vlastní text zprávy.

Poukazy na zdravotnické prostředky

Balíček obsahuje případy užití související s vystavování a evidencí již vystavených poukazů na zdravotnické prostředky.



Obrázek 14 : Poukazy na ZP

Prohlížet poukazy

public Případ užití: Zobrazí seznam všech poukazů, které byly pacientovi vystaveny.

Tisknout poukaz

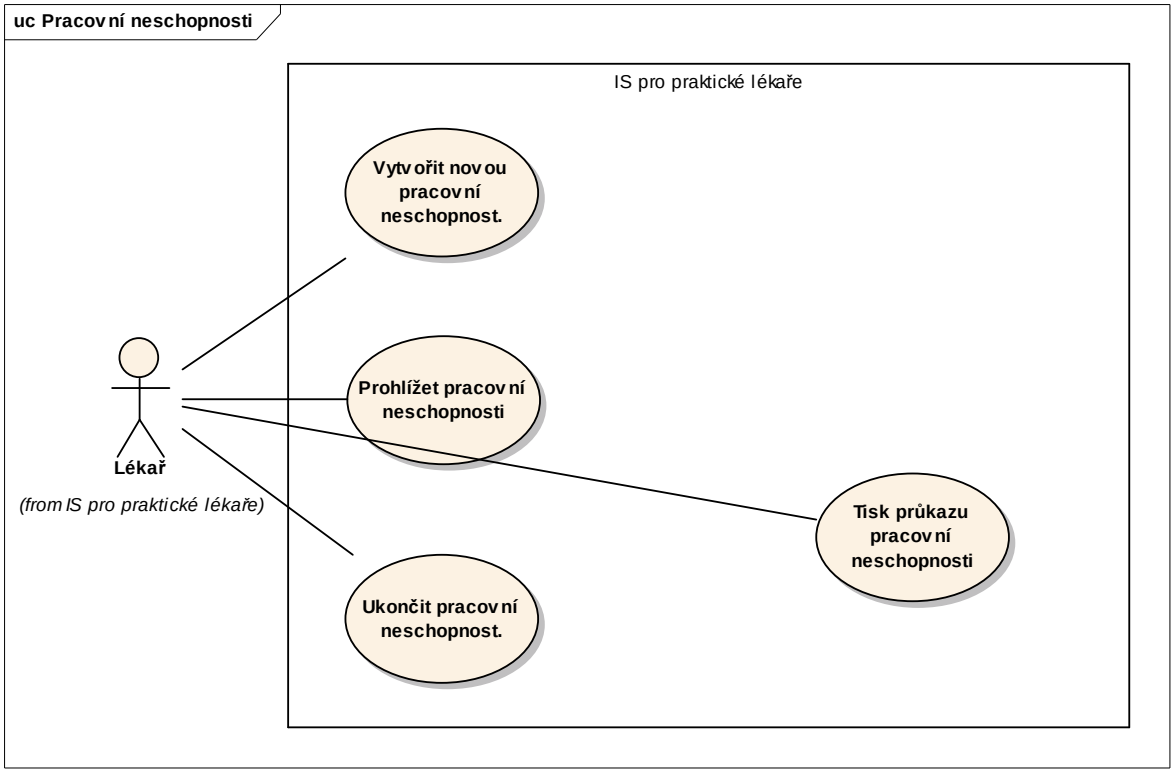
public Případ užití: Vytiskne poukaz na zdravotní pomůcku na předepsaný formulář nebo na čistý papír.

Vytvořit nový poukaz

public Případ užití: Vytvoří nový záznam o vystavení poukazu pacientovi.

Pracovní neschopnosti

Balíček obsahuje případy užití související s vystavováním a evidencí již vystavených průkazů pracovní neschopnosti.



Obrázek 15 : Pracovní neschopnosti

Prohlížet pracovní neschopnosti

public Případ užití: Zobrazí seznam všech pracovních neschopností vybraného pacienta.

Tisk průkazu pracovní neschopnosti

public Případ užití: Vytiskne průkaz o pracovní neschopnosti pacienta.

Ukončit pracovní neschopnost.

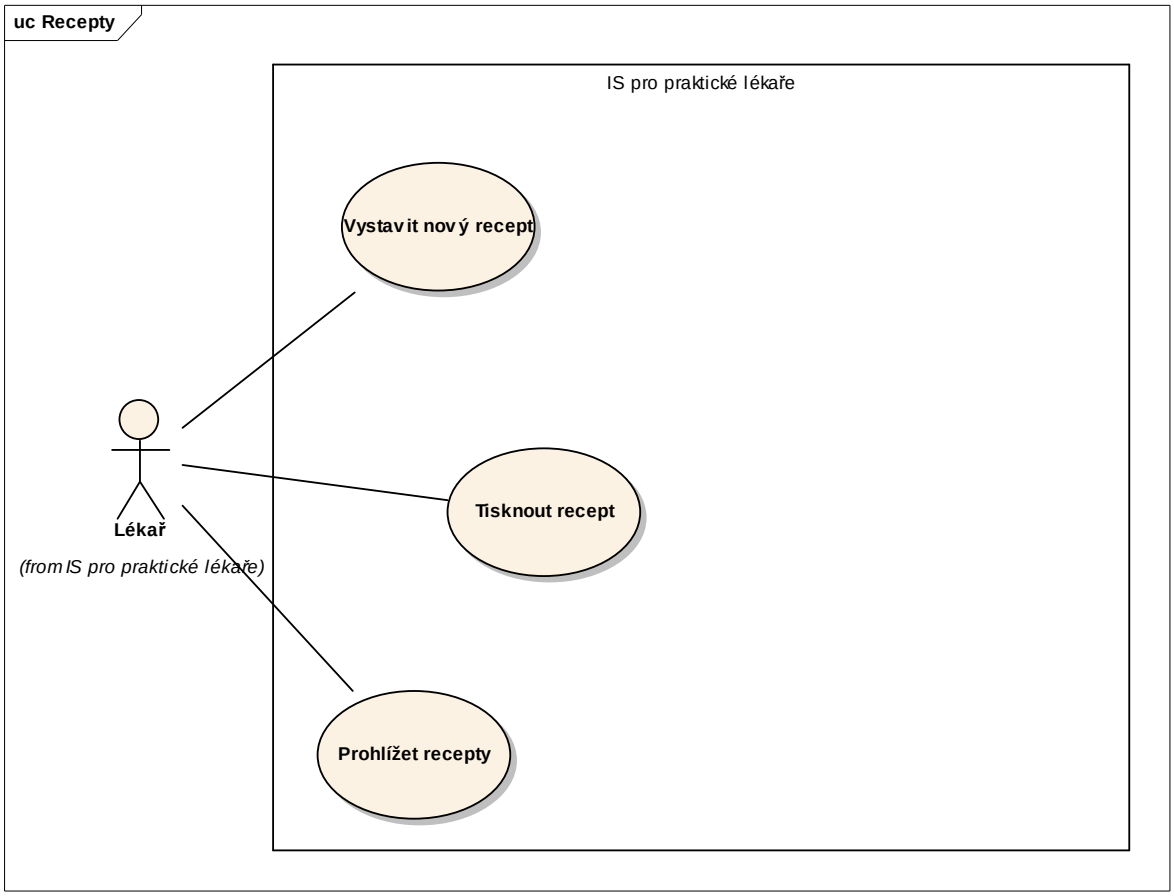
public Případ užití: Umožňuje zadat datum ukončení pracovní neschopnosti.

Vytvořit novou pracovní neschopnost.

public Případ užití: Vytvoří novou pracovní neschopnost. Záznam o pracovní neschopnosti obsahuje údaje: datum začátku neschopnosti, datum ukončení neschopnosti, číslo průkazu a diagnózu.

Recepty

Balíček obsahuje případy užití související s vystavování a evidencí již vystavených receptů.



Obrázek 16 : Recepty

Prohlížet recepty

public Případ užití: Zobrazí seznam všech receptů, které byly pacientovi vystaveny.

Tisknout recept

public Případ užití: Vytiskne zvolený recept na příslušný formulář. Tiskopis receptu vydávaného pojišťovnou je v příloze.

Associated Files

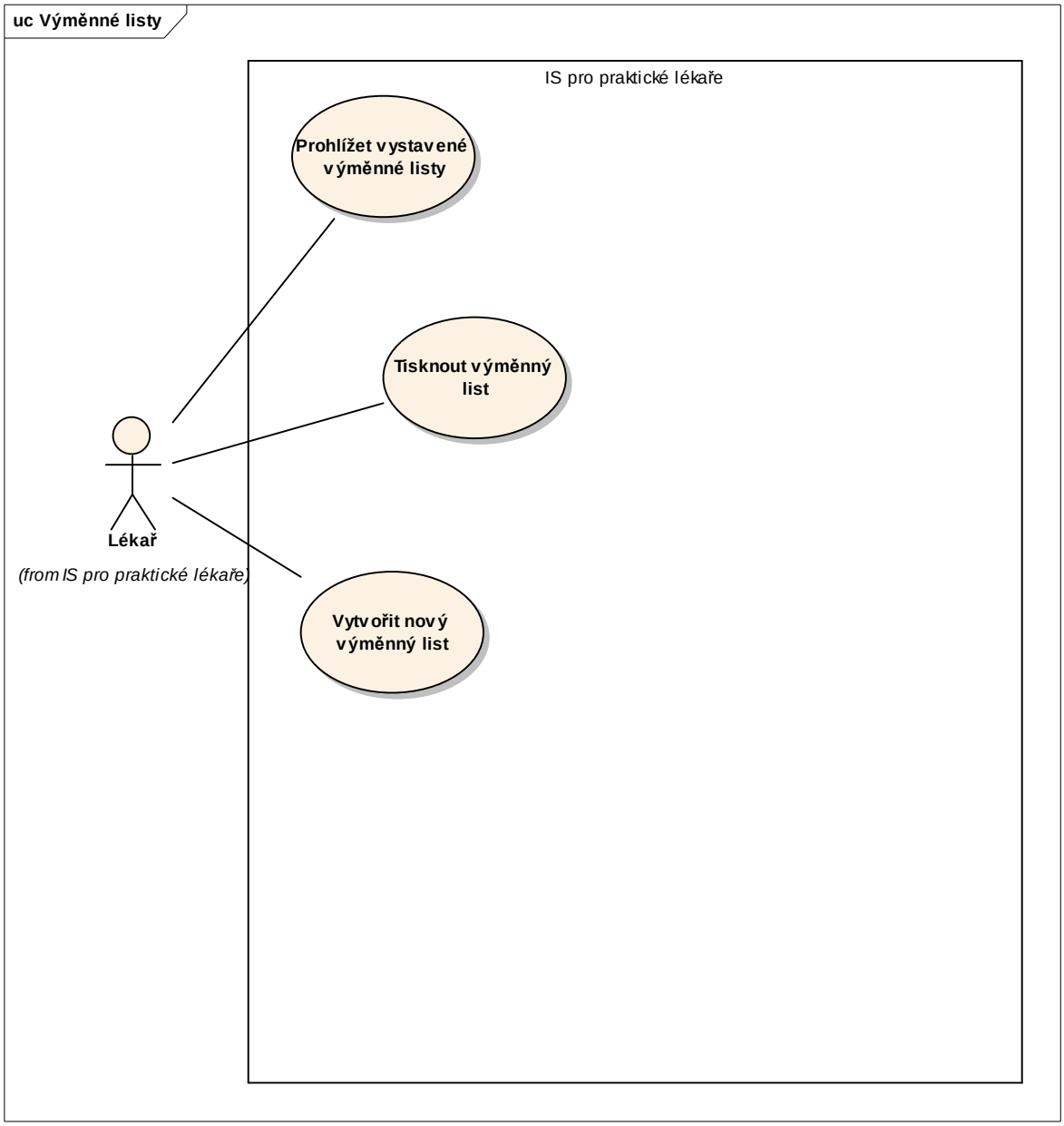
D:\Dokumenty\Diplomka\VZP\Tiskopisy\Recept.pdf (Local File) .
Tiskopis receptu vydávaného VZP.

Vystavit nový recept

public Případ užití: Vytvoří nový záznam o vydání receptu pacientovi. Recept obsahuje kód a název předepsaného léku, dávkování, popřípadě další poznámku.

Výměnné listy

Balíček obsahuje případy užití související s vystavováním a evidencí již vystavených výměnných listů.



Obrázek 17 : Výměnné listy

Prohlížet vystavené výměnné listy

public Případ užití: Zobrazí seznam všech dosud vystavených výměnných listů, které byly pacientovi vystaveny.

Tisknout výměnný list

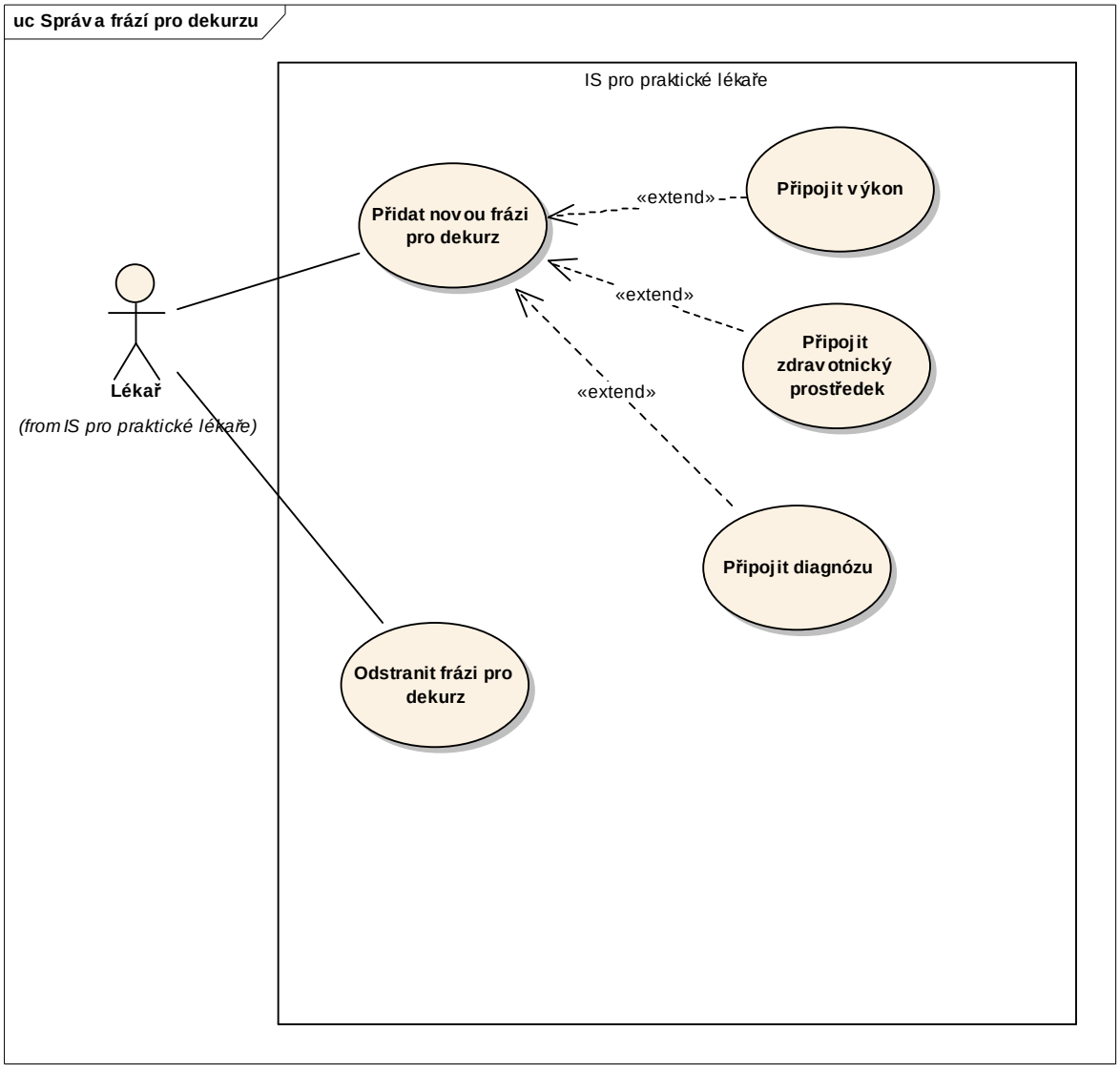
public Případ užití: Vytiskne požadovaný výměnný list.

Vytvořit nový výměnný list

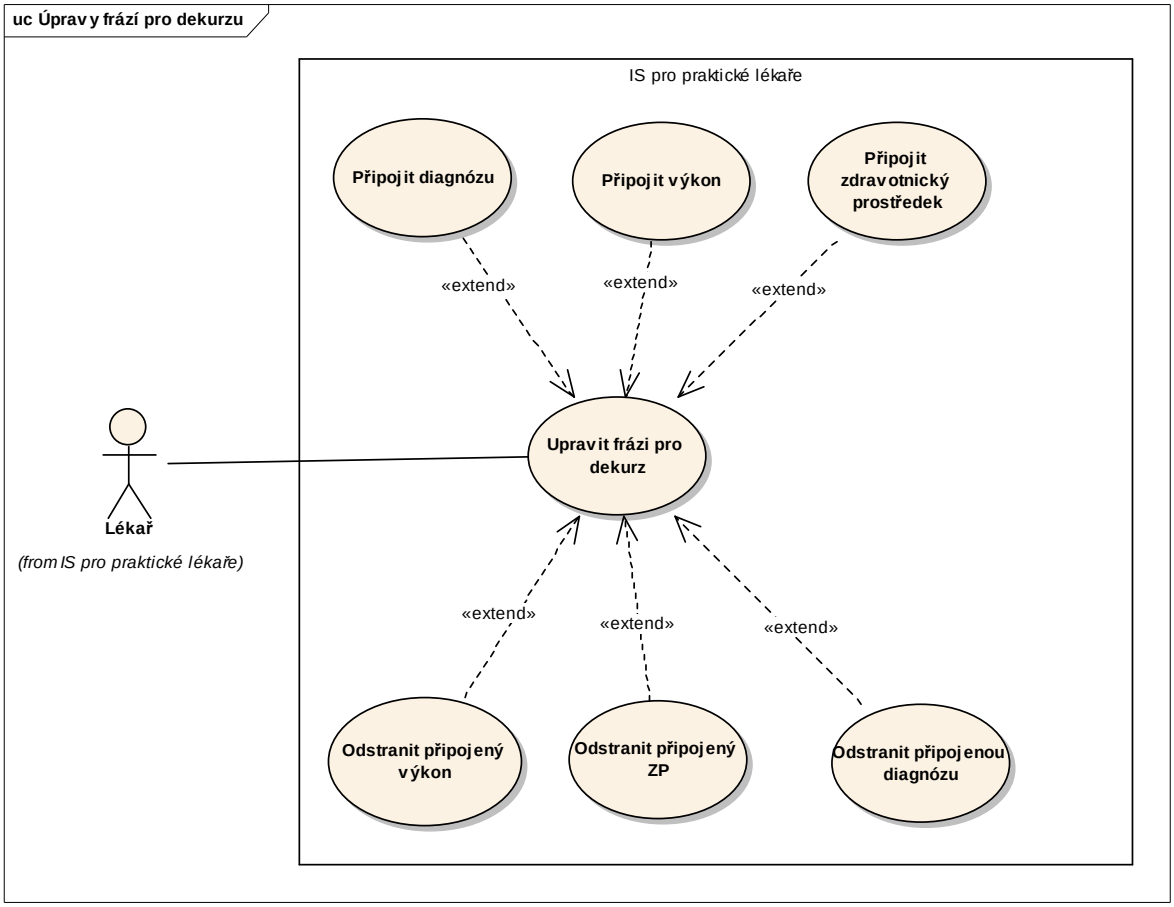
public Případ užití: Vytvoří nový výměnný list. Výměnný list obsahuje: datum, adresu cílového zdravotnického zařízení, které je možno vybrat z katalogu, a dále vlastní text výměnného listu.

Správa frází k dekurzu

Balíček obsahuje případy užití související se správou frází pro dekurz, které může lékař využívat při zápisu průběhu návštěvy pacienta.



Obrázek 18 : Správa frází pro dekurzu



Obrázek 19 : Úpravy frází pro dekurzu

Odstranit frázi pro dekurz

public Případ užití: Odstraní frázi pro dekurz.

Odstranit připojenou diagnózu

public Případ užití: Odstraní připojenou diagnózu.

Odstranit připojený výkon

public Případ užití: Odstraní výkon připojený k vybrané frázi pro dekurz.

Odstranit připojený ZP

public Případ užití: Odstraní připojený zdravotnický prostředek.

Přidat novou frázi pro dekurz

public Případ užití: Přidá novou frázi pro dekurz.

Připojit diagnózu

public Případ užití: Připojí k frázi pro dekurz diagnózu.

Připojit výkon

public Případ užití: Připojí k frázi pro dekurz výkon.

Připojit zdravotnický prostředek

public Případ užití: Připojí k frázi pro dekurz zdravotnický prostředek.

Upravit frázi pro dekurz

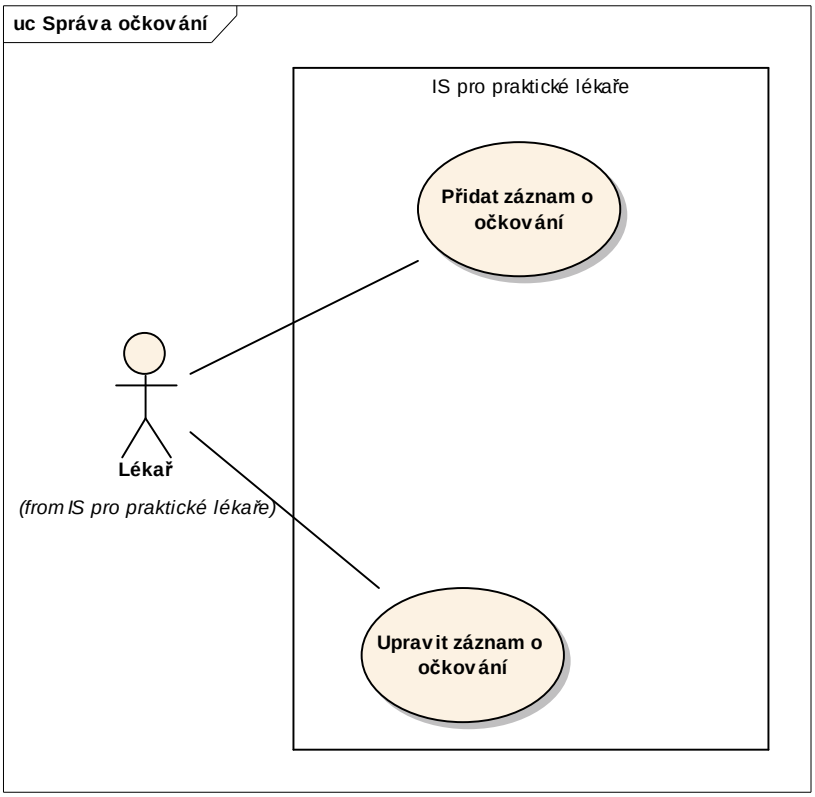
public Případ užití: Upraví již existující frázi pro dekurz.

Zobrazit fráze pro dekurz

public Případ užití: Zobrazí všechny fráze pro dekurz.

Správa očkování

Balíček obsahuje případy užití související s evidencí provedených očkování.



Obrázek 20 : Správa očkování

Přidat záznam o očkování

public Případ užití: Přidá do seznamu očkování nový záznam.

Upravit záznam o očkování

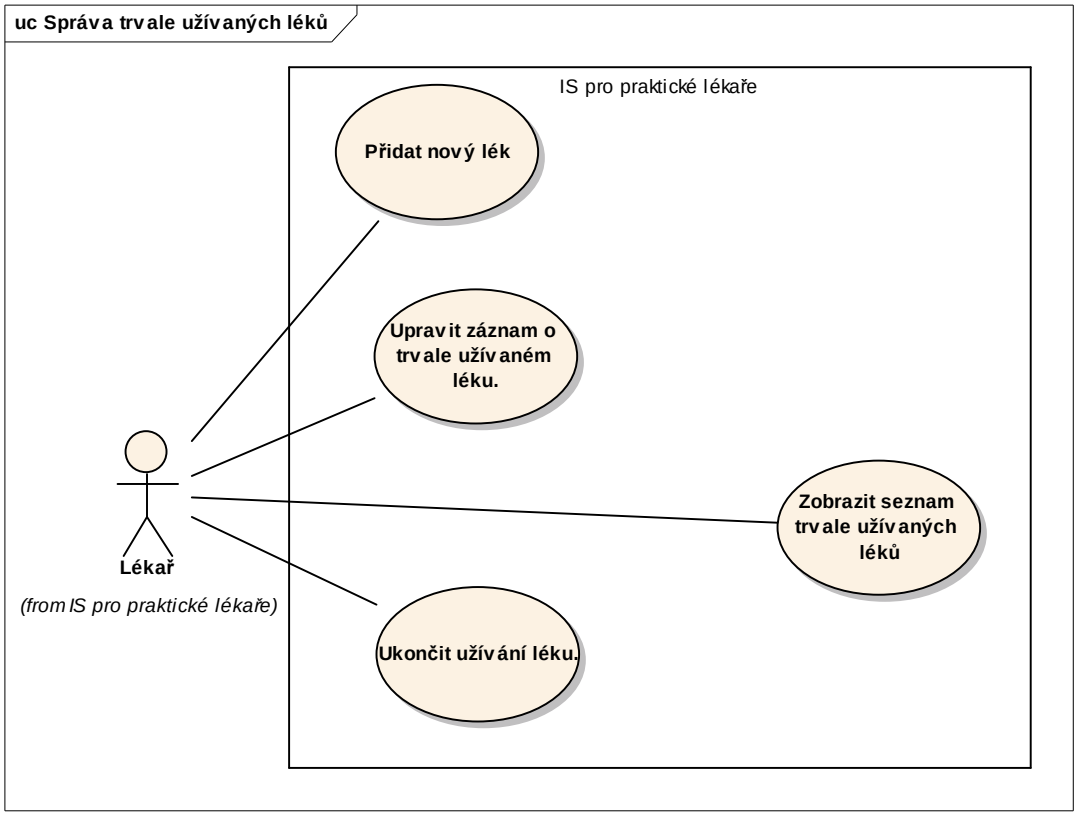
public Případ užití: Umožňuje změnit údaje evidované v záznamu o očkování.

Zobrazit seznam očkování

public Případ užití: Zobrazí seznam všech očkování provedených u pacienta.

Správa trvale užívaných léků

Balíček obsahuje případy užití související s evidencí trvale užívaných léků pacientem.



Obrázek 21 : Správa trvale užívaných léků

Přidat nový lék

public Případ užití: Přidá na seznam trvale užívaných léků nový lék. U každého trvale užívaného léku je evidováno: kód léku, název léku, doplněk názvu, počet balení, dávkování, datum od, datum do a vlastní poznámku k danému léku.

Ukončit užívání léku.

public Případ užití: Pro vybraný lék ukončí používání zaznamenáním aktuálního datumu do položky datum do. Lék zůstane v seznamu trvale užívaných léků, pouze je označen jako již neužívaný.

Upravit záznam o trvale užívaném léku.

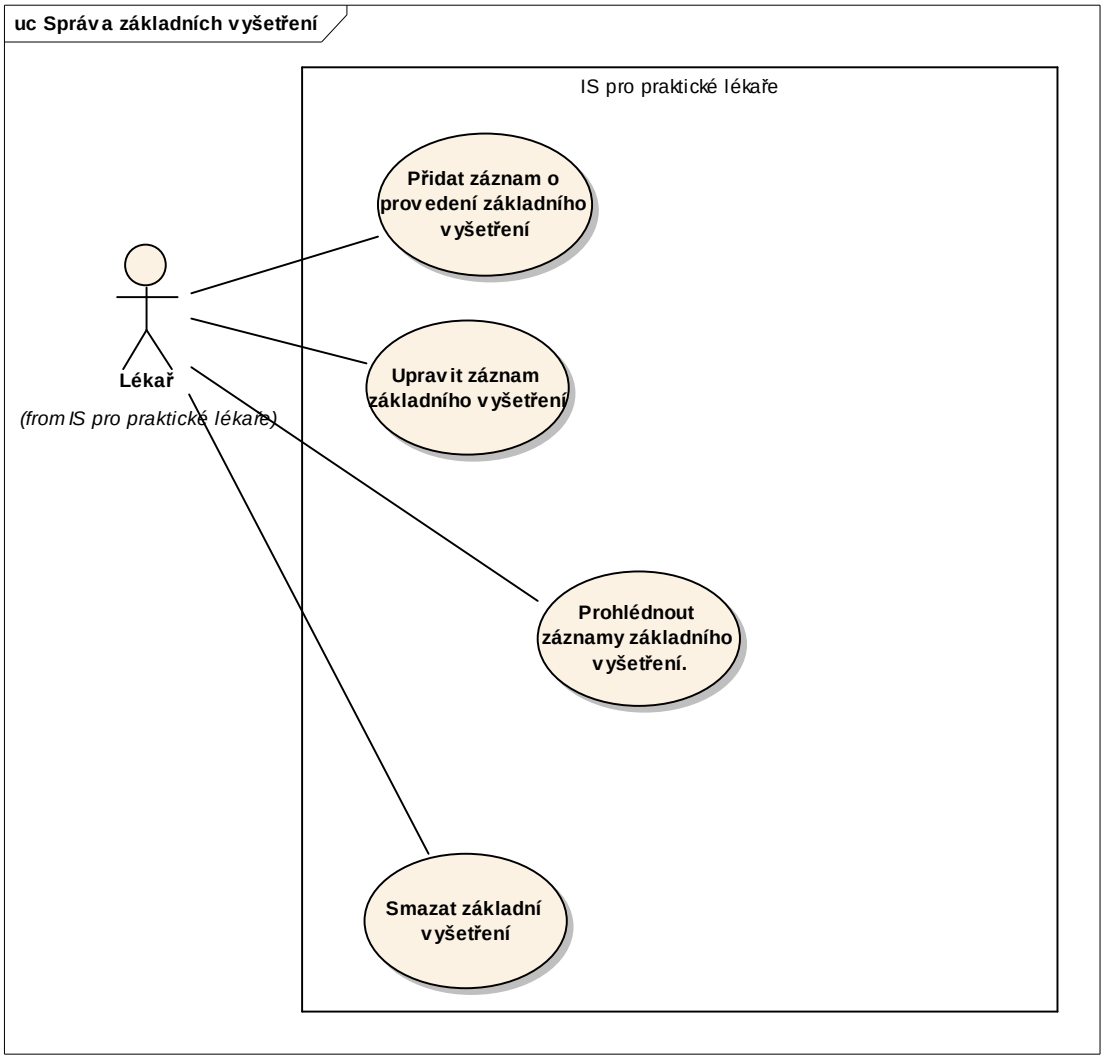
public Případ užití: Umožňuje měnit veškeré údaje záznamu trvale užívaného léku.

Zobrazit seznam trvale užívaných léků

public Případ užití: Zobrazí seznam všech trvale užívaných léků u vybraného pacienta. Léky, u kterých byla medikace ukončena zobrazí odlišným formátem popř. umožní skrytí takovýchto léků.

Správa základních vyšetření

Balíček obsahuje případy užití související s evidencí výsledků základních vyšetření. Jednotlivé typy základních vyšetření jsou: tlak a puls, hmotnost a výška, sedimentace.
Případy užití v tomto balíčku jsou shodné pro všechny typy základních vyšetření. Pouze se liší různými atributy, které jsou u jednotlivých základních vyšetření evidovány.



Obrázek 22 : Správa základních vyšetření

Prohlédnout záznamy základního vyšetření.

public Případ užití: Zobrazí seznam záznamů o provedení základního vyšetření. Umožňuje jeden ze zobrazených záznamů vybrat.

Přidat záznam o provedení základního vyšetření

public Případ užití: Přidá záznam o provedení základního vyšetření pacienta. Zobrazí formulář umožňující zadat údaje evidované u zvoleného vyšetření.

Smazat základní vyšetření

public Případ užití: Po potvrzení smaže záznam o provedení vybraného základního vyšetření.

Upravit záznam základního vyšetření

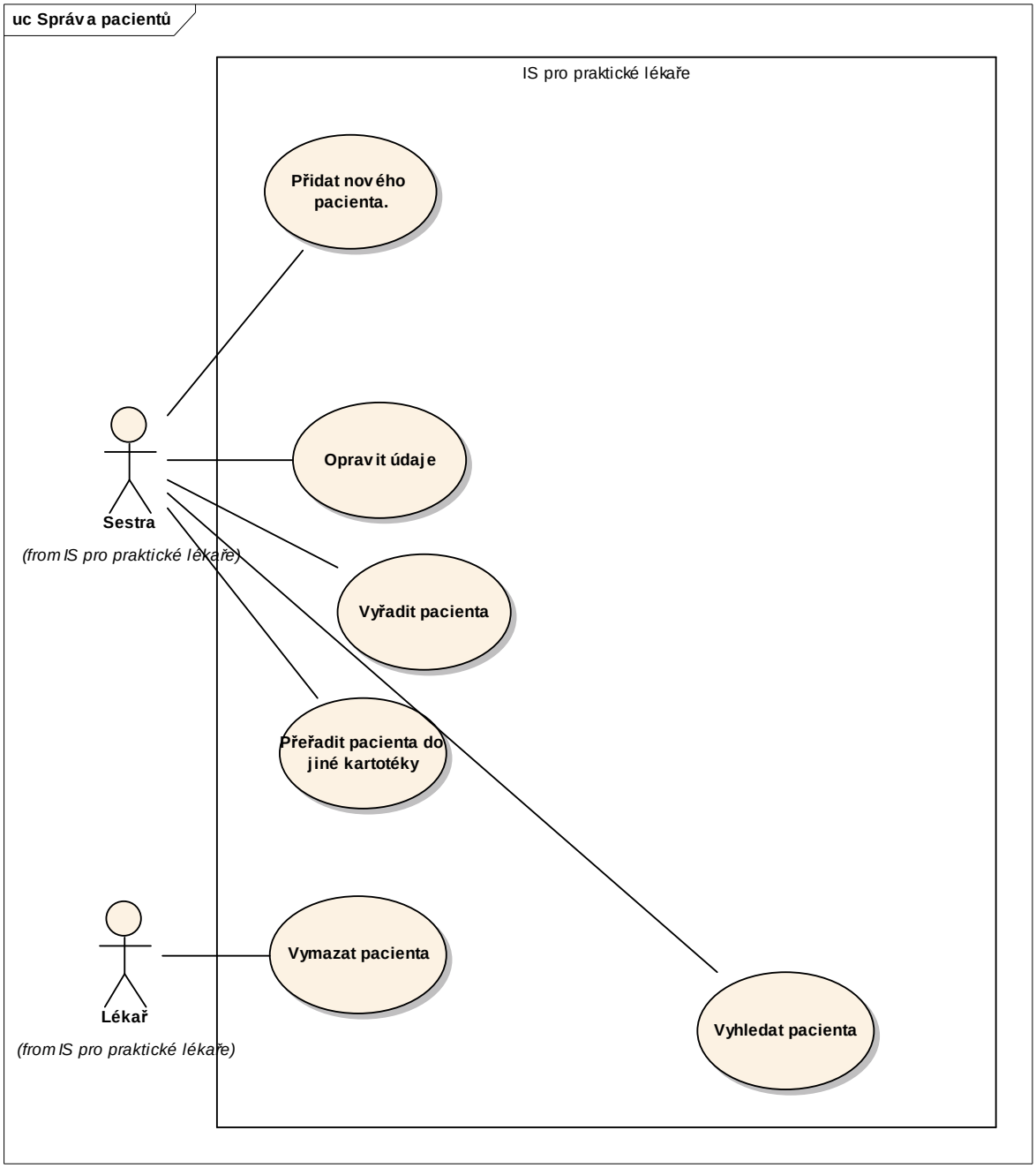
public Případ užití: Umožní upravit veškeré údaje evidovaného u zvoleného základního vyšetření.

Vybrat typ základního vyšetření

public Případ užití: Umožňuje vybrat typ základního vyšetření (Tlak a puls, Hmotnost a výška, Sedimentace).

Modul správa pacientů

Balíček obsahuje případy užití související s evidencí pacientů. Každý evidovaný pacient je zařazen právě v jedné kartotéce. Každý pacient může být členem několika skupin vytvářených lékařem.
Pro každou ordinaci evidovanou v rámci systému je vytvářena samostatná kartotéka pacientů. Dále existuje další samostatná kartotéka pacientů nepravidelné péče společná pro všechny ordinace v rámci systému.



Obrázek 23 : Správa pacientů

Opravit údaje

public Případ užití: Umožňuje měnit veškeré údaje, které jsou u pacienta evidovány.

Přeřadit pacienta do jiné kartotéky

public Případ užití: Umožní přesunout pacienta do jiné kartotéky evidované v rámci systému.

Přidat nového pacienta.

public Případ užití: Přidá nového pacienta do kartotéky vybrané ordinace nebo do kartotéky nepravidelné péče.Kartotéka ordinace, do které má být pacient zařazen je dána přihlášením lékaře.

Vyhledat pacienta

public Případ užití: Umožňuje vyhledat pacienta podle rodného čísla nebo jména.

Vymazat pacienta

public Případ užití: Vyprší-li zákonem daná lhůta pro evidenci lékařské dokumentace vyřazeného pacienta, lze ho odstranit trvale i z kartotéky vyřazených pacientů.

Vyřadit pacienta

public Případ užití: Přesune pacienta do kartotéky vyřazených pacientů. Pacient zůstává dále evidován z důvodu uchování jeho zdravotní dokumentace.

Zkontrolovat zadané údaje

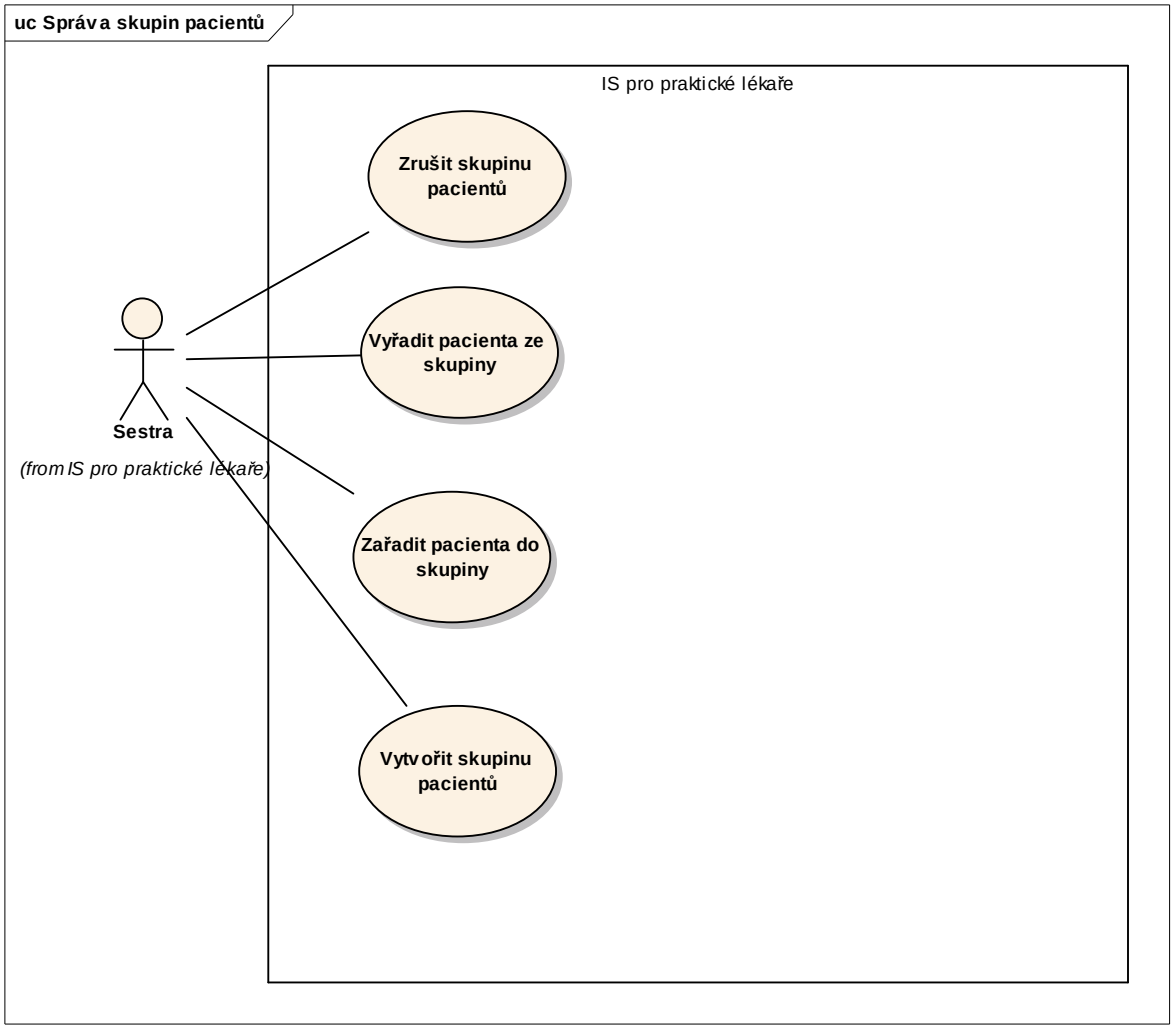
public Případ užití: Provede kontrolu, zda jsou zadány všechny povinné údaje.

Zobrazit kartotéku pacientů

public Případ užití: Zobrazí kartotéku pacientů. Umožňuje výběr způsobu řazení pacientů (vzestupně, sestupně, podle rodného čísla, podle jména). Umožňuje určit skupiny pacientů, které mají být zobrazeny (pouze z této ordinace, nepravidelná péče, vyřazení pacienti, všichni pacienti, či jinou zvolenou skupinu, kterou si lékař sám vytvořil).

Správa skupin pacientů

Balíček obsahuje případy užití související s vytvářením skupin pacientů tak, aby bylo pro lékaře snazší vyhledávat pacienty, kteří mají nějaké vlastnosti společné (patří do stejné skupiny).



Obrázek 24 : Správa skupin pacientů

Výběr množiny zobrazovaných pacientů

public Případ užití: Umožní vybrat množinu pacientů, která má být v kartotéce zobrazena. Zobrazit lze: všechny pacienty, pouze pacienty z kartotéky aktivní ordinace, pacienty z kartotéky nepravidelná péče, vyřazené pacienty. Dále je možné zobrazovat pouze pacienty, kteří jsou členy vybrané skupiny nadefinované lékařem.

Vytvořit skupinu pacientů

public Případ užití: Vytvoří novou skupinu pacientů, kterou lékař může využít pro rozčlenění pacientů.

Vyřadit pacienta ze skupiny

public Případ užití: Vyřadí vybrané pacienty ze skupiny.

Zařadit pacienta do skupiny

public Případ užití: Zařadí vybrané pacienty do zvolené skupiny.

Zobrazit seznam skupin pacientů

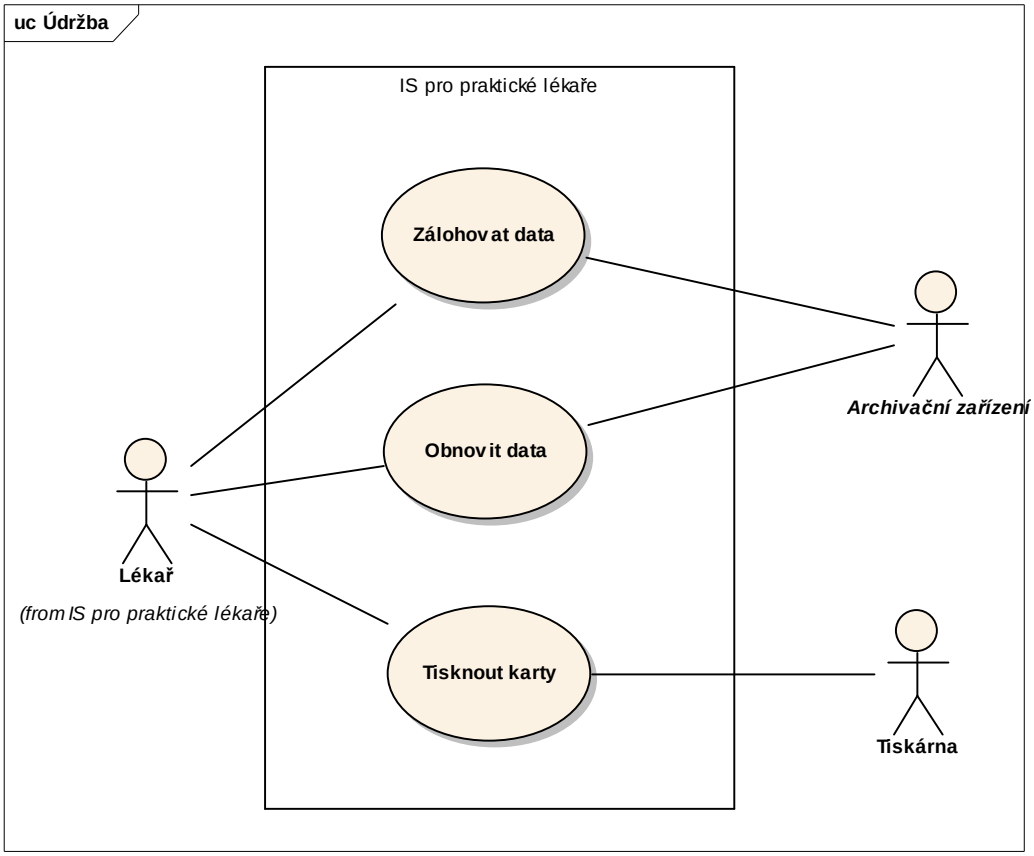
public Případ užití: Zobrazí všechny skupiny vytvořené uživatelem. Umožní jednu z nich vybrat.

Zrušit skupinu pacientů

public Případ užití: Zruší skupinu pacientů - pacienti zůstávají registrováni pouze se ruší jejich členství v této skupině.

Modul údržba dat

Balíček obsahuje případy užití související s archivací pořízených dat v elektronické a papírové podobě.



Obrázek 25 : Údržba

Archivační zařízení

public abstract Actor: Zařízení umožňující archivovat pořízené údaje. Může se jednat o síťový disk, ZIP, Flash paměť, CD-RW nebo jiné zálohovací médium.

Tiskárna

public Actor: Zařízení umožňující tisk na volné papíry tak na vložené speciální formuláře dodávané pojišťovnou.

Obnovit data

public Případ užití: Umožní obnovit pořízená data ze zálohy.

Tisknout karty

public Případ užití: Umožňuje převést elektronicky pořízená data do tištěné podoby. Systém si musí uchovávat informaci o poloze naposledy vytištěného záznamu v kartě každého pacienta, aby bylo možné pokračovat s tiskem na stejném listu.

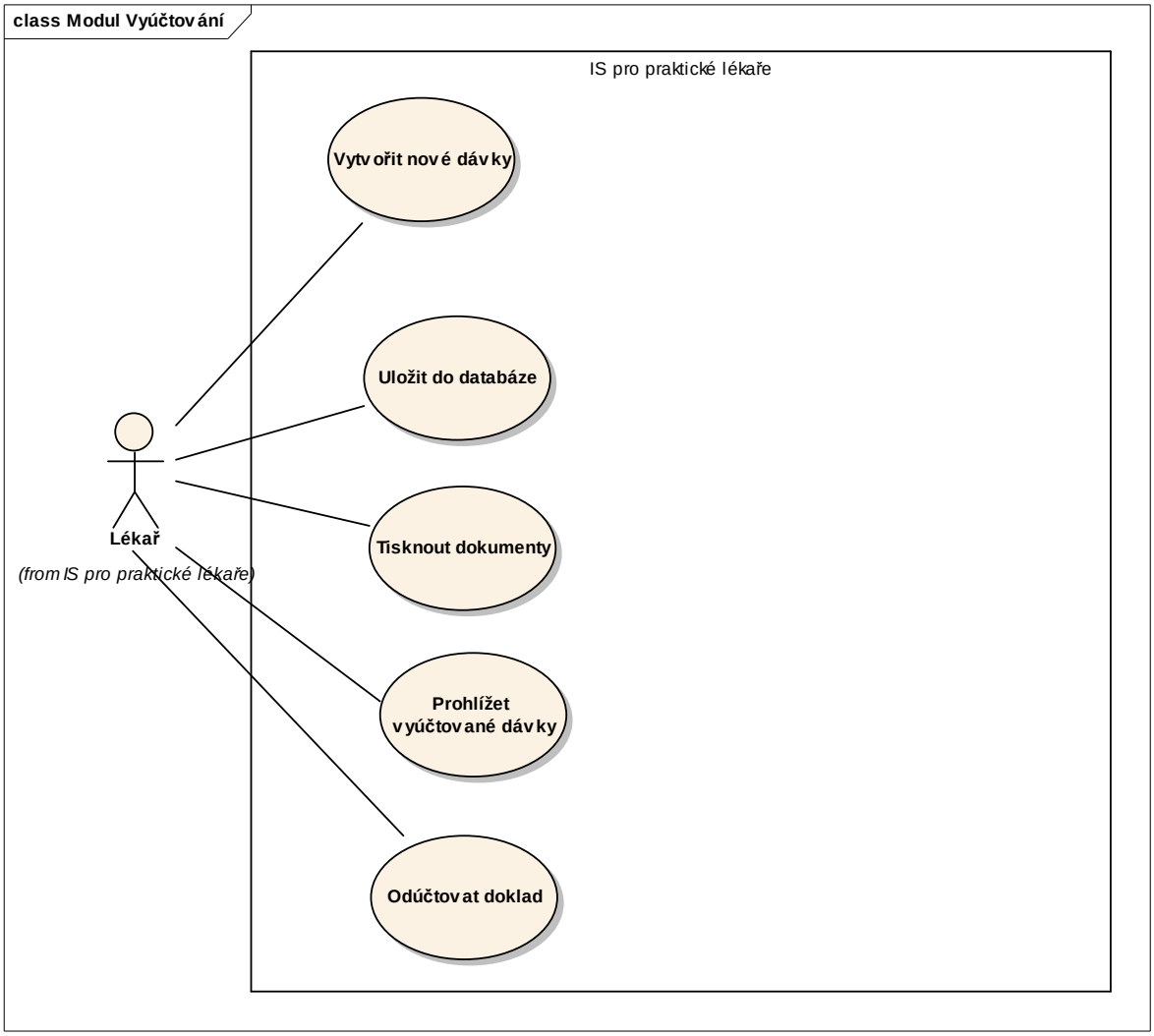
Zálohovat data

public Případ užití: Umožní zálohovat veškerá pořízená data.

Modul vyúčtování

V tomto balíčku jsou zahrnuty případy užití související s vykazováním provedené péče pojišťovnám a vystavováním potřebných dokladů. Dávky dokladů, které bude systém generovat:

05 - Dávka vyúčtování výkonů nepravidelné péče a LSPP
98 - Dávka ambulantní smíšená
80 - Dávka pro registraci pojištěnců



Obrázek 26 : Modul Vyúčtování

Odúčtovat doklad

public Případ užití: Umožní odúčtovat záznamy, na které byl doklad vystaven tak, aby byly příště opět zahrnuty do vyúčtování.

Prohlížet vyúčtované dávky

public Případ užití: Zobrazí seznam všech vyúčtovaných dávek pro vybranou pojišťovnu.

Tisknout dokumenty

public Případ užití: Umožňuje vytisknout na tiskárně potřebné dokumenty. Faktura za dávky se nejčastěji používá při opravném účtování dokladů, které byly při předchozím zpracování pojišťovnou odmítnuty.

Associated Files

D:\Dokumenty\Diplomka\VZP\Tiskopisy\FakturaZaDavky.pdf (Local File) .
Ukázka tiskopisu Faktura za dávky dodávaného VZP.

D:\Dokumenty\Diplomka\VZP\Tiskopisy\FakturaZaObdobi.pdf (Local File) .
Ukázka tiskopisu Faktura za období dodávaného VZP.

D:\Dokumenty\Diplomka\VZP\Tiskopisy\PruvodniListDavky.pdf (Local File) .
Ukázka tiskopisu průvodního listu dávky dodávaného VZP.

D:\Dokumenty\Diplomka\VZP\Tiskopisy\PruvodniListDiskety.pdf (Local File) .
Ukázka tiskopisu průvodního listu diskety dodávaného VZP.

Uložit do databáze

public Případ užití: Uloží generované dávky do databáze.

Vytvořit nové dávky

public Případ užití: Generovaný soubor má standardní název KDAVKA.XXX, kde XXX představuje kód pojišťovny z číselníku Zdravotní pojišťovny. Soubor obsahuje buď jednu dávku nebo několik dávek řazených za sebou. Na každé disketě předávané pojišťovně smí být vždy pouze jeden soubor.

Associated Files

D:\Dokumenty\Diplomka\VZP\DRDokladu.pdf (Local File) .
Popis datového rozhraní dávek dokladů pro pojišťovnu.

D:\Dokumenty\Diplomka\VZP\DRDokladuPripravovaneZmeny.pdf (Local File) .
Popis připravovaných změn v datovém rozhraní dávek dokladů pro pojišťovnu.

D:\Dokumenty\Diplomka\VZP\DRDokladuZmeny.pdf (Local File) .
Popis změn datového rozhraní dávek dokladů pro pojišťovnu.

D:\Dokumenty\Diplomka\VZP\Metodika.pdf (Local File) .
Metodika pro pořizování a předávání dokladů VZP.

D:\Dokumenty\Diplomka\VZP\MetodikaZmeny.pdf (Local File) .
Změny v metodice v pořizování a předávání dokladů VZP.

Project Glossary

Item	Type	Meaning
Anamnéza	Business	Souhrn informací z určité oblasti usnadňující stanovení diagnózy. Je součástí vyšetřování pacienta lékařem. (rodinná, sociální, alergická, pracovní, atd.)
Charakter dávky	Business	Určuje, zda se jedná o dávku řádnou (obsahuje pouze původní doklady), nebo opravnou (obsahuje opravené doklady dříve odmítnuté).
Dávka	Business	Pomocná jednotka pro vyúčtování a předávání dokladů pojišťovně.
Dekurz	Business	Záznamy jednotlivých vyšetření, nálezů, neschopností, vystavených receptů a dalších informací do karty pacienta.
Diagnóza	Business	Rozeznání nemoci a její pojmenování. Číselné kódy jednotlivých diagnóz lze najít v mezinárodní klasifikaci nemocí (MKN).
Číselník	Business	viz. Globální číselník
Globální číselník	Technical	Seznam jednotlivých položek dodávaných pojišťovnou, pravidelně aktualizovaný. Typem položky obsažené v číselníku je dán i typ číselníku (Léky, Diagnózy, apod.) Přesný popis formátu souboru je dán datovým rozhraním pojišťovny.
IČP	Business	Identifikační číslo zdravotnického zařízení.
Kapitace	Business	Paušální platba za registrované pacienty, ve které jsou zahrnuté některé standardní výkony prováděné pro danou věkovou skupinu pacientů – již se neuvádějí ve vyúčtování výkonů za registrované pacienty.
Kapitační koeficient	Business	Koeficient, kterým se násobí základní hodnota kapitace. Koeficient je závislý na věku pacienta. Velikost kapitačního koeficientu pro jednotlivé věkové skupiny vydává pojišťovna v podobě číselníku.
Katalog	Technical	Seznam položek, které lékař často používá. Jednotlivé položky katalogu musí lékař do systému zadávat ručně. Datové rozhraní katalogu není nijak upřesněno. Katalogy, které lékař může v systému využít: Spolupracující lékaři, Adresy, Magistraliter.
Lékařská zpráva	Business	Dokument, kterým lékař předává jinému zdravotnickému zařízení informaci o výsledcích provedeného vyšetření nebo ošetření.
Lokální číselník	Technical	Výběr položek z globálního číselníku, které lékař často používá. Zjednodušuje lékaři vyhledávání často používaných položek.
LSPP	Business	Lékařská služba první pomoci.
Magistraliter	Business	Předpis pro přípravu léčiva v lékárně. Vhodné pro přípravky, které nejsou vyráběny farmaceutickými společnostmi.
Mezinárodní klasifikace nemocí (MKN)	Business	Mezinárodně platný seznam nemocí, úrazů a příčin smrti s přiřazenými čísly.
Recept	Business	Formulář, na který lékař pacientovi předepisuje léky nebo magistraliter. Na základě receptu vydá lék pacientovi lékárna. Na jednom receptu mohou být uvedeny maximálně dva druhy léků.
Sedimentace	Business	Vyšetření krve prováděné v laboratoří nebo přímo v ordinaci.
Typ dávky	Business	Určuje zda dávka obsahuje pouze doklady jednoho druhu (jednoduchá dávka) nebo obsahuje-li doklady více druhů (smíšená dávka).
Výměnný list	Business	Dokument, kterým lékař žádá jiné zdravotnické zařízení o provedení vyšetření nebo ošetření pacienta.
VZP	Business	Všeobecná zdravotní pojišťovna
ZP, ZUM	Business	Zdravotnické prostředky. Pomůcky nebo materiál, který lékař poskytne pacientovi. Vyúčtování takovýchto prostředků se provádí spolu s vyúčtování zdravotních výkonů pojišťovně.
ZUM	Business	viz. ZP