Unified Modeling Language

Unified Modeling Language

- Unified=zjednotený
  - Three Amigos: Booch, Rumbaugh, Jacobson
- Modeling=modelovací
  - grafický
  - vizuálny
- Language=jazyk
  - syntax
  - sémantika

UML - definícia

- Object Management Group (OMG):
  - "The Unified Modeling Language (UML) is a graphical language for **visualizing, specifying, constructing, and documenting** the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."

UML - história

- 1996: verzia 0.9 – pridávajú sa IBM, HP, MS, Oracle,...
- 1997: UML štandardizovaný - Object Management Group (OMG), verzie 1.0 a 1.1
- 1998-2001: verzie 1.2, 1.3, 1.4 – malé zmeny
- ISO/IEC 19501:2005 (UML verzia 1.4.2)
- 2005: verzia 2.0
- 2011: verzia 2.4.1
- od decembra 2017: v2.5.1 - (súčasnosť)

UML - odkazy

- www.uml.org
- www.omg.org
  - špecifikácia: https://www.omg.org/spec/UML/2.5
- www.uml-diagrams.org

Diagramy v UML 2.x

- **Structural UML diagrams (Diagramy štruktúr)**
  - **Class diagram**
  - **Component diagram**
  - Composite structure diagram
  - Deployment diagram
  - Object diagram
  - Package diagram
  - Profile diagram
- **Behavioral UML diagrams (Diagramy chovania)**
  - **Activity diagram**
  - **State machine diagram**
  - **Use case diagram**
- **Interaction diagrams (Diagramy interakcií)**
  - **Interaction overview diagram**
  - **Sequence diagram**
  - Timing diagram

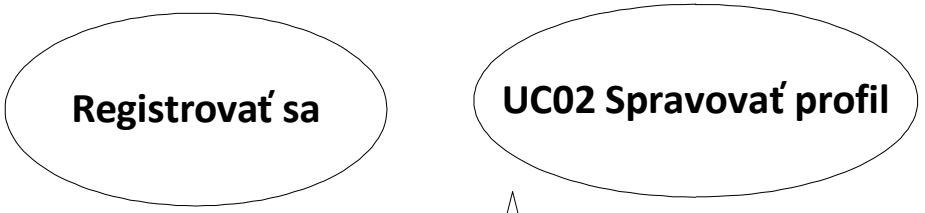
Use Case Diagram

(diagram prípadov použitia)

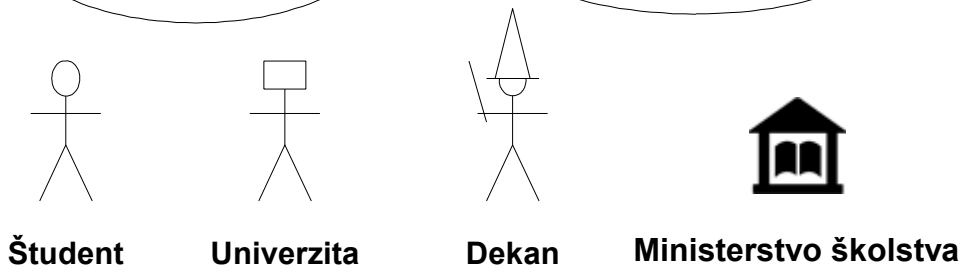
- **Význam**
  - popisuje správania navrhovaného systému z pohľadu používateľov
- **Komu je určený**
  - pre vývojárov aj manažérov
- **Popis**
  - diagram dáva informáciu o tom, kto (aktér) môže vykonávať ktoré činnosti (prípady) v systéme.
  - úlohou diagramu je identifikovať “všetky” prípady použitia a priradiť ich k jednotlivým používateľom.
  - diagram **nezachytáva** procesy v systéme ako je napr. kauzalita jedného prípadu (aktivity) na druhom prípade v rámci zoznamu prípadov aktéra. To sa modeluje v inom diagrame (diagram aktivít).

Syntaktické prvky Use Case diagramu

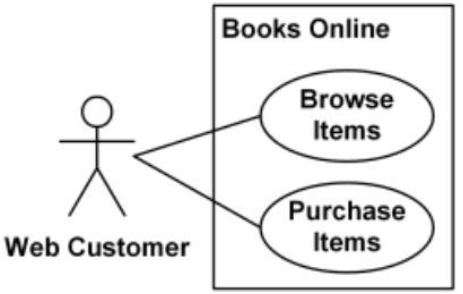
- Prípad



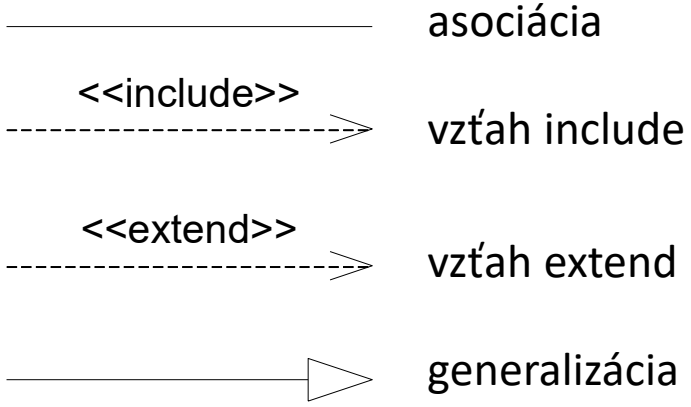
- Aktéri



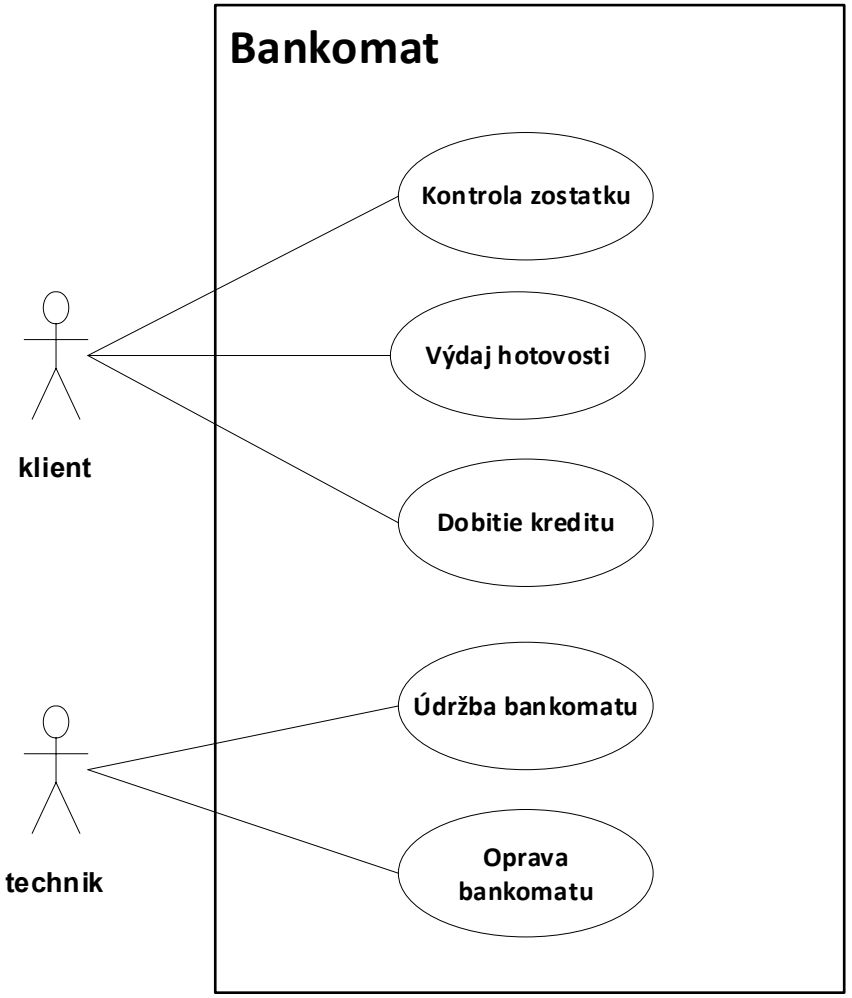
- Ohraničenie diagramu



- vzťahy

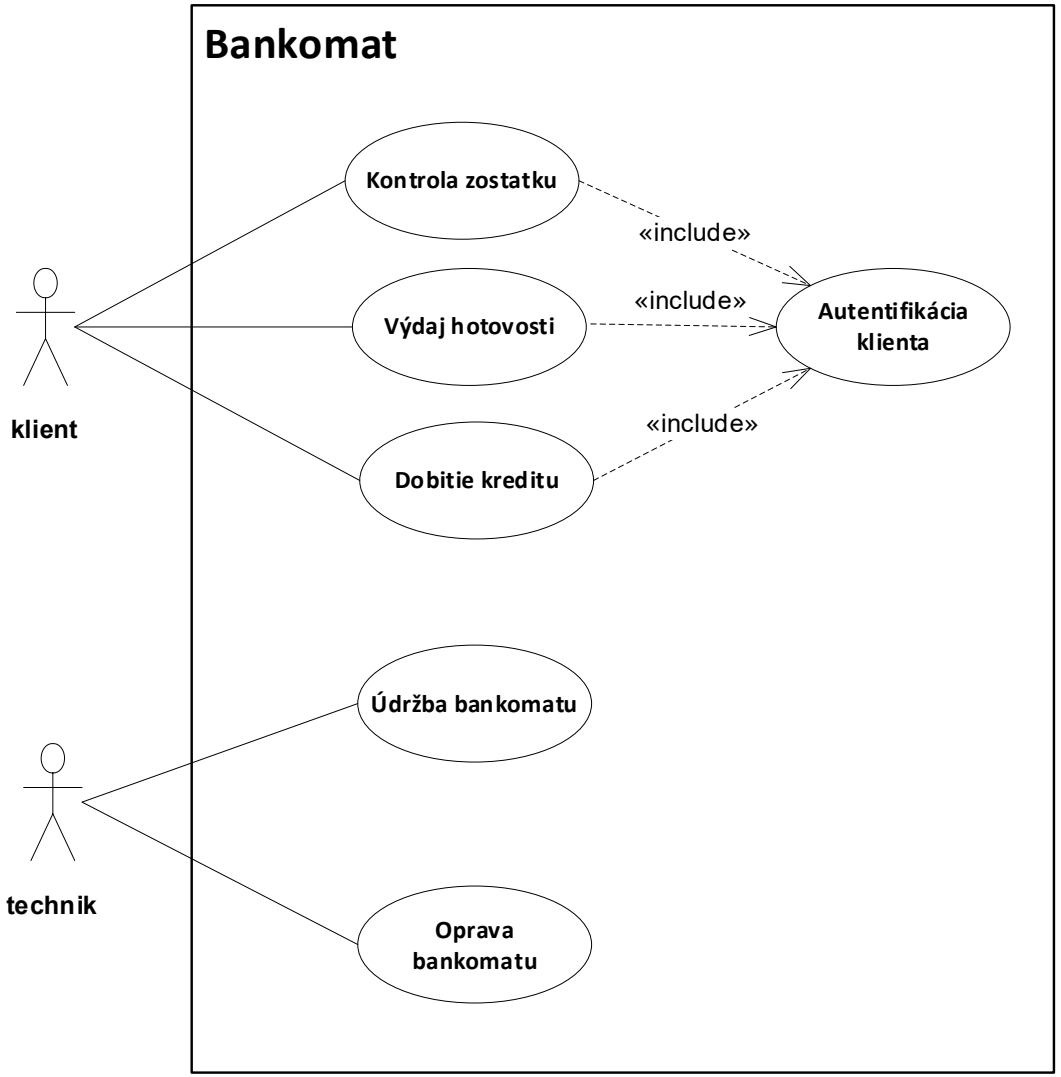


Príklad1a: asociácia



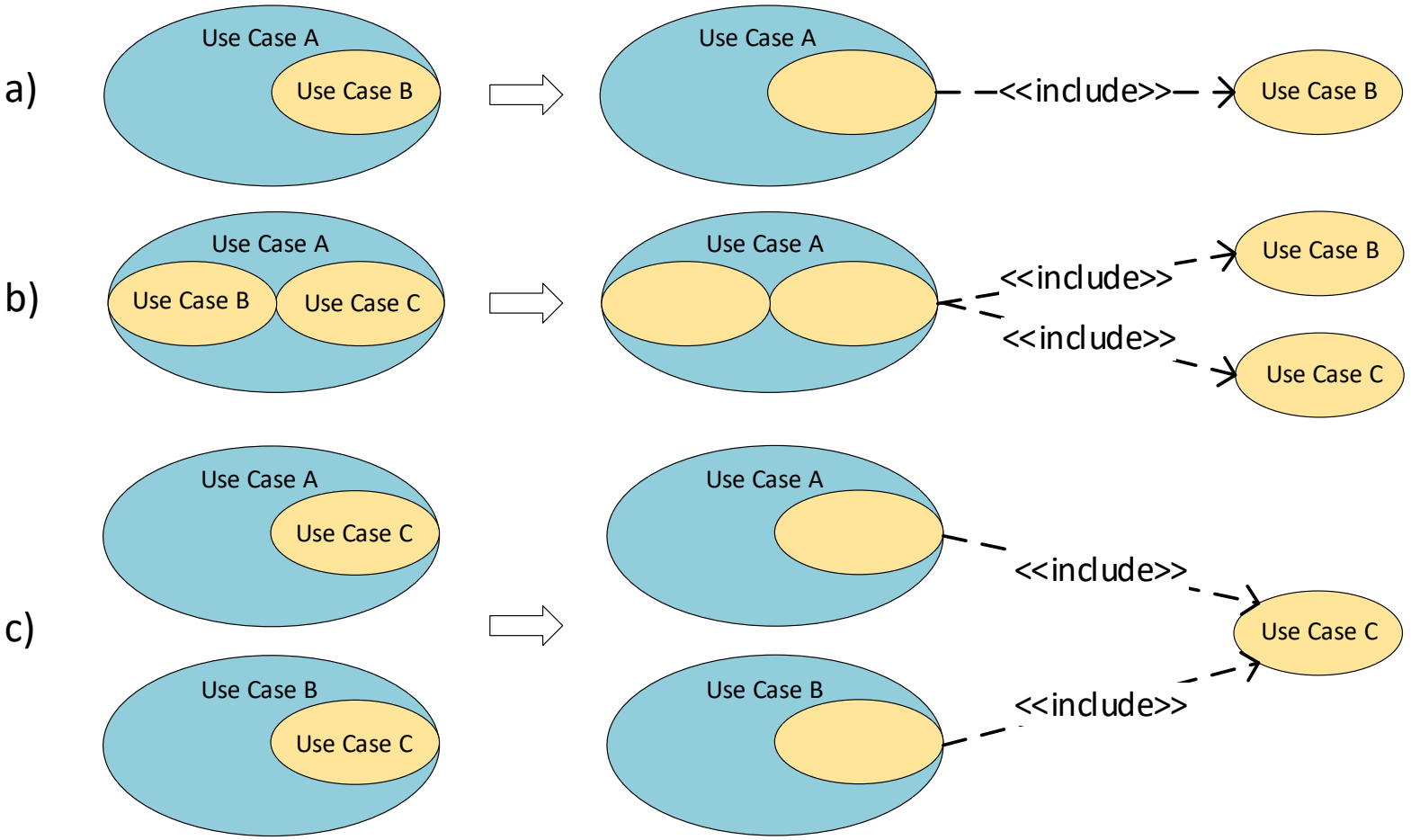
- Asociácia sa **kreslí** iba medzi **aktérom** a **prípadom**
- Asociácia sa **nikdy nekreslí** medzi jednotlivými prípadmi

Príklad1b: include

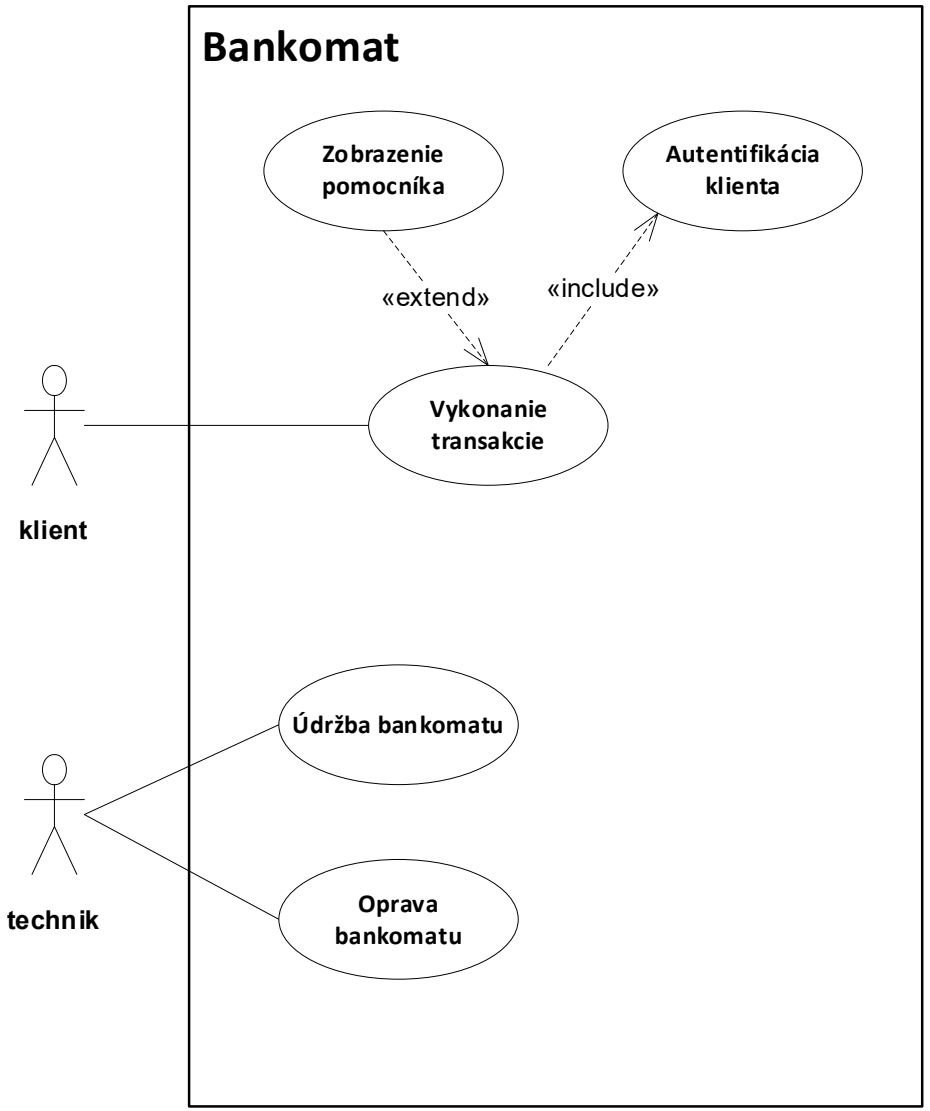


- Rozdelenie komplexnejších prípadov na jednoduchšie (vedľajšie), hlavne z dôvodu znovu-použitia
- Vedľajšie prípady, ktoré sú prepojené cez väzbu «include» sú nevyhnutnou súčasťou hlavného prípadu
- Jeden prípad je zahrnutý v ďalších prípadoch

Možnosti použitia „include“

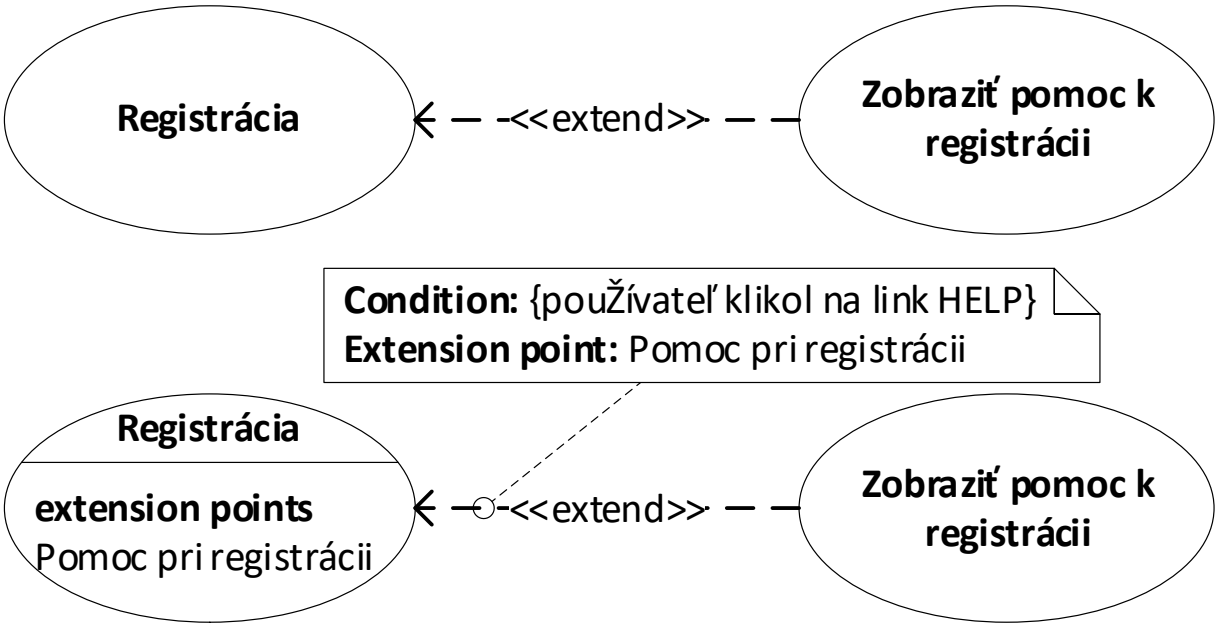


Príklad1c: extend



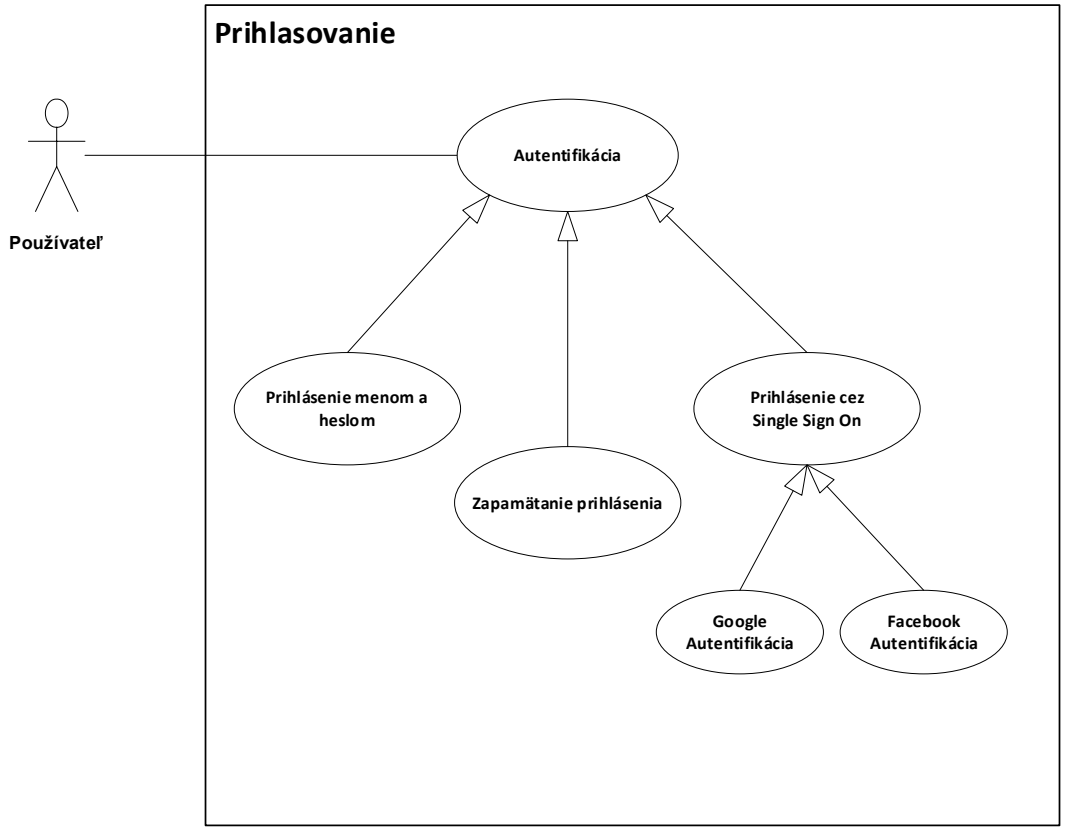
- Pomocou extend vyjadrujeme doplnkové, „nepovinné“, správanie, ktoré nemusí byť nutne súčasťou hlavného prípadu “Vykonanie transakcie”
- Vykonanie rozširujúceho prípadu môže byť definované podmienkou

Príklad1d: extension points



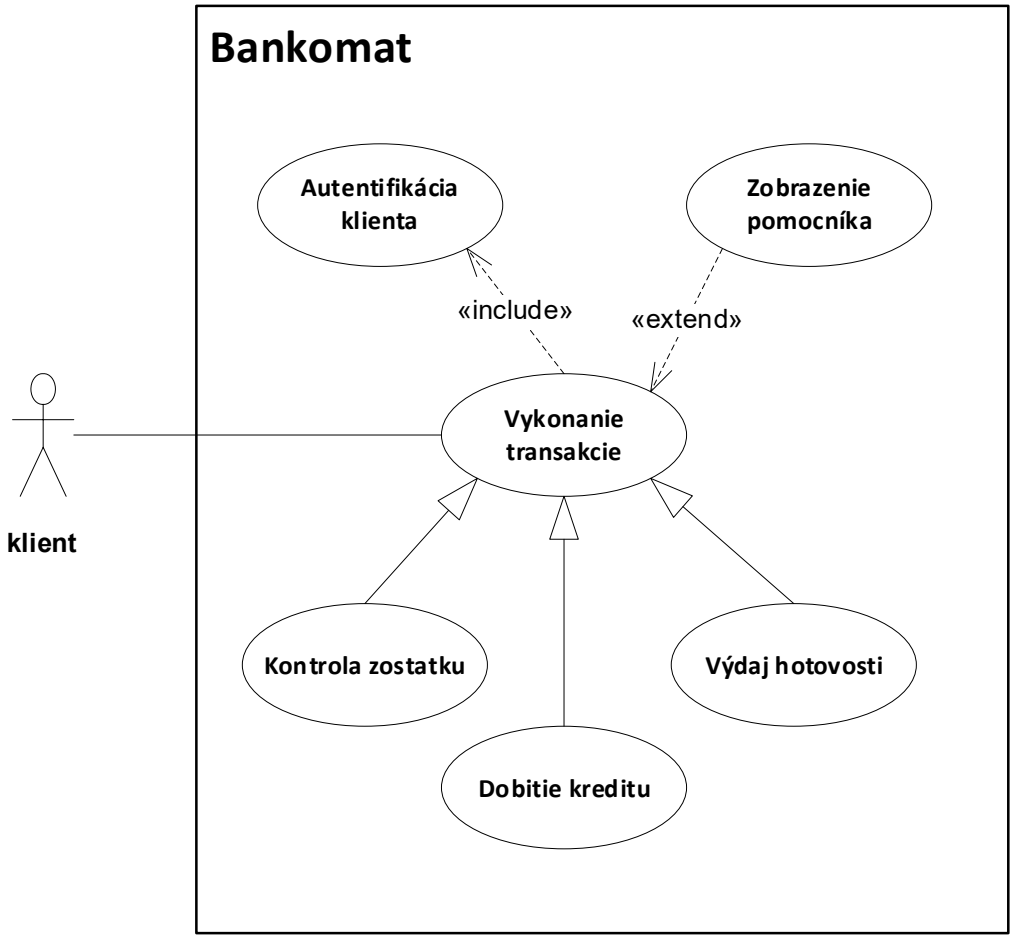
- Extension point – označenie prípadu, ktorý je možné rozšíriť pomocou vzťahu <<extend>> ďalšími (rozširujúcimi) prípadmi
- Je možné pridať podmienku pre vykonanie rozširujúceho prípadu

Príklad1f: generalizácia medzi prípadmi



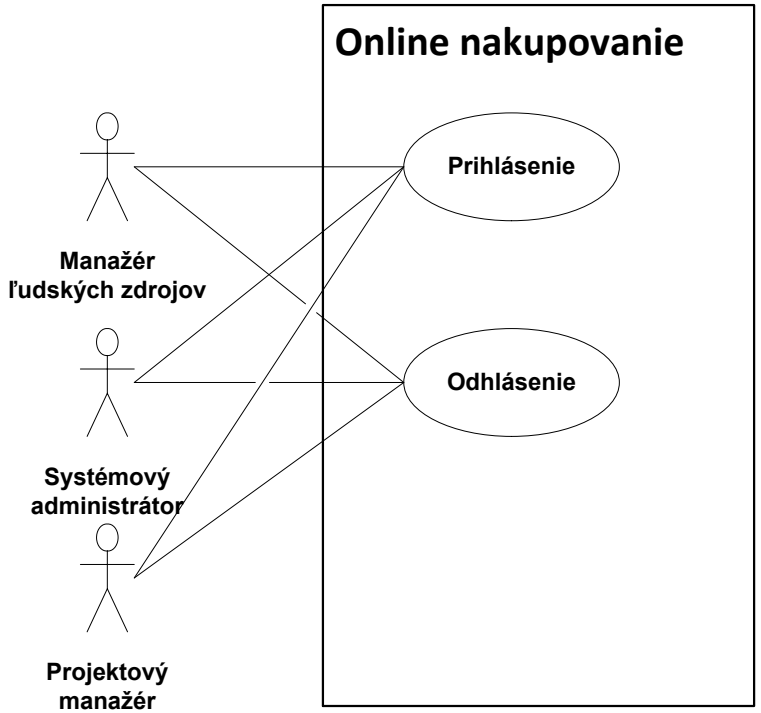
- Rodičovský prípad:
  - Autentifikácia
- Detský prípad:
  - Prihlásenie menom a heslom
  - Zapamätanie prihlásenia
  - Prihlásenie cez Single Sign On
- Detské prípady zdedili celé správanie rodičovského prípadu
- Medzi detskými prípadmi platí “XOR logika”

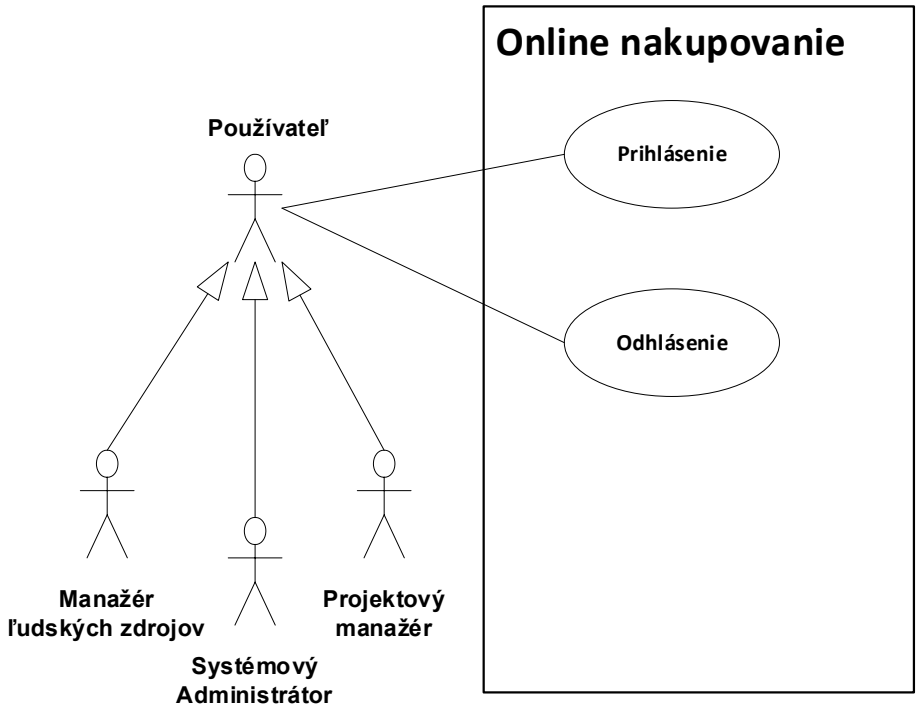
Príklad1g: generalizácia medzi prípadmi



- Detské prípady zdedili celé správanie rodičovského prípadu (aj Autentifikáciu klienta aj Zobrazenie pomocníka)
- V rámci vykonanej transakcie je možné vykonať iba jeden konkrétny typ transakcie (operácia XOR)

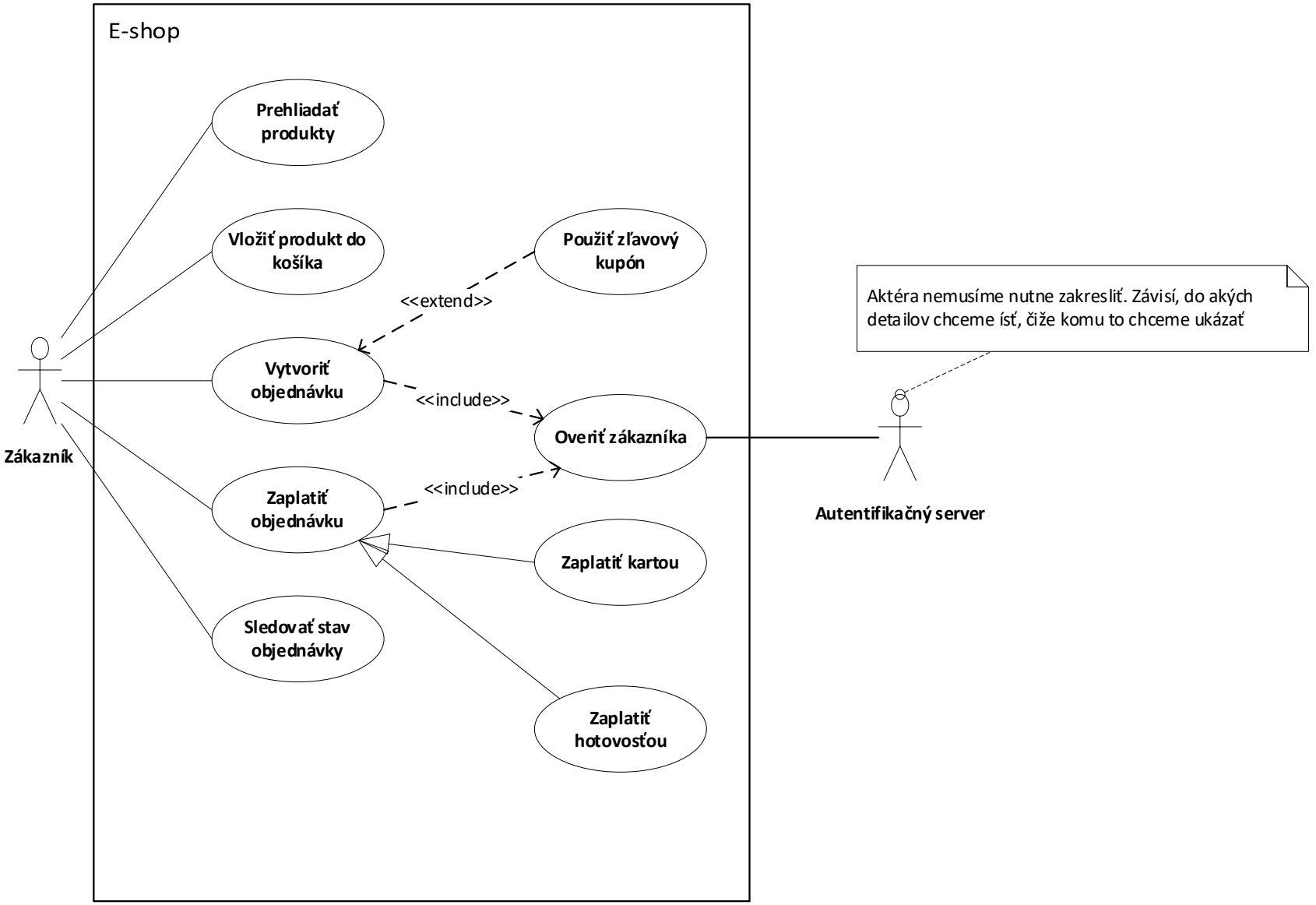
Príklad1h: generalizácia medzi aktérmi





- Využíva sa častejšie ako generalizácia medzi prípadmi
- Sprehľadňuje celý diagram

Príklad1i – E-shop



Najčastejšie chyby pri tvorbe Use Case diagramu

- Nesprávne identifikovanie prípadov ako „položiek menu“ (functional decomposition)
- Príliš veľa alebo príliš málo detailov
- Málo výstižné pomenovania
- Nesprávne používanie vzťahov (include, extend, generalizácia)
- Problém tzv. CRUD funkcionalít

Problém dekompozície funkcií

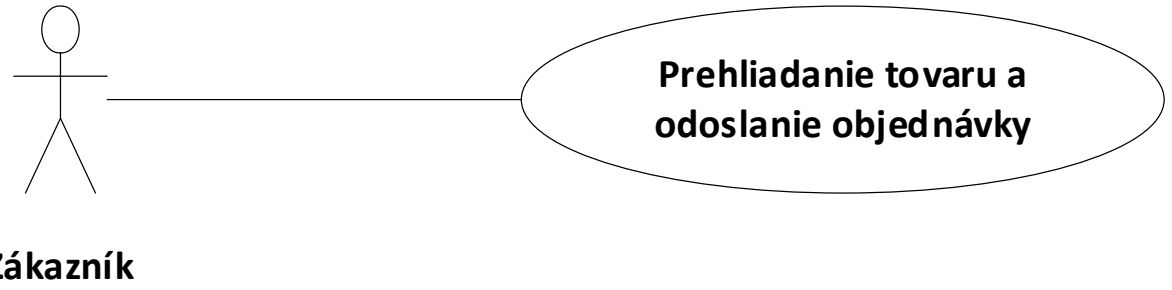
- Tieto prípady predstavujú čiastkovú funkcionalitu systému
- Sú to len izolované činnosti
- **Je to v poriadku?**

Problém dekompozície funkcií

NESPRÁVNE

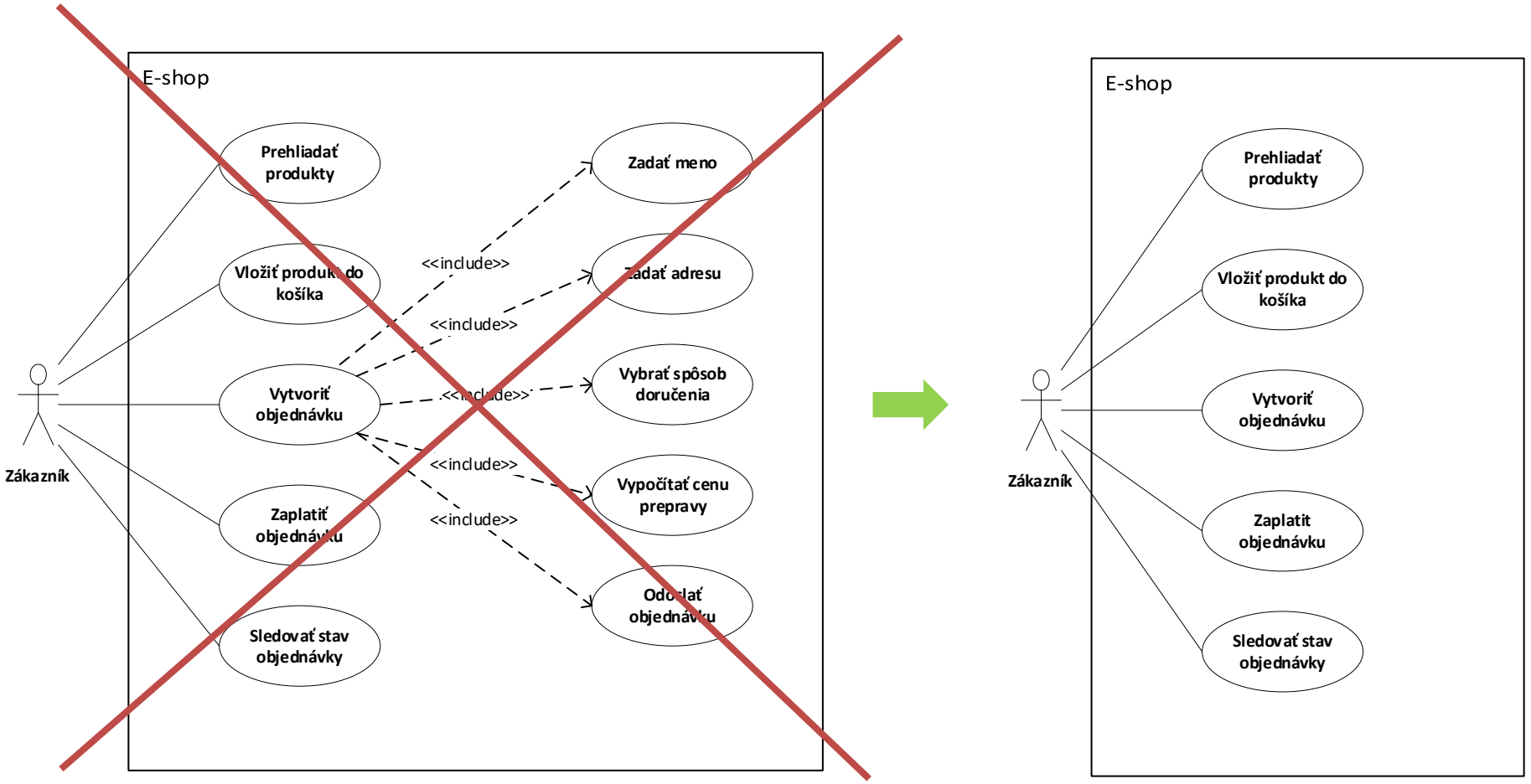
- Účel Use case diagramu:
- zobraziť vzťahy medzi aktérmi a ich cieľmi a nie medzi internými operáciami
- Prípady síce sú užitočné, ale až vtedy, keď zákazník vytvoril objednávku = to je jeho cieľ v eshope.

Problém dekompozície funkcií



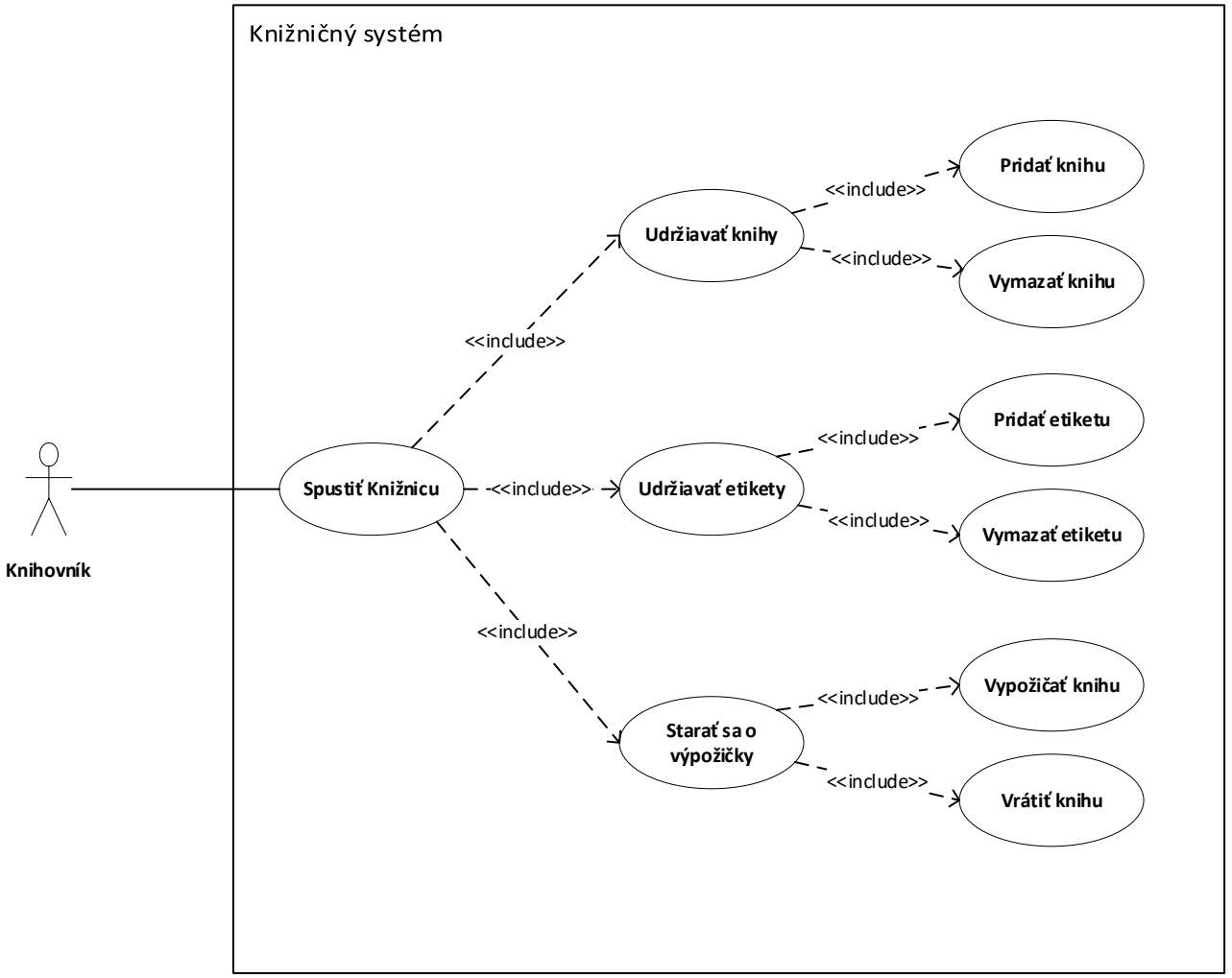
- Teraz diagram vyjadruje to, čo Zákazník chce robiť (nakupovať) a nie ako je systém rozdelený na čiastkové funkcionality, ktoré sú od seba izolované.

Dekompozícia funkcií: ďalší príklad



- diagram má popisovať ciele používateľa a nie to, aké kroky (funkcie) systém vykonáva vo vnútri => to môžeme znázorniť iným typom diagramu

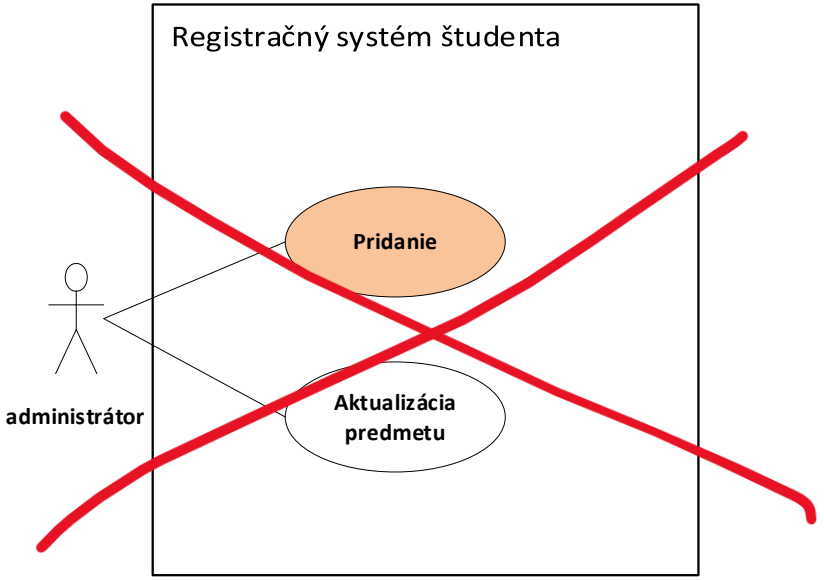
Dekompozícia funkcií: ďalší príklad

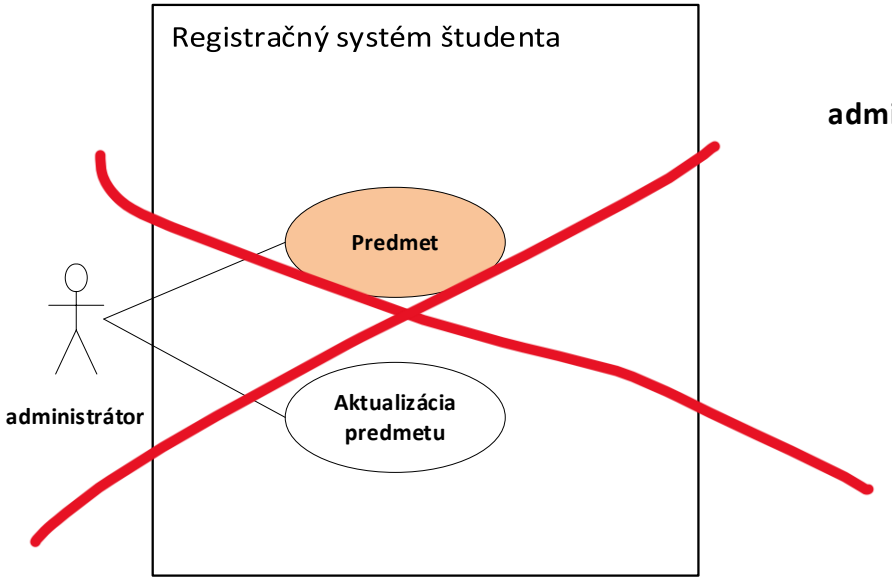


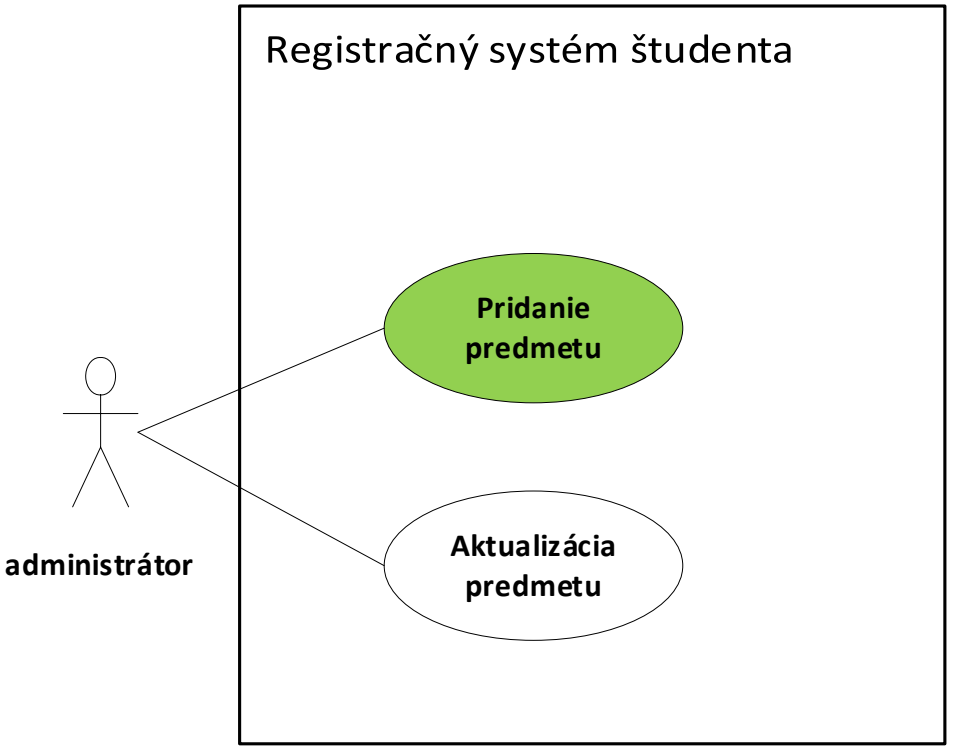
NESPRÁVNE – celé je to zle

- Chyby:
- nezachytávajú sa požiadavky, ale štruktúra požiadaviek = dekompozícia funkcií
- Posledné prípady sú zaujímavé, ostatné ich len volajú a nie sú pre model prínosné
- príliš abstraktné a mätúce názvy = Spustiť Knižnicu, Udržovať knihy
- je prirodzené vytvárať hierarchie, ale max. do úrovne 2, tu sú až 3
- celý model nikdy nemôže „prameniť“ z jedného prípadu

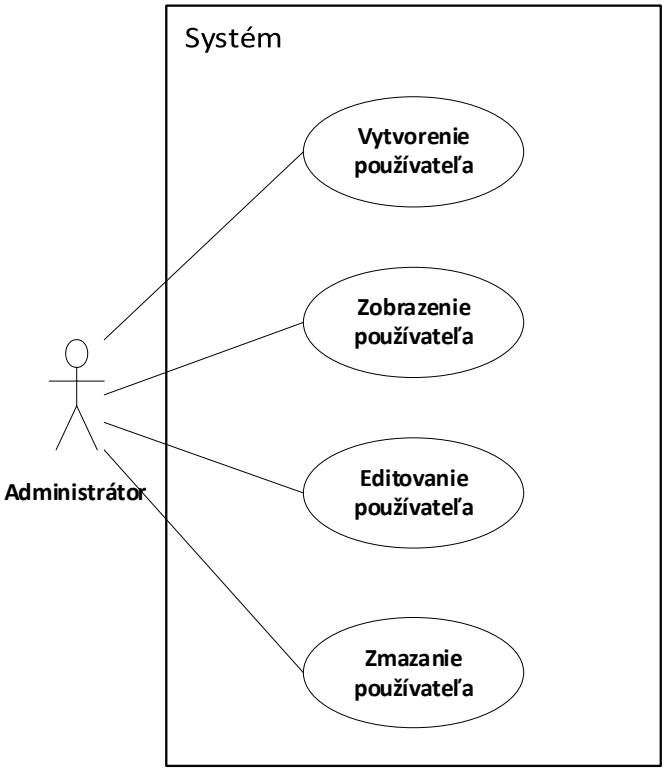
Málo výstižné pomenovania

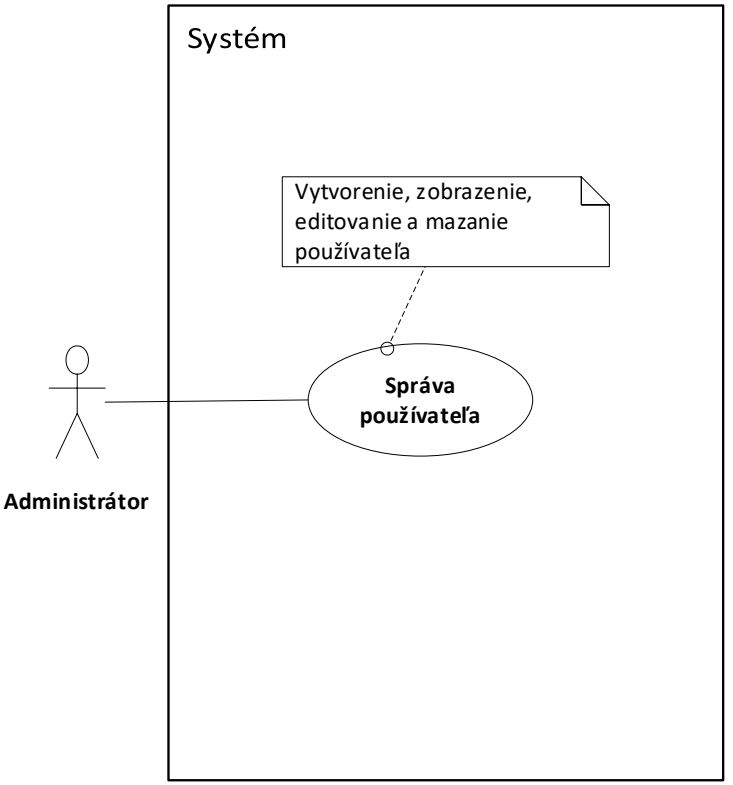






Problém činností CRUD





Radšej NIE

OK