MASARYKOVA UNIVERZITA

Modelování informačních systémů s využitím jazyka UML

Jaroslav Šmarda

Využití jazyka UML při vývoji IS na příkladu jednoduché aplikace pro evidenci knih

- Model IS
- Modelování případů užití
  - Diagram případů užití
- Modelování procesní struktury
  - Diagram aktivit
- Modelování logické struktury
  - Diagram tříd

Techniky pro modelování IS a systémů

- Při tvorbě systému se používá široké spektrum technik pro různé fáze životního cyklu vývoje:
  - od čistě formálních (např. Petriho sítí), jejichž cílem je jednoznačnost a úplnost,
  - přes poloformální techniky (např. jazyk UML - standardní jazyk pro zobrazení, specifikaci, konstrukci a dokumentaci prvků systémů nejčastěji se softwarovou charakteristikou),
  - až po neformální techniky (textový popis).

Model IS

- **Model** je abstrakce reálné věci
- Model je zjednodušení reality
- Potřebujeme jazyk pro popis modelu
- **Modelovací jazyk**:
  - kód programovacího jazyka, obrázky, diagramy, dlouhé textové popisy,
  - cokoliv, co pomůže popsat systém

Modelovací jazyk a meta-model jazyka

- **Notace**:
  - prvky, které tvoří modelovací jazyk
- **Sémantika** modelovacího jazyka:
  - Popis, co znamená notace
- Meta-model jazyka:
  - Obsahuje kompletní sémantiku jazyka

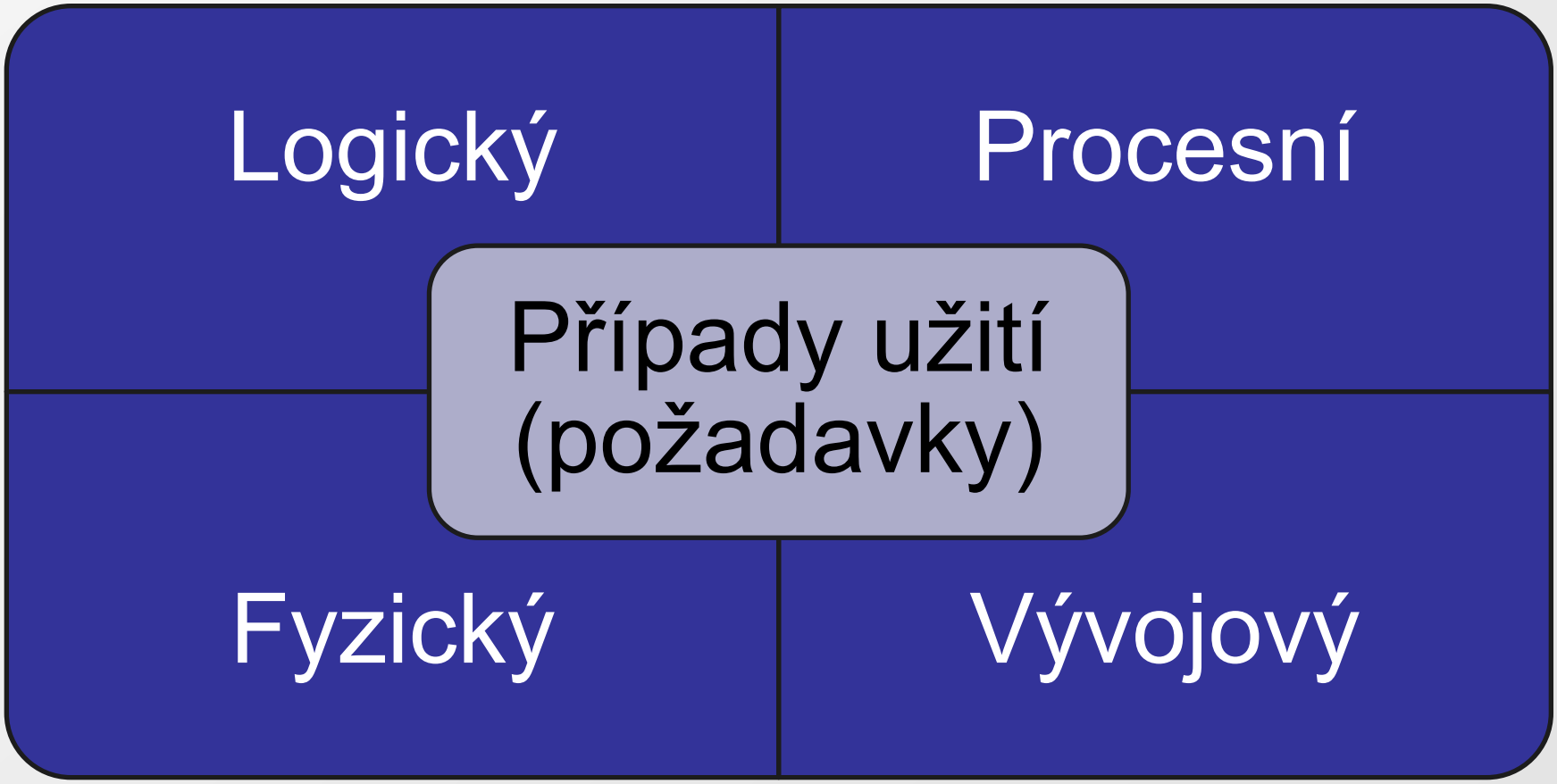
UML – Unified Modeling Language

- Výhody:
  - Formální jazyk
    - každý prvek má přesný význam (přesná notace)
  - Stručný
    - jednoduchá grafická notace
  - Obsažný
    - popisuje všechny důležité aspekty systému
  - Škálovatelný
  - Vytvořen na základě zkušeností
  - Standard
    - verze 2.0

Modelem řízená architektura (MDA – Model-Driven Architecture) vývoje IS

- UML může být základem vývoje IS řízeného modelem:
  - Na základě modelu je generován systém
- Snaha o efektivní vývoj
- Upřednostňována definice modelu IS (v UML) před specifikací algoritmů
  - Algoritmy standardní (parametrizovány)
- Model-driven engineering

Pohledy na model IS



Pohledy na model IS

- Případy užití (Use Case):
  - Funkcionalita systému z vnějšího pohledu
  - Případ užití – uživatelský příběh (User Story)
- Logický pohled:
  - Abstraktní popis částí systému – objekty, třídy objektů
- Procesní pohled:
  - Diagramy aktivity
- Fyzický pohled:
  - Diagramy nasazení
- Vývojový pohled:
  - Diagramy komponent

Diagram případů užití

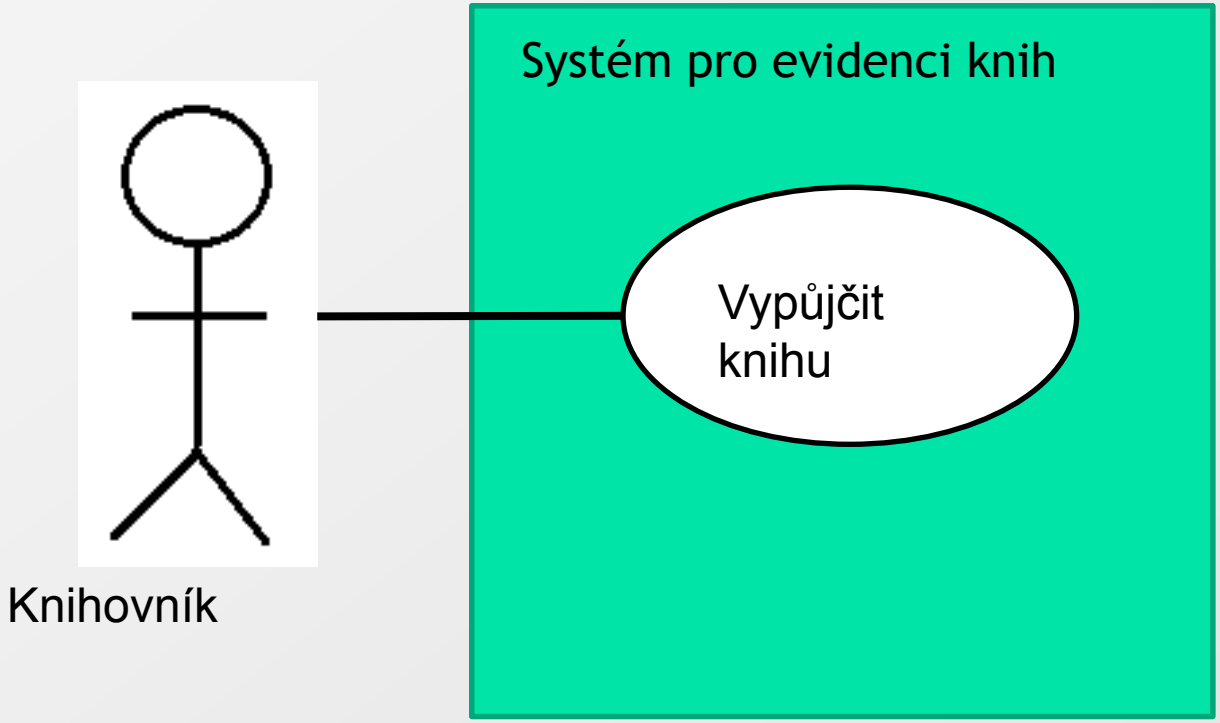
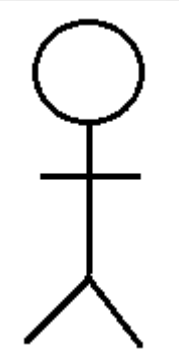


Diagram případů užití

- Aktor (Actor – herec):
  - Modeluje okolí systému
- Aktor:
  - 1. osoba interagující se systémem
  - 2. nebo něco, co nelze ovlivnit v návrhu systému



Knihovník

<< actor>>
Knihovník

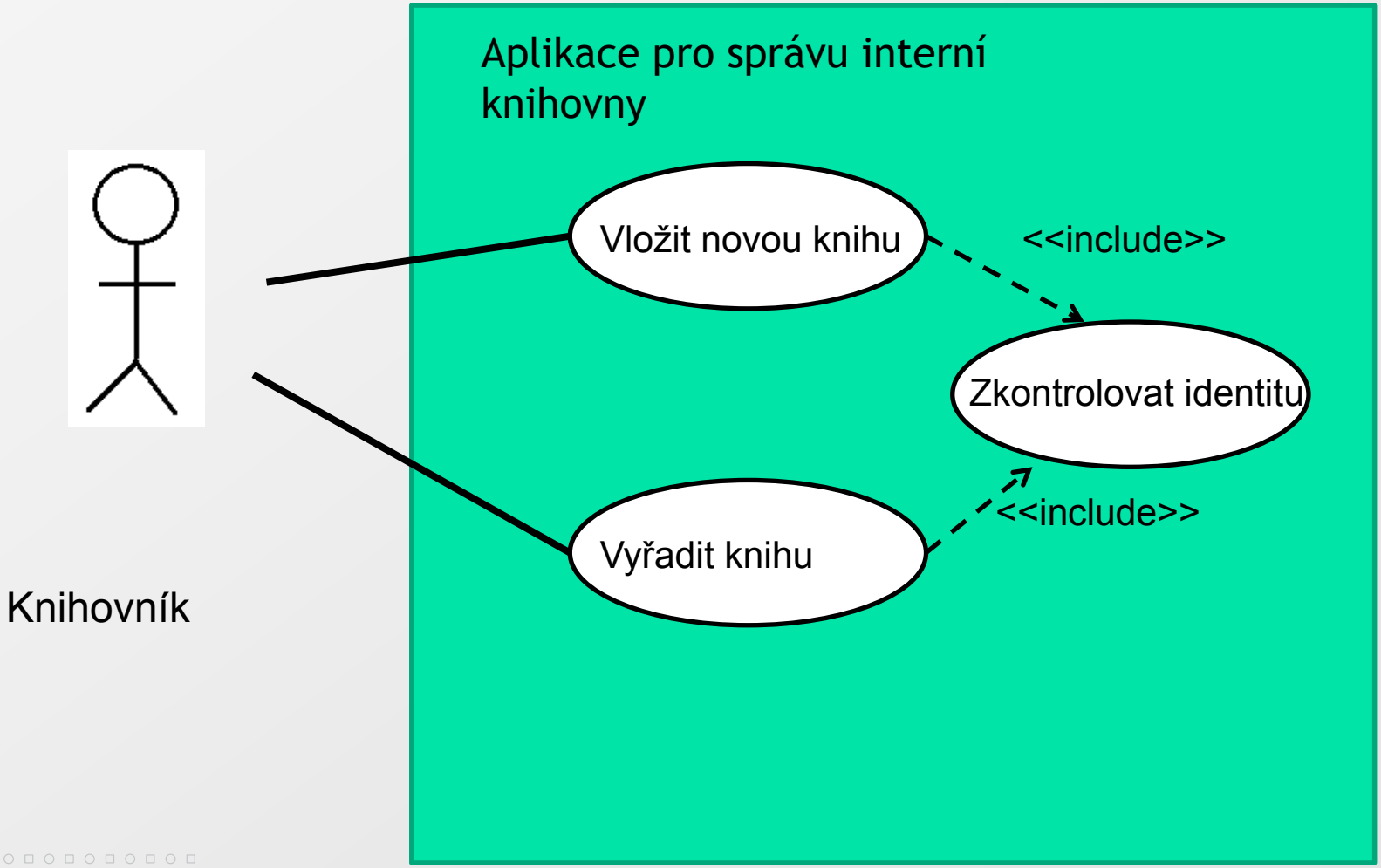
Vztahy mezi případy užití

- Zahrnutí případu užití (<<include>>)
- Specializace případu užití
- Rozšíření případu užití (<<extend>>)

Vložení případu užití <<include>>

- Strukturování případů užití do samostatných částí
- Include = obsahovat, zahrnovat
- Podpora opakovaně použitelného kódu
- V programovacích jazycích jsou **podprogramy** nebo v některých (jazyk C) **funkce**

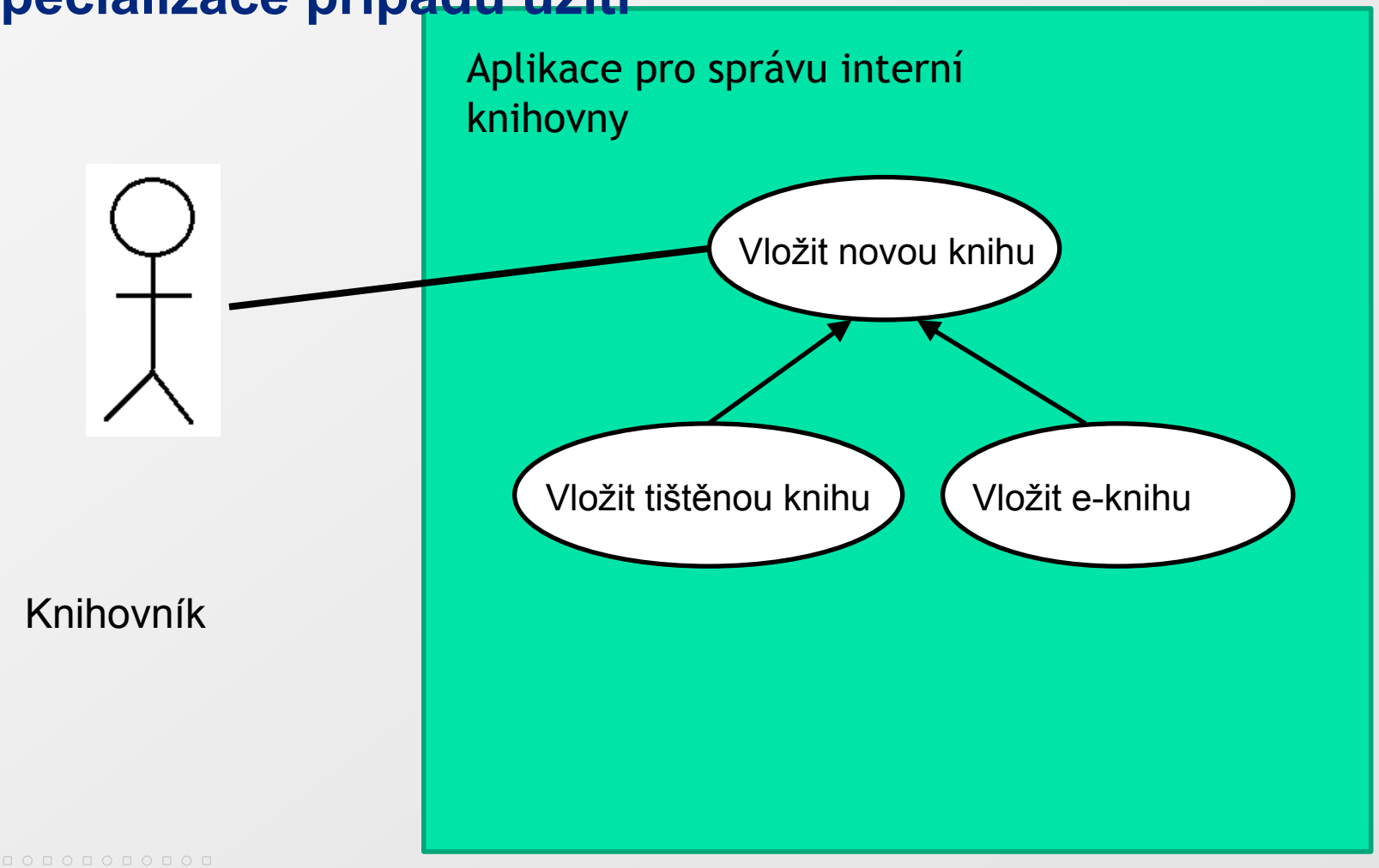
Použití <<include>>



Detailní popis případu užití

Název případu užití	Vložit nového čtenáře
Vazba na požadavky	K.1
Cíl (v kontextu)	Knihovník vytváří záznam nového čtenáře
Předpoklady	Knihovník je identifikován a autorizován pro vkládání záznamů
Úspěšný koncový stav	Záznam nového čtenáře je vytvořen
Neúspěšný koncový stav	Záznam čtenáře již existuje
Primární aktor	Knihovník
Sekundární aktor	Není
Start	Čtenář, který dosud nemá záznam v seznamu čtenářů, si chce vypůjčit knihu

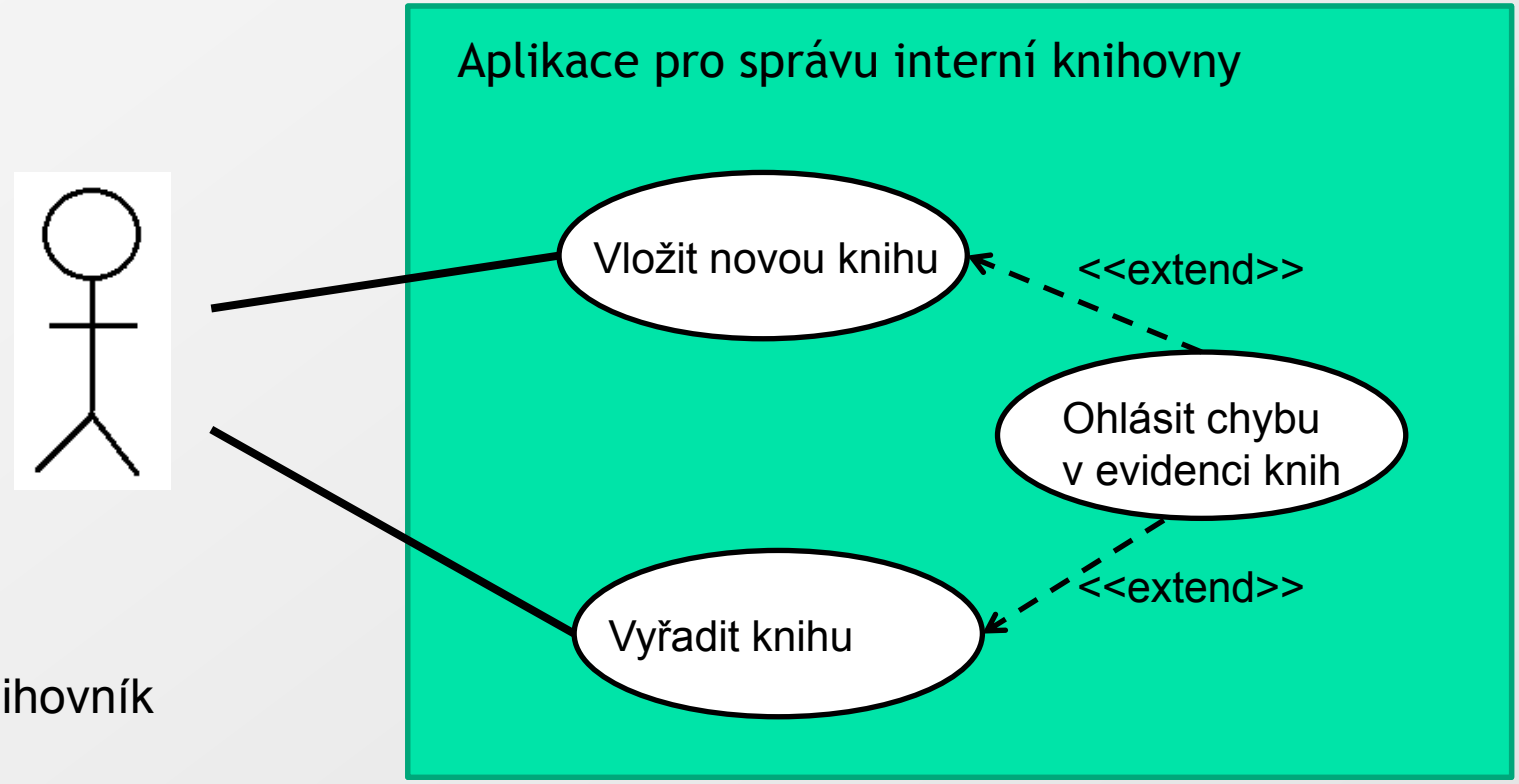
Specializace případu užití



Specializace případu užití

- Vložit knihu
  - Vložení všech údajů společných tištěným knihám i e-knihám
- Vložit tištěnou knihu
  - Vložit údaj, zda se jedná o brožovanou nebo vázanou knihu
- Vložit e-knihu
  - Vložit údaj o formátu e-knihy (PDF, ePub, AmazonKindle, Nook,…)

Rozšíření případu užití <<extend>>

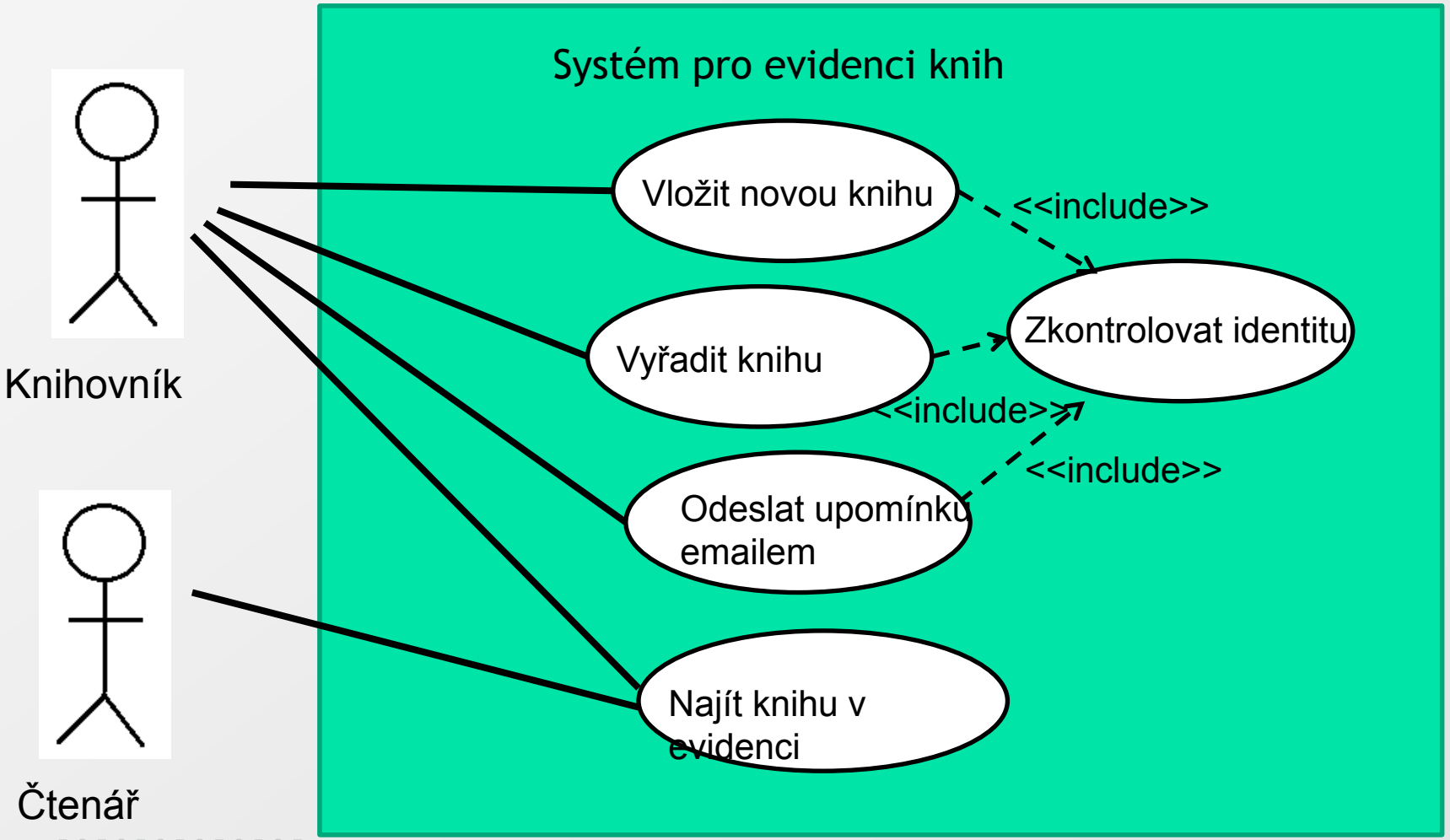


- Ohlásit chybu v evidenci knih:
  - Vkládaná kniha už existuje nebo vyřazovaná kniha neexistuje

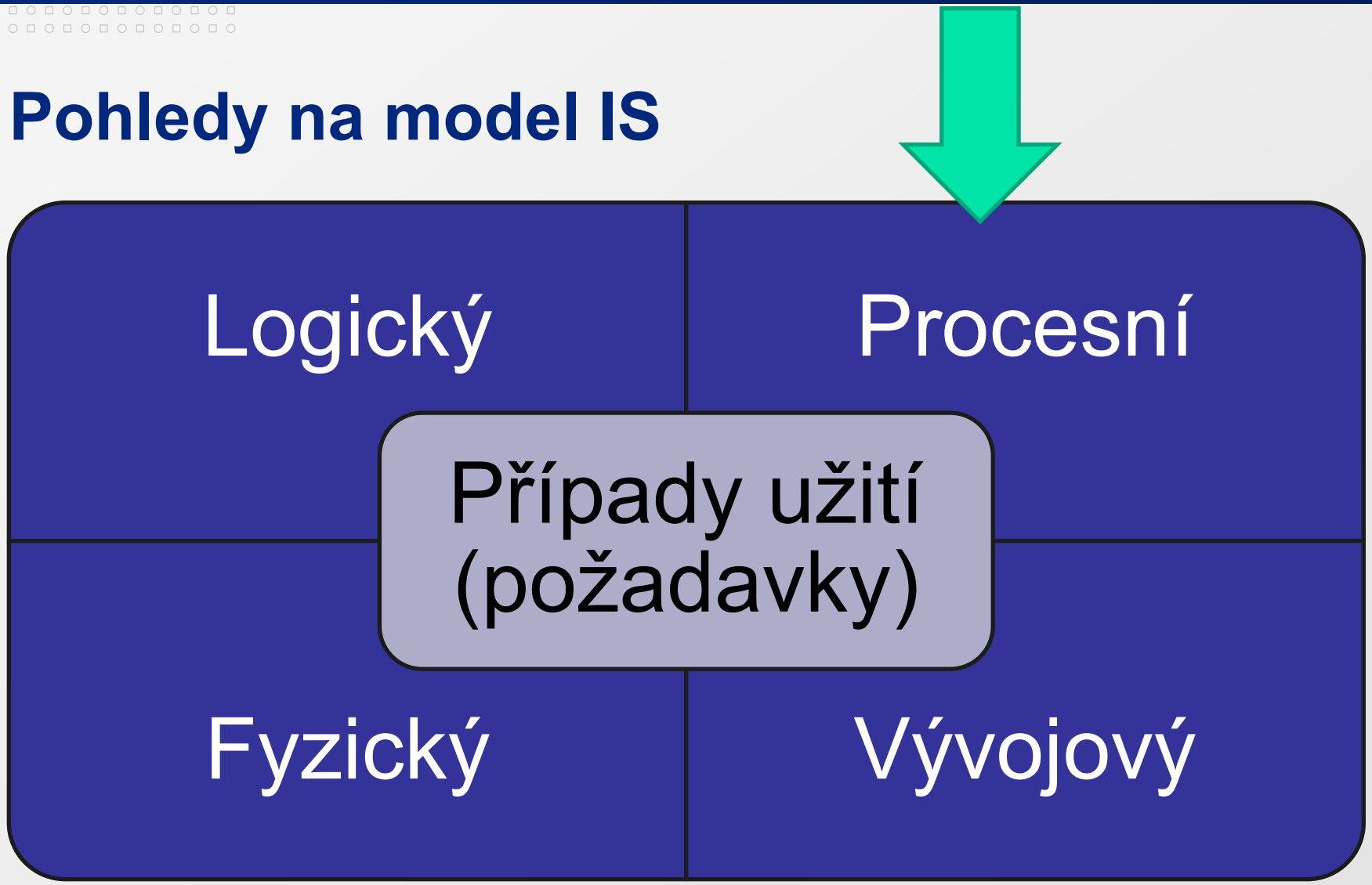
Rozšíření případu užití <<extend>>

- Obohacení případu užití o chování jiného případu užití, které nenastane vždy
- Na rozdíl od rozšíření <<extend>> zahrnutí <<include>> nastává vždy

Příklad diagramu případů užití



Pohledy na model IS



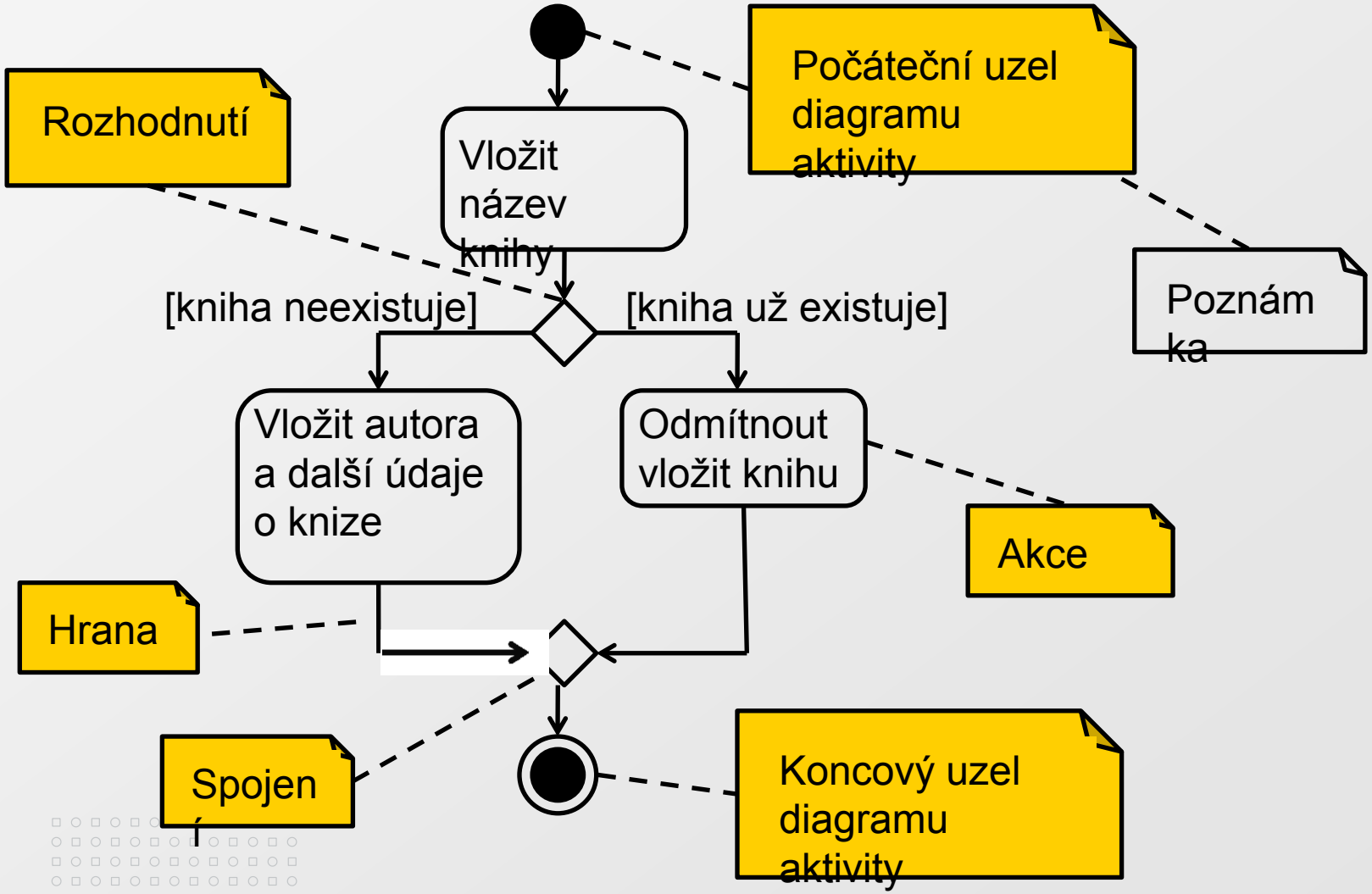
Procesní pohled: Diagram aktivity

- Případy užití specifikují **CO** má systém dělat
- Diagram aktivity specifikuje **JAK** to dělá

Diagram aktivity

- Diagram aktivity se zvláště hodí pro modelování podnikových procesů (Business Processes)
- Diagramy aktivit využívají nástroje pro BPM (Business Processes Management)
- Diagram aktivity vychází z:
  - vývojových diagramů,
  - diagramu datových toků a
  - Petriho sítí

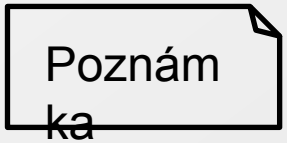
Diagram aktivity případu užití



Základní prvky diagramu aktivity

	Počáteční uzel (Initial node)
	Koncový uzel (Final node)
	Nestandardní konec

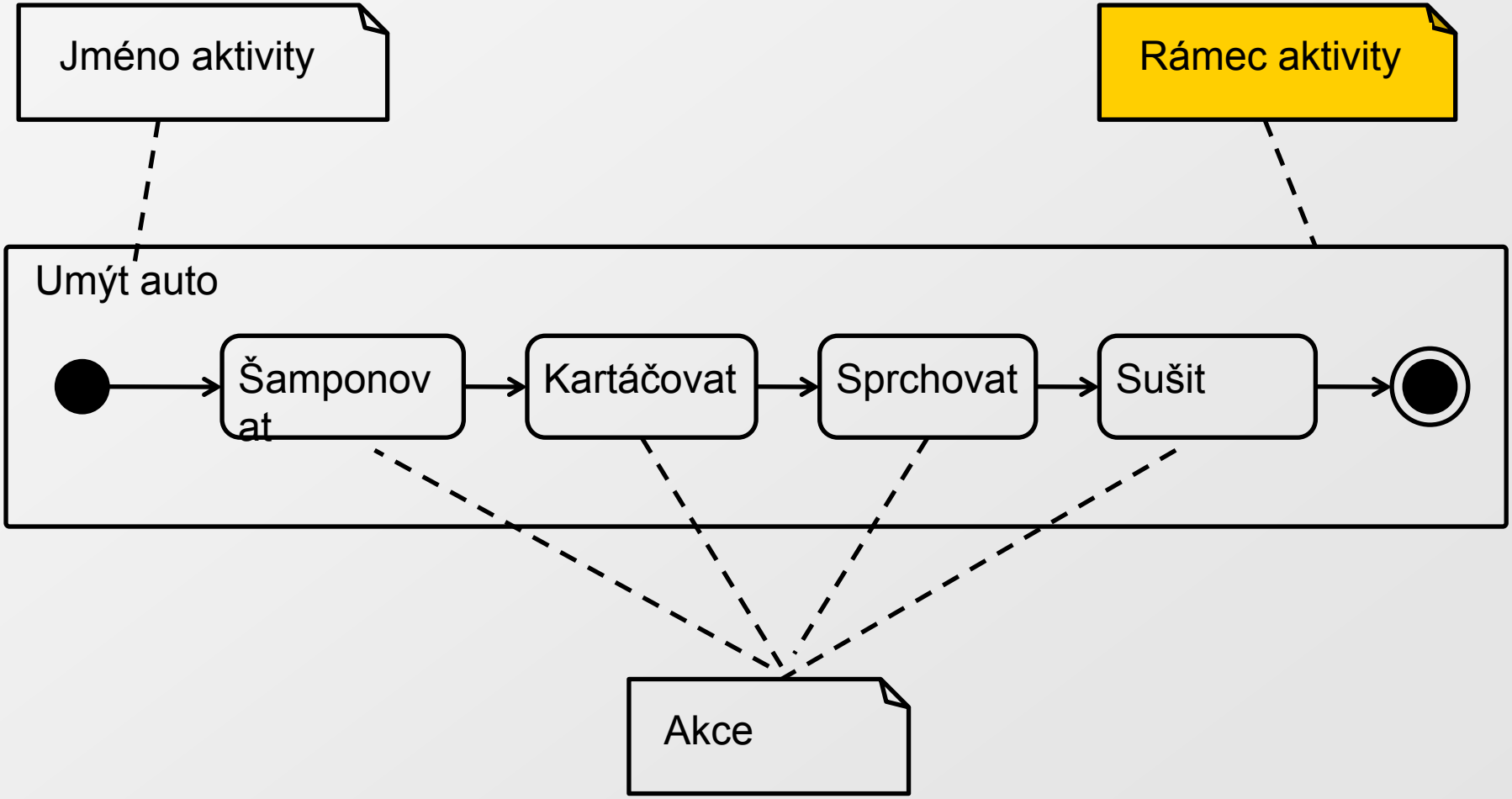
Základní prvky diagramu aktivity

	Akce (Action)
	Rozhodnutí (Decision) nebo spojení (Merge)
	Hrana (Edge)
	Poznámka

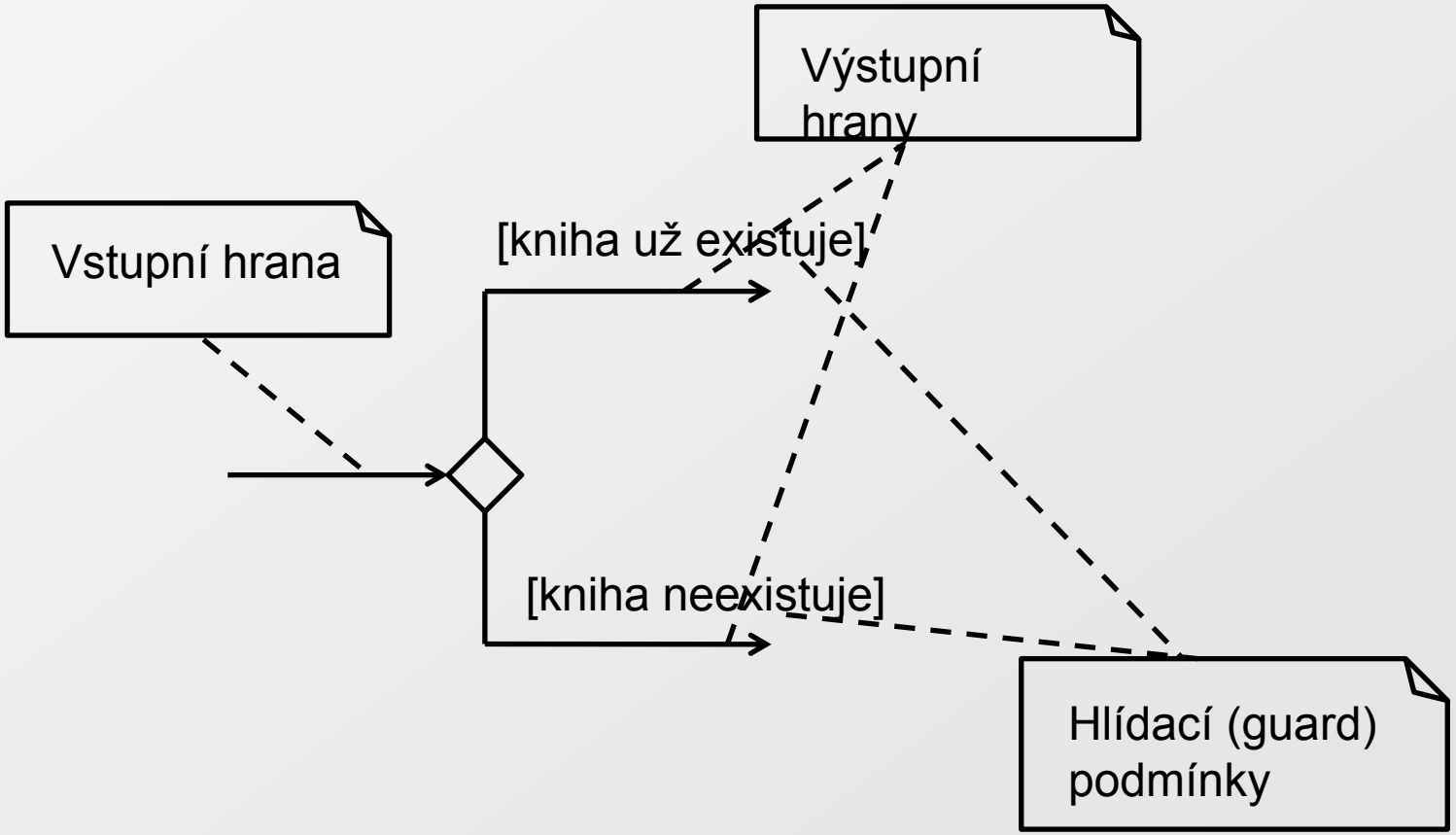
Akce a aktivity

- Aktivita:
  - obvykle odpovídá případu užití
- Akce:
  - Aktivní kroky v procesu aktivity

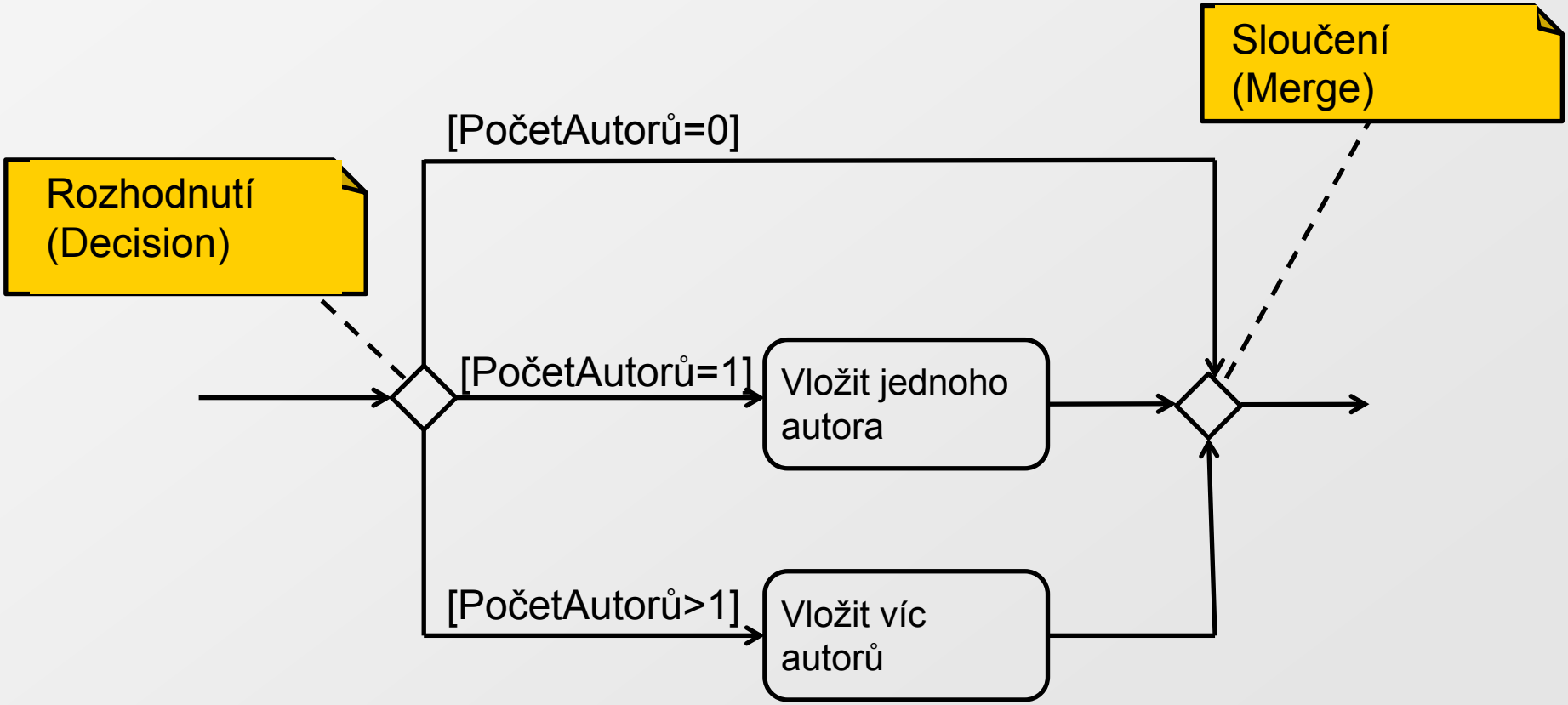
Více aktivit v jednom diagramu



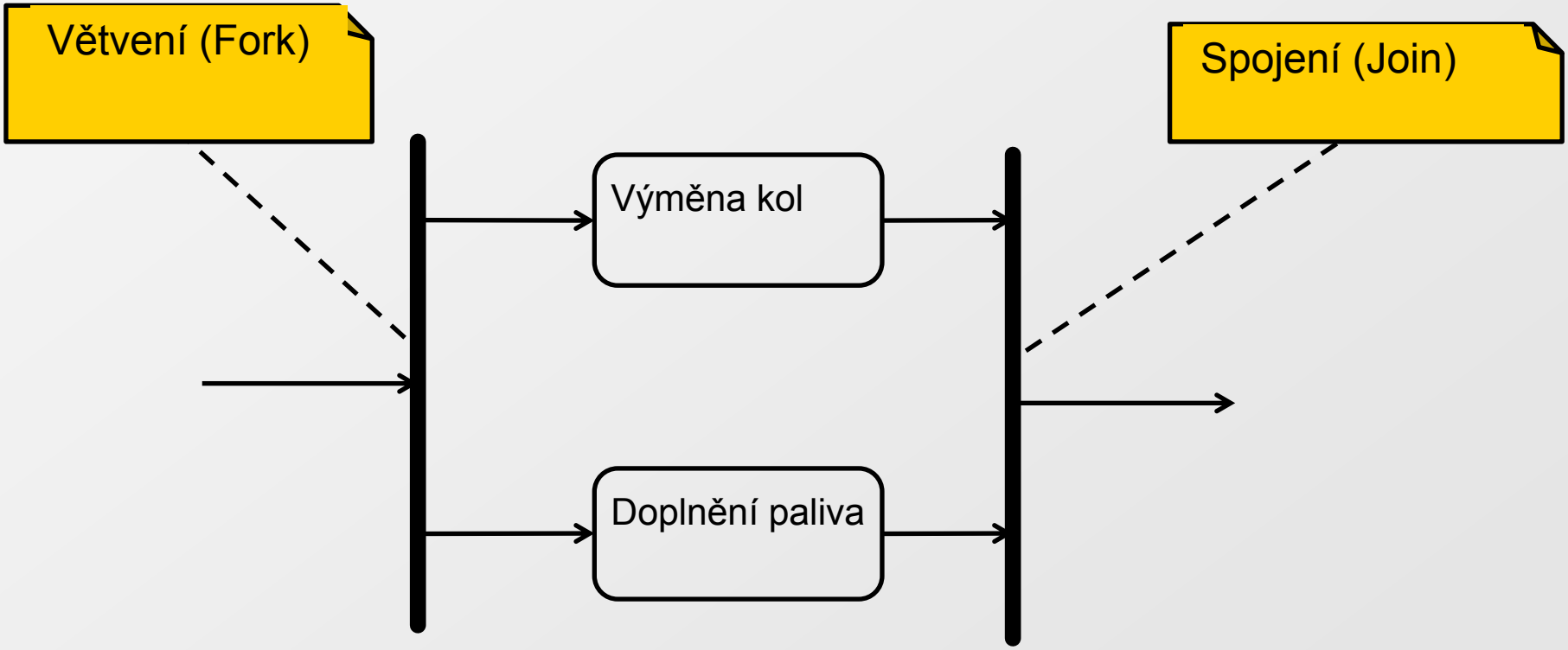
Rozhodnutí (Decision) a sloučení (Merge)



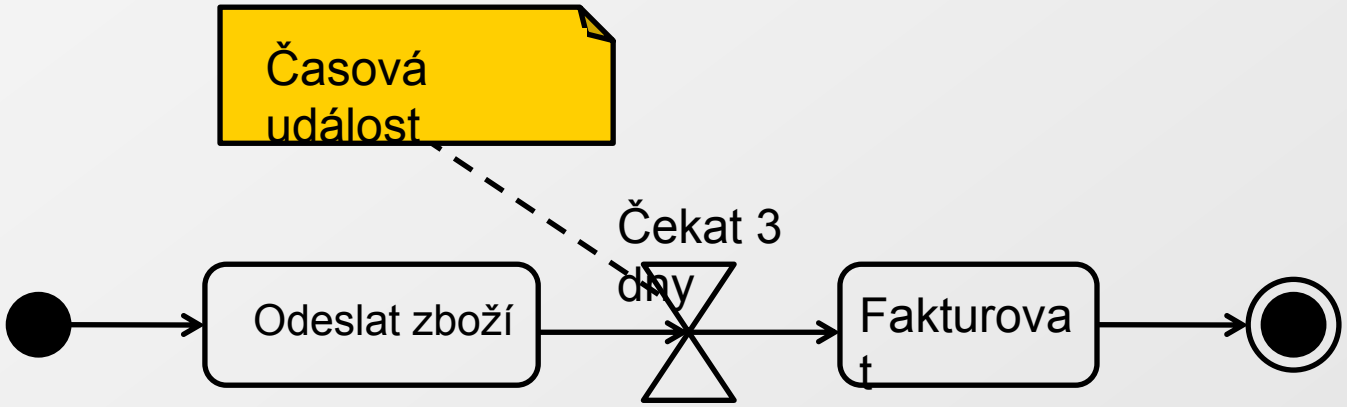
Rozhodnutí a sloučení



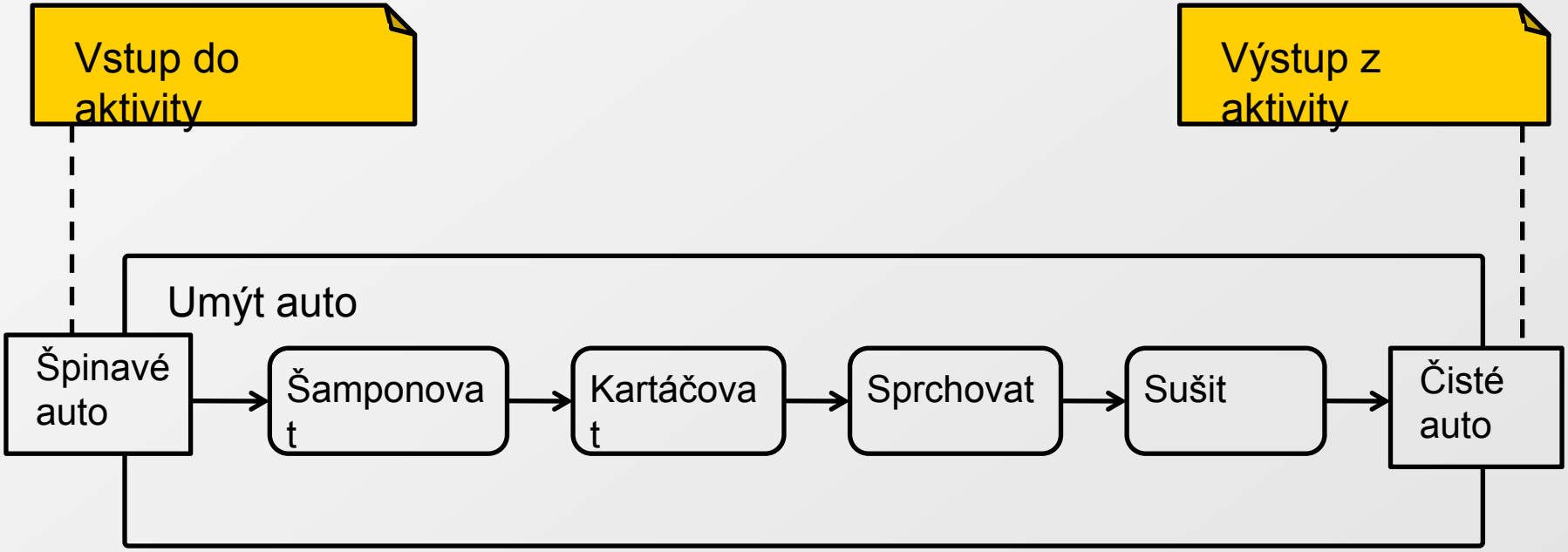
Paralelní akce
Větvení (Fork) a spojení (Join)



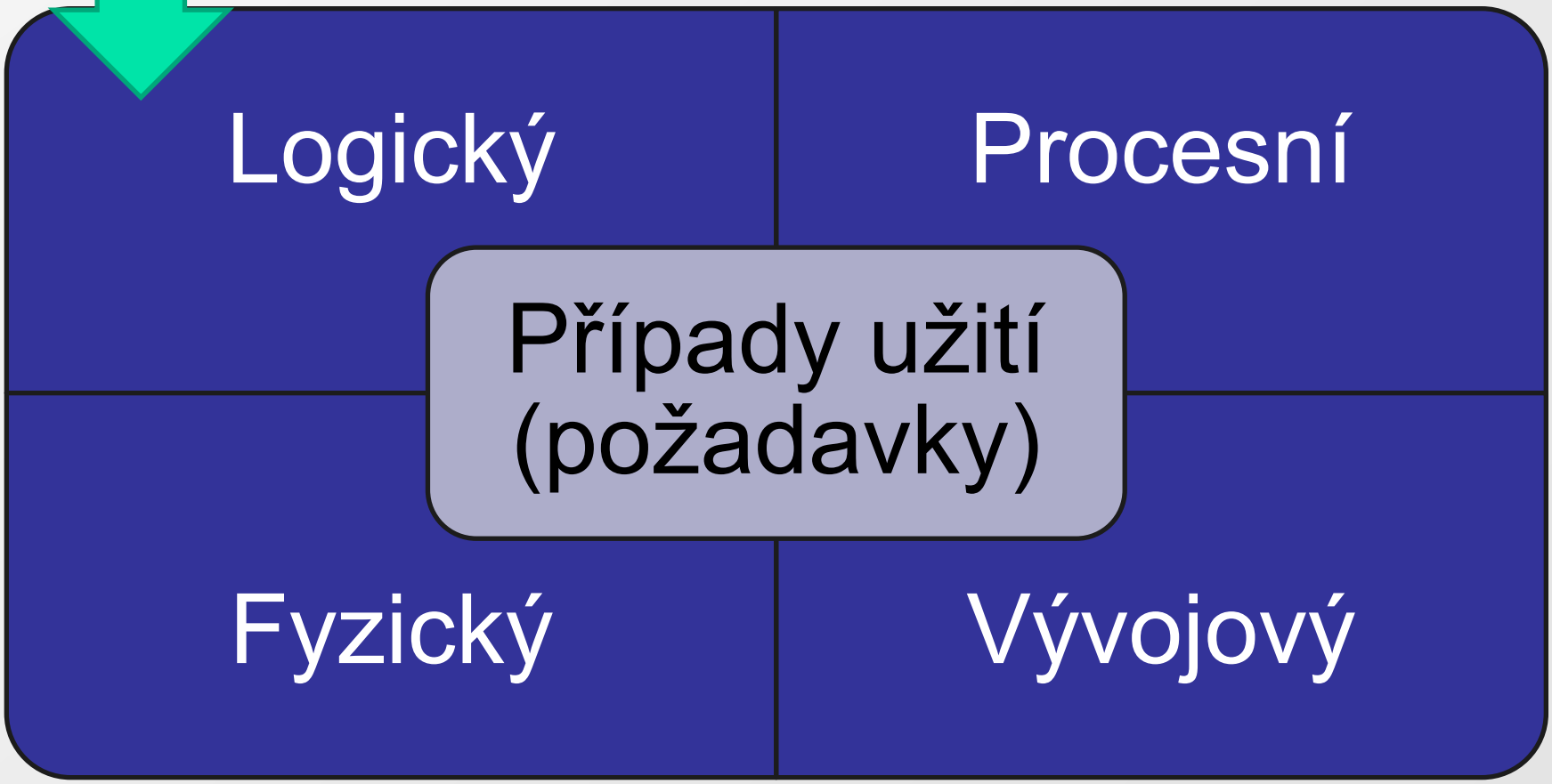
Časové události



Vstup do a výstup z aktivity



Pohledy na model IS



Modelování logické struktury

- Případy užití popisují:
  - chování systému jako množinu aktivit
- Třídy (Classes) popisují:
  - typy objektů, které jsou potřebné k tomu, aby systém byl schopen vykonávat tyto aktivity

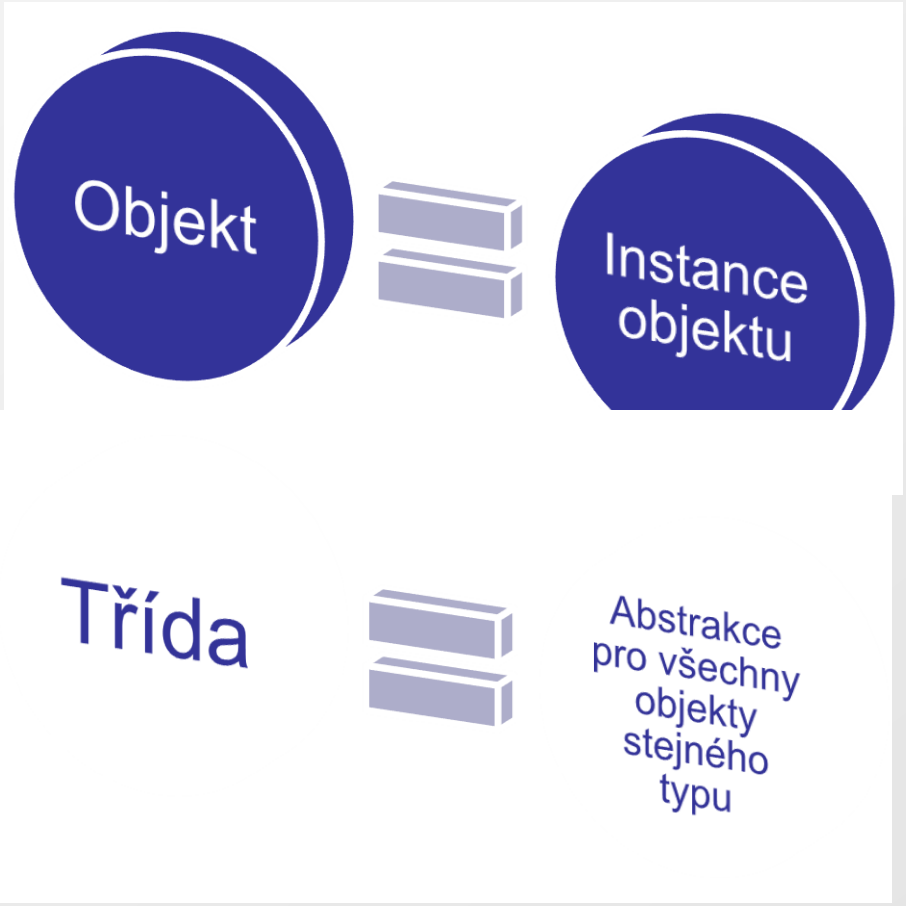
Třídy a objekty

- Objekt:
  - kombinuje data a funkce do jediné soudržné jednotky
  - ukrývá svoje data za vrstvu funkcí
- Objekt má:
  - Stav:
    - hodnoty atributů
  - Chování:
    - operace, které popisují chování objektu
    - implementace (realizace) operace se nazývá **metoda**
    - metody zpravidla mění stav (atributy) objektu
  - Identitu objektu

Třídy a objekty

- Třídy
  - Zapouzdření (Encapsulation)
    - Objekt jako černá skříňka, která s okolím komunikuje prostřednictvím metod
    - Snadno lze změnit chování třídy a nemá to vliv na okolí

Třídy a objekty

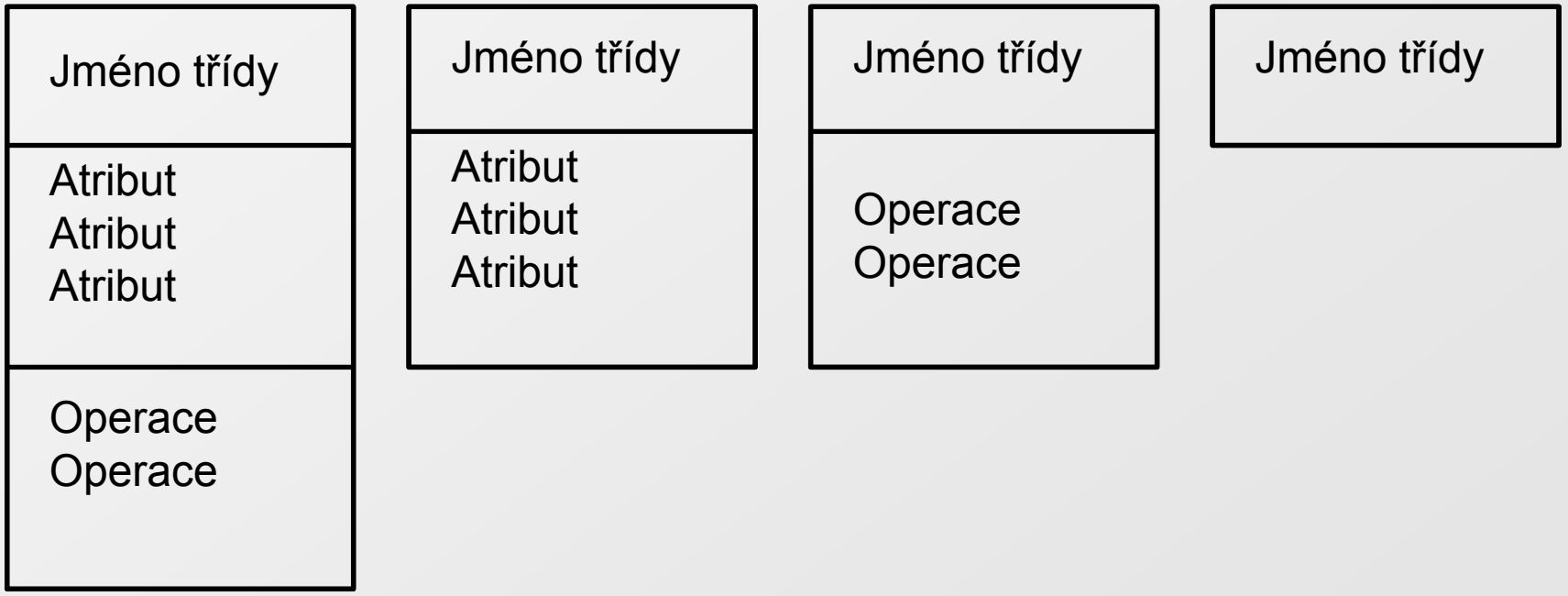


Třídy a objekty

- Objekt:
  - Osobní automobil
    - typ: Škoda Octavia
    - barva: červená
    - RZ: 4J0 3695
- Třída
  - Osobní automobil
    - Popisuje atributy (typ, barva,RZ) a chování všech osobních automobilů

Diagramy tříd (Classes)

- Možnosti zobrazení třídy v UML

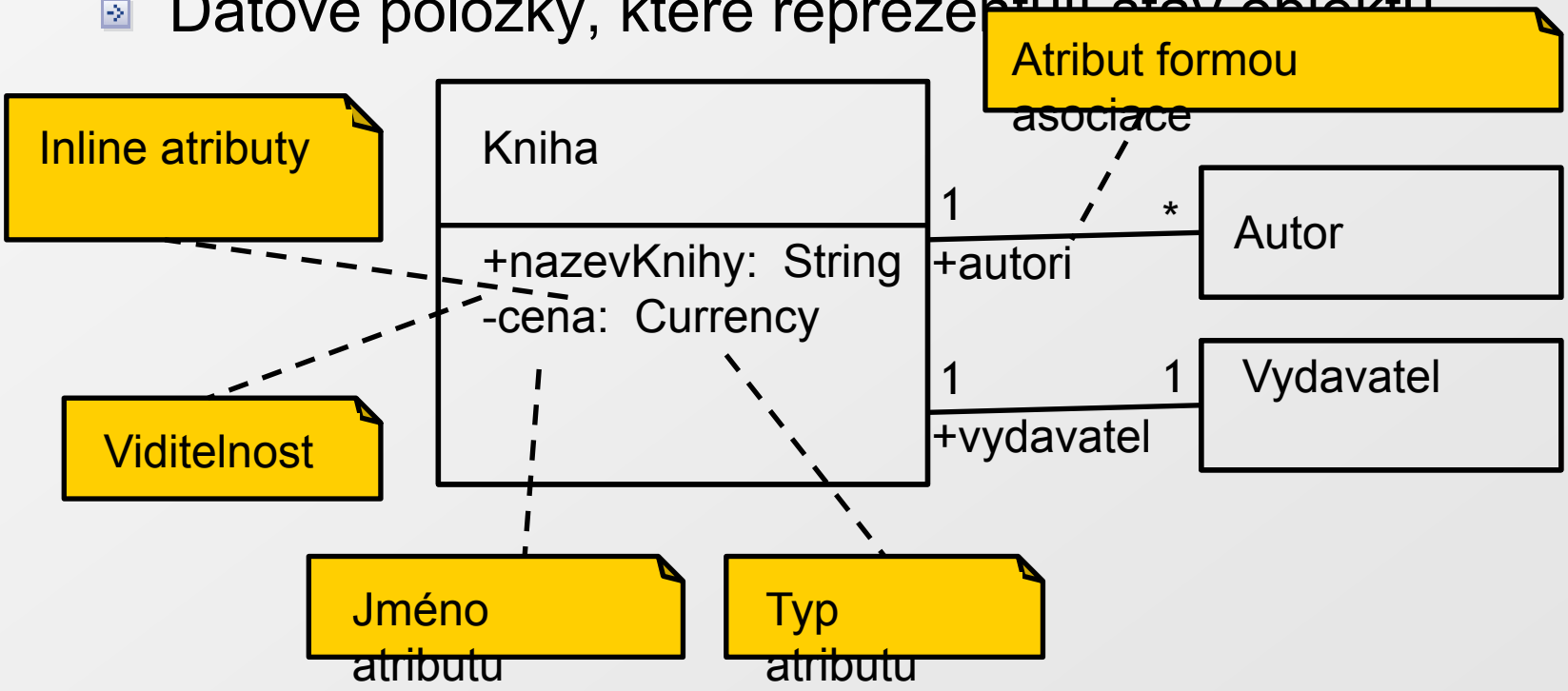


Viditelnost atributů a operací

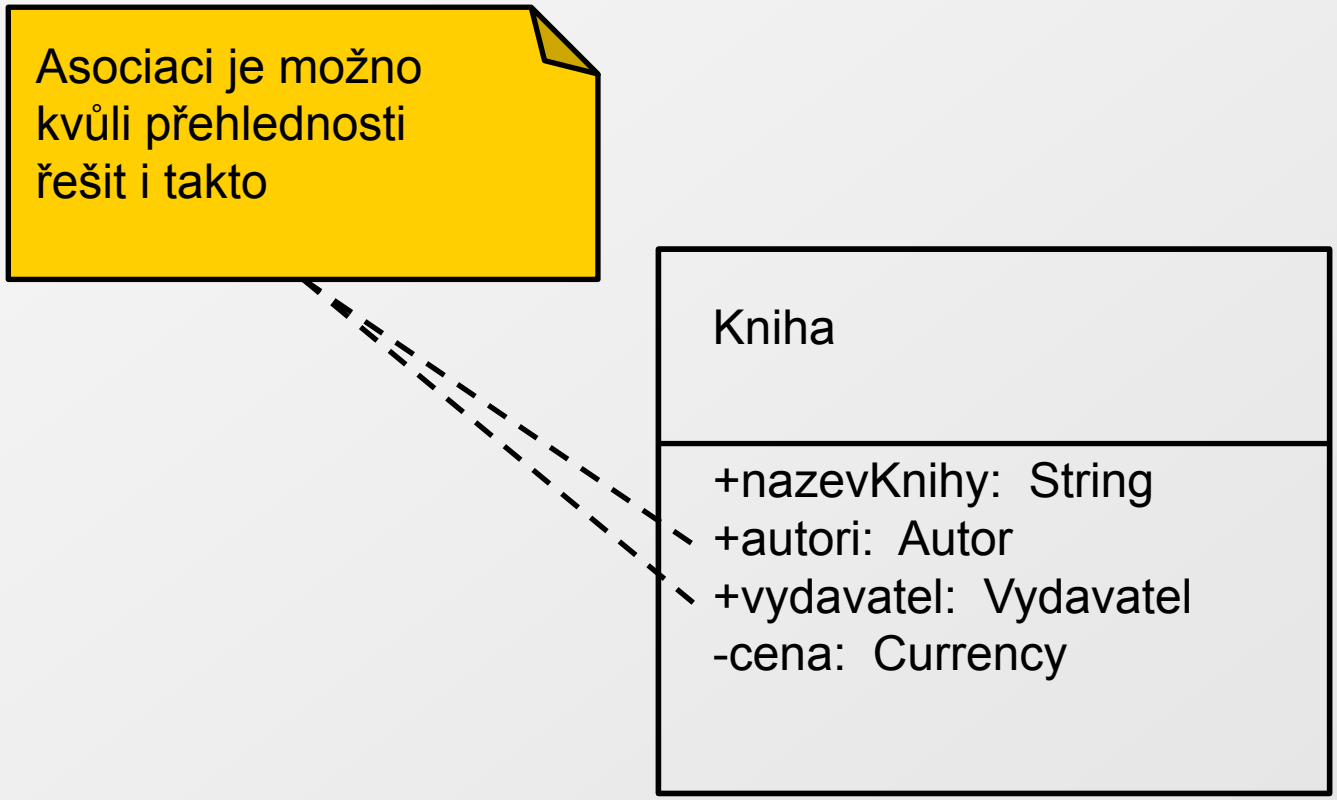
Znak před atributem nebo operací	Viditelnost	Viditelnost anglicky	Atribut nebo operace jsou viditelné pro:
+	veřejný	Public	všechny třídy
#	chráněný	Protected	třídu, kde je deklarován, a třídy, které dědí z této třídy
~	balíčková	Package	všechny třídy z balíčku
-	privátní	Private	jen třída, kde je deklarován

Atributy třídy

- Datové položky, které reprezentují stav objektu



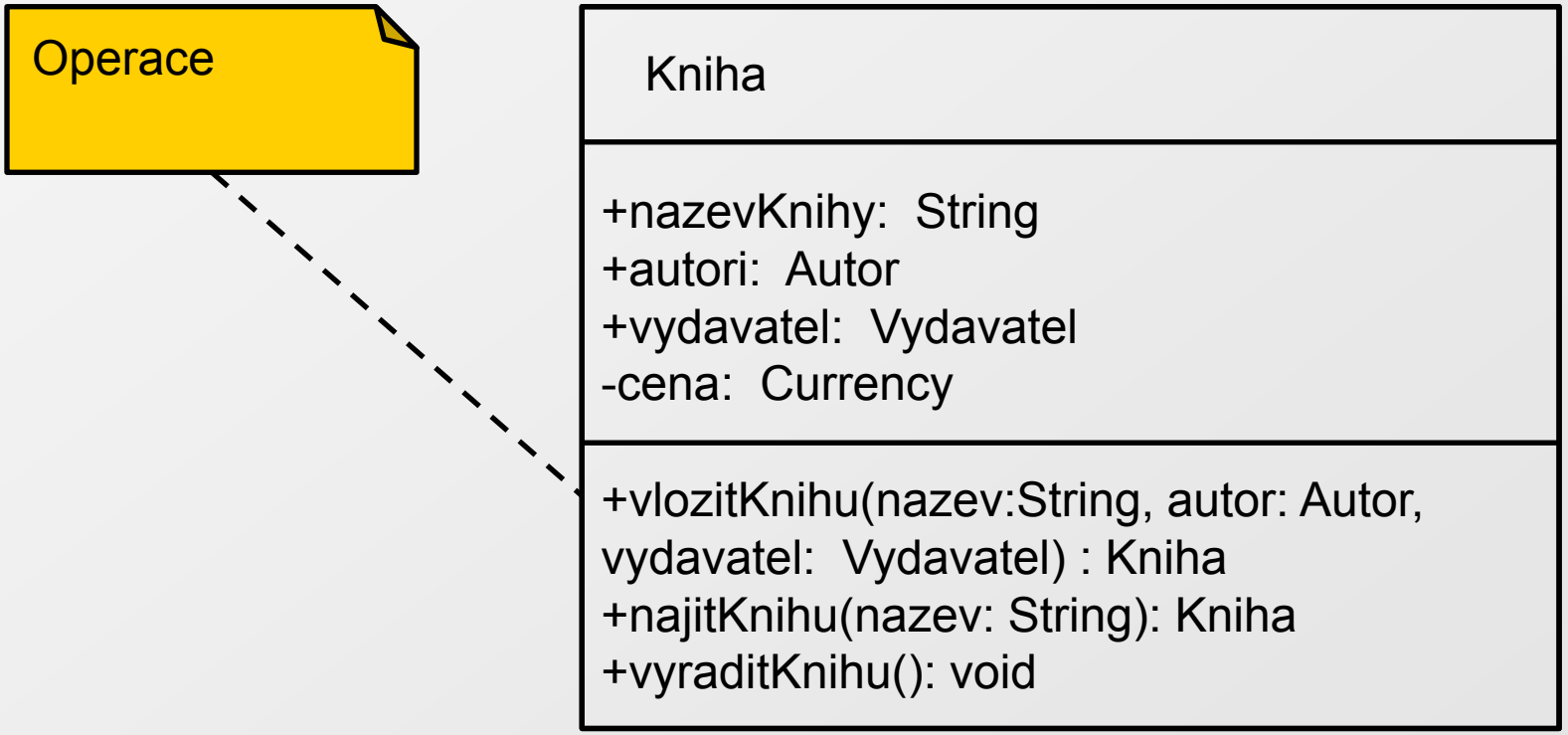
Atributy třídy: asociace tříd



Atributy třídy

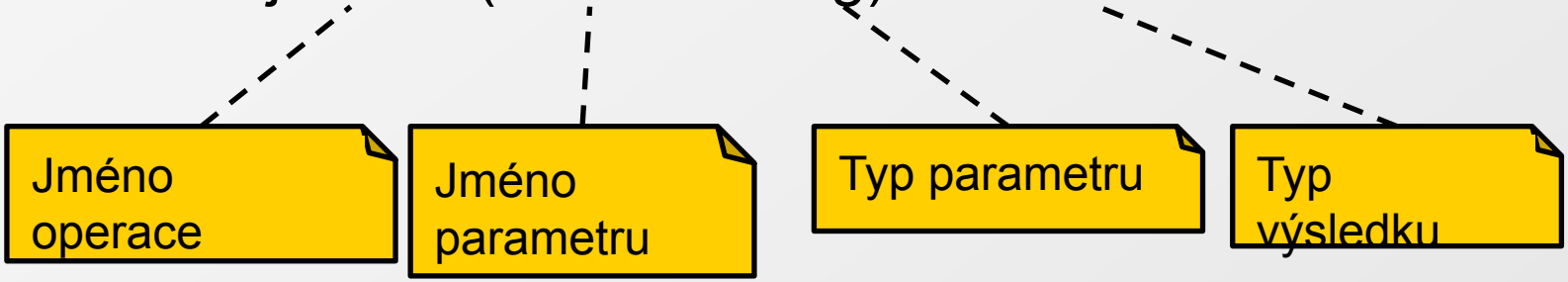
- Jména:
  - neobsahují české znaky
- Jména tříd:
  - Např. začíná velkým písmenem
- Jméno atributu:
  - Např. začíná malým písmenem, další slova začínají velkým písmenem
- Typ atributu:
  - Např. String, Integer, Float, Boolean, Char, Currency (měna) – inspirace z konkrétního jazyka, který se bude používat (Java, C++, Objective C)

Chování třídy: operace



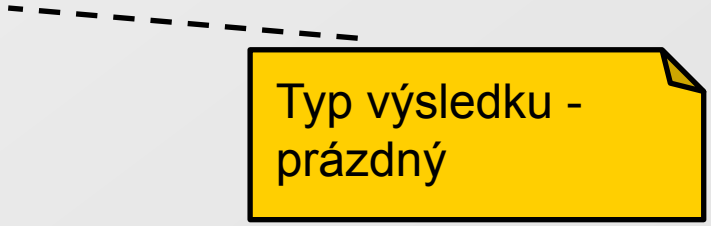
Parametry a výsledek operace

- +najitKnihu(nazev: String): Kniha



- +vlozitKnihu(nazev:String, autor: Autor, vydavatel: Vydavatel) : Kniha

- +vyraditKnihu(): void

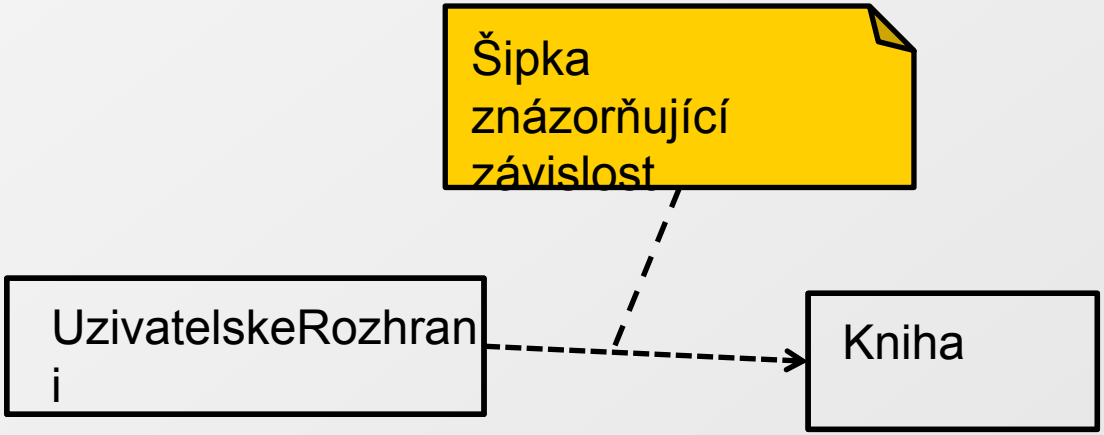


Vztahy mezi třídami

- Závislost (Dependency)
- Asociace (Association)
- Agregace (Aggregation)
- Kompozice (Composition)
- Generalizace/Dědičnost (Generalization/Inheritance)

Závislost

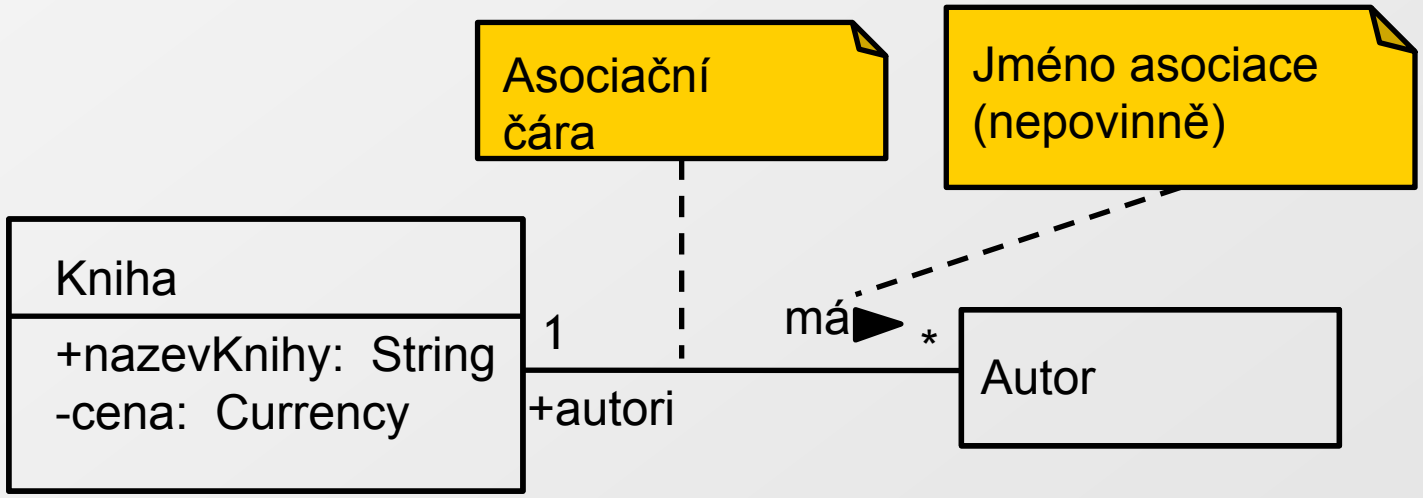
- Třída potřebuje jinou třídu pro svoji činnost
- Nejslabší vazba mezi třídami



- Aby bylo možno zobrazit objekt Kniha prostřednictvím uživatelského rozhraní, musí toto rozhraní rozumět třídě Kniha

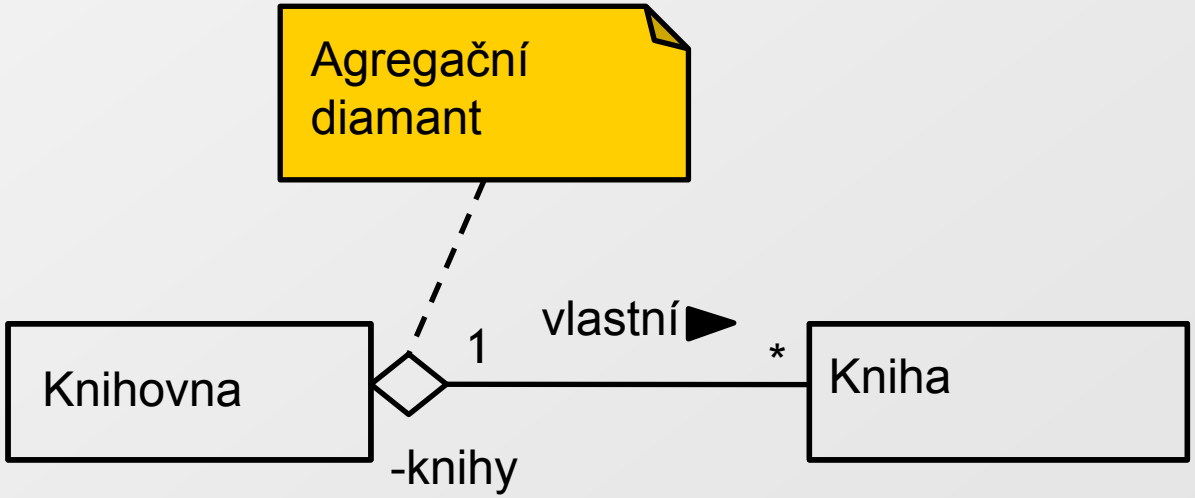
Asociace

- Třída obsahuje odkaz na objekt jiné (asociované) třídy



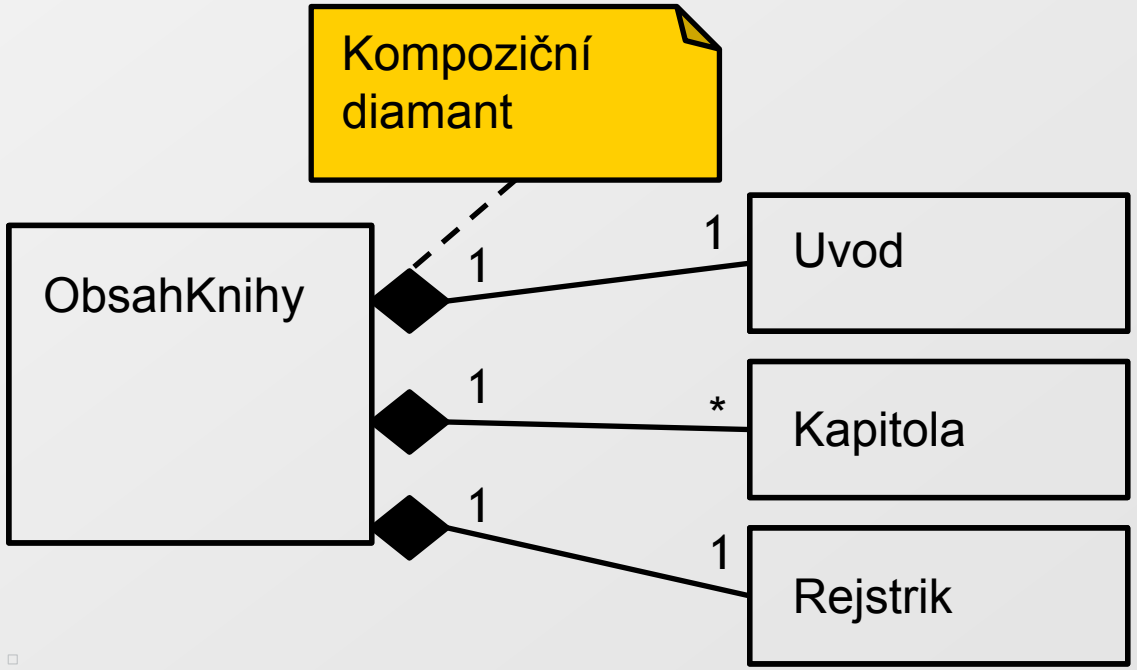
Agregace

- Silnější vazba než asociace
- Třída vlastní a může sdílet objekty jiné třídy



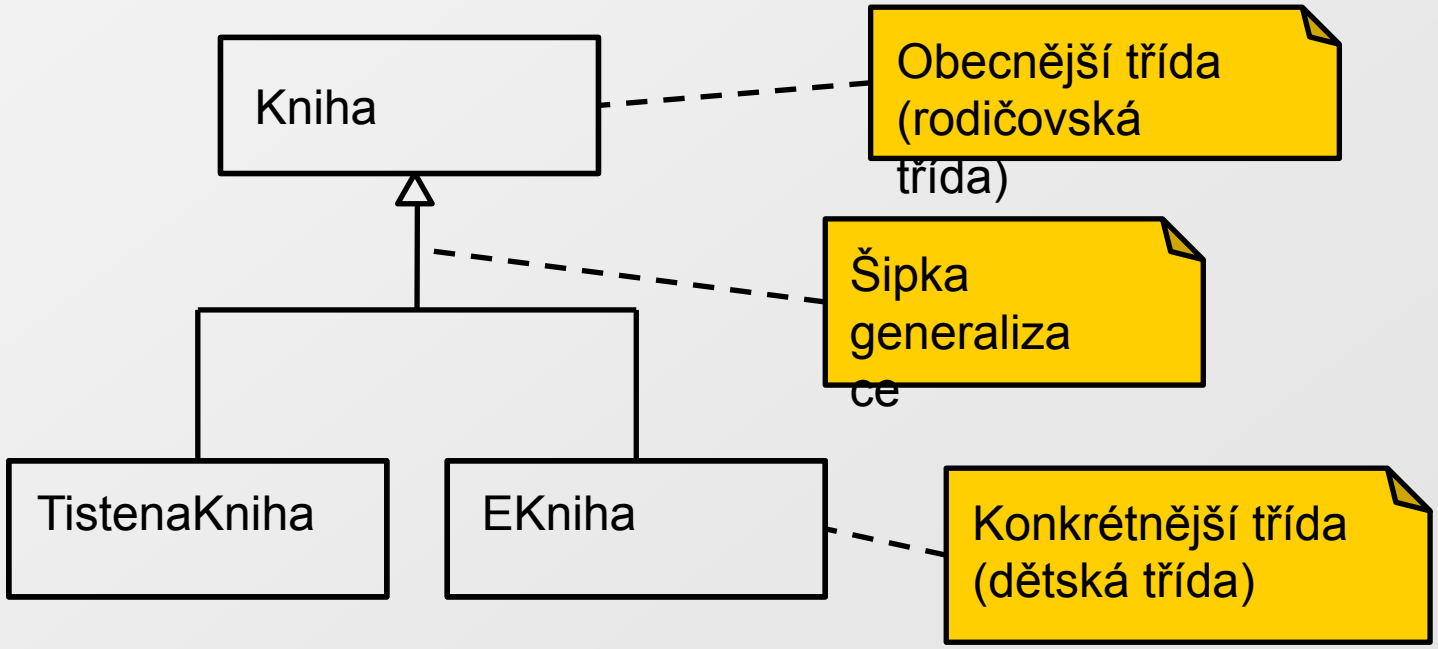
Kompozice

- Silnější vazba než agregace
- Objekt třídy se skládá z objektů tříd, které tvoří kompozici a nemají samostatně smysl



Generalizace (Dědičnost)

- Vztah vyjadřuje, že třída je typem jiné třídy



Generalizace a dědění atributů a metod

- Dětská třída dědí a znovu používá všechny atributy a metody rodičovské třídy, které jsou pro ni viditelné
  - viditelnost **Public**, **Protected** a **Package** , pokud je definována ve stejném balíčku
- Dětská třída samozřejmě obsahuje další atributy a metody

Násobná dědičnost (Multiple inheritance)

- Dětská třída může dědit atributy a metody od více rodičovských tříd