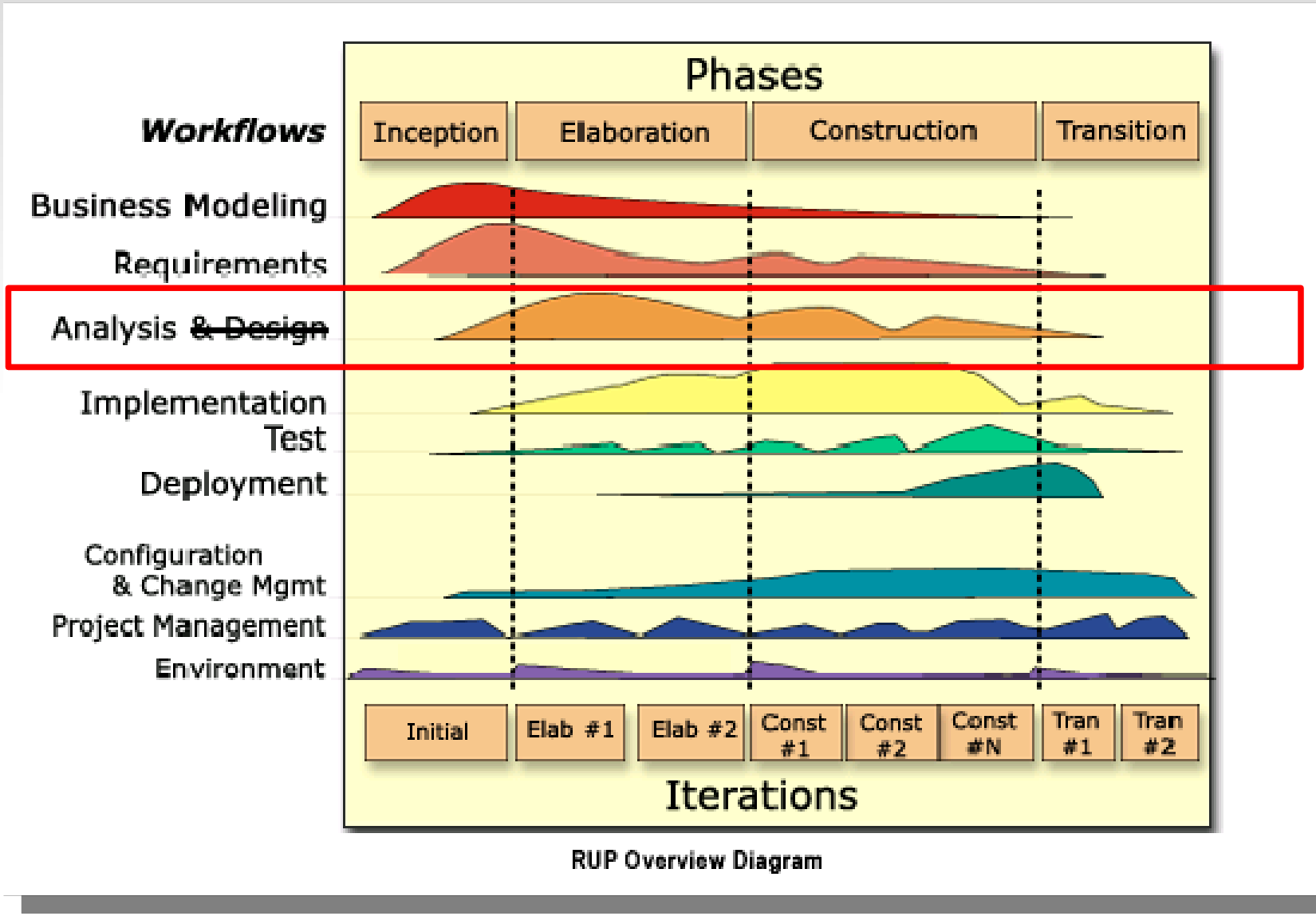
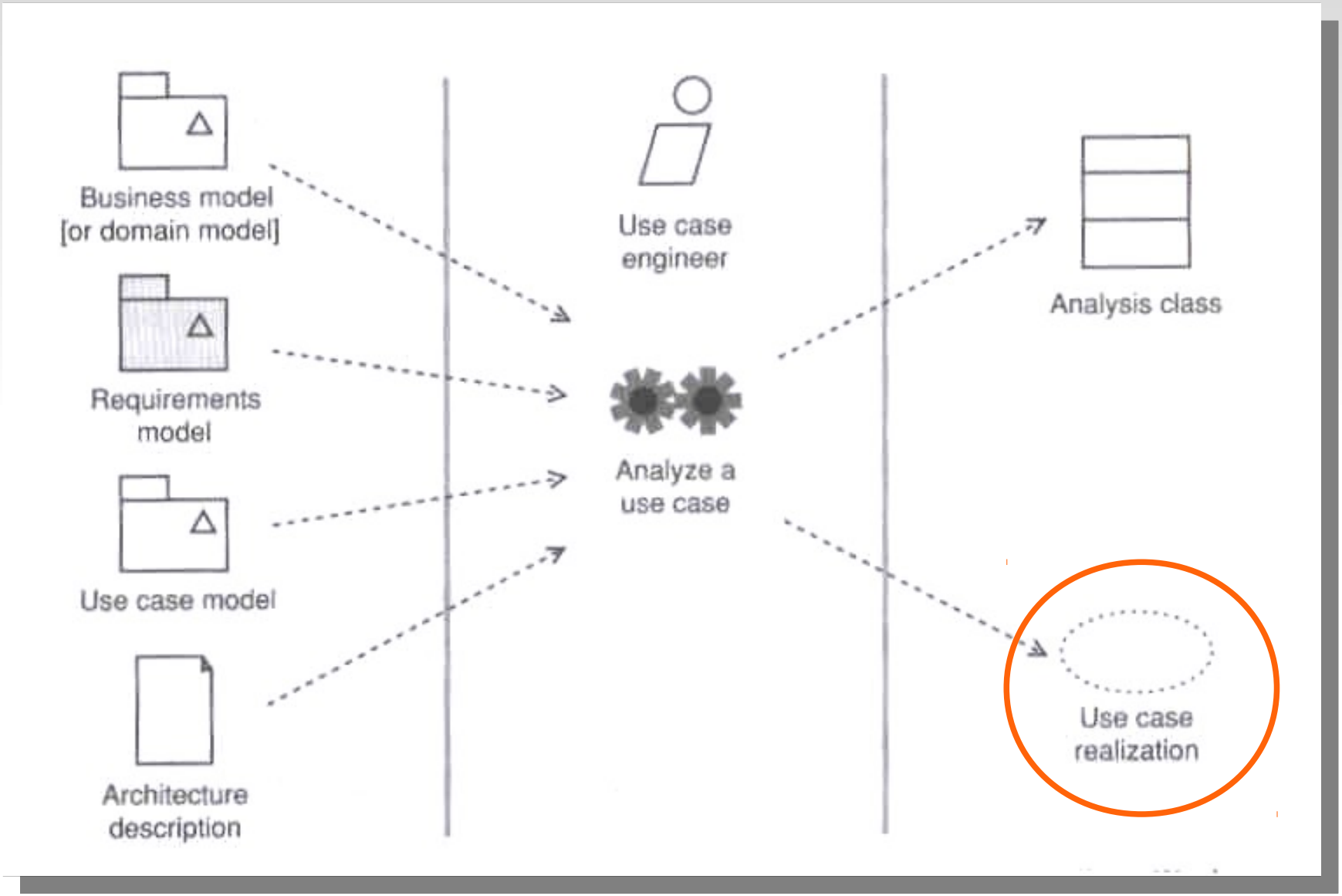
Analýza:
Realizace případů užití

© Radek Ošlejšek
Fakulta informatiky MU
oslejsek@fi.muni.cz

Analysis workflow

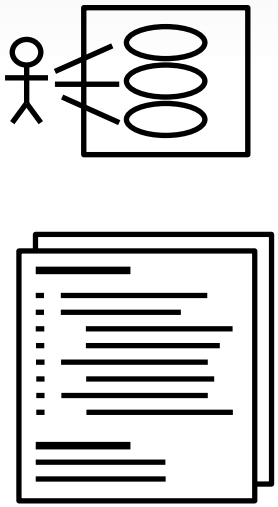


UP aktivita: Analyse a use case

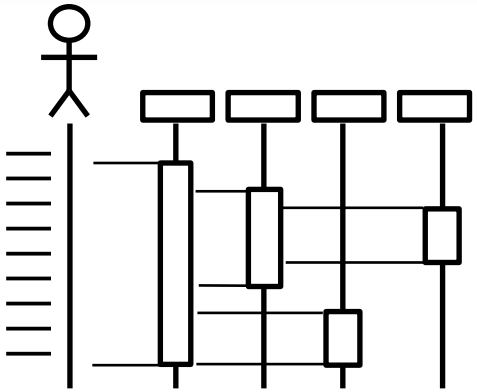


Realizace případů užití

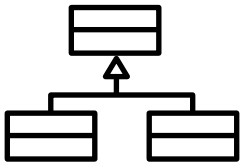
- Analytický diagram tříd „vypráví příběh“ o jednom nebo více případech užití
- Interakční digramy ukazují, jak instance *klasifikátorů* uvnitř systému interagují s cílem realizovat požadované chování systému



UC model
(požadavky na systém)



Realizace případů užití
(interakční diagramy)



Analytický model
(model doménové oblasti)

Cíle realizace UC v analytické fázi

- Odhalení interakcí mezi analytickými třídami, které vedou k realizaci jednotlivých případů užití
  - mohou se nalézt nové analytické třídy
- Zjištění konkrétních zpráv, které se musí předávat pro realizaci specifického chování
  - nalezení klíčových operací, které musí analytická třída mít
  - nalezení klíčových atributů
  - nalezení důležitých vztahů mezi analytickými třídami
- Dosažení konzistence mezi UC modelem a analytickým modelem tříd
- Není nutné dělat realizaci případů užití pro všechny p.u., stačí vybrat ty nejdůležitější
- Diagramy interakcí modelujte tak, aby byly co nejjednodušší

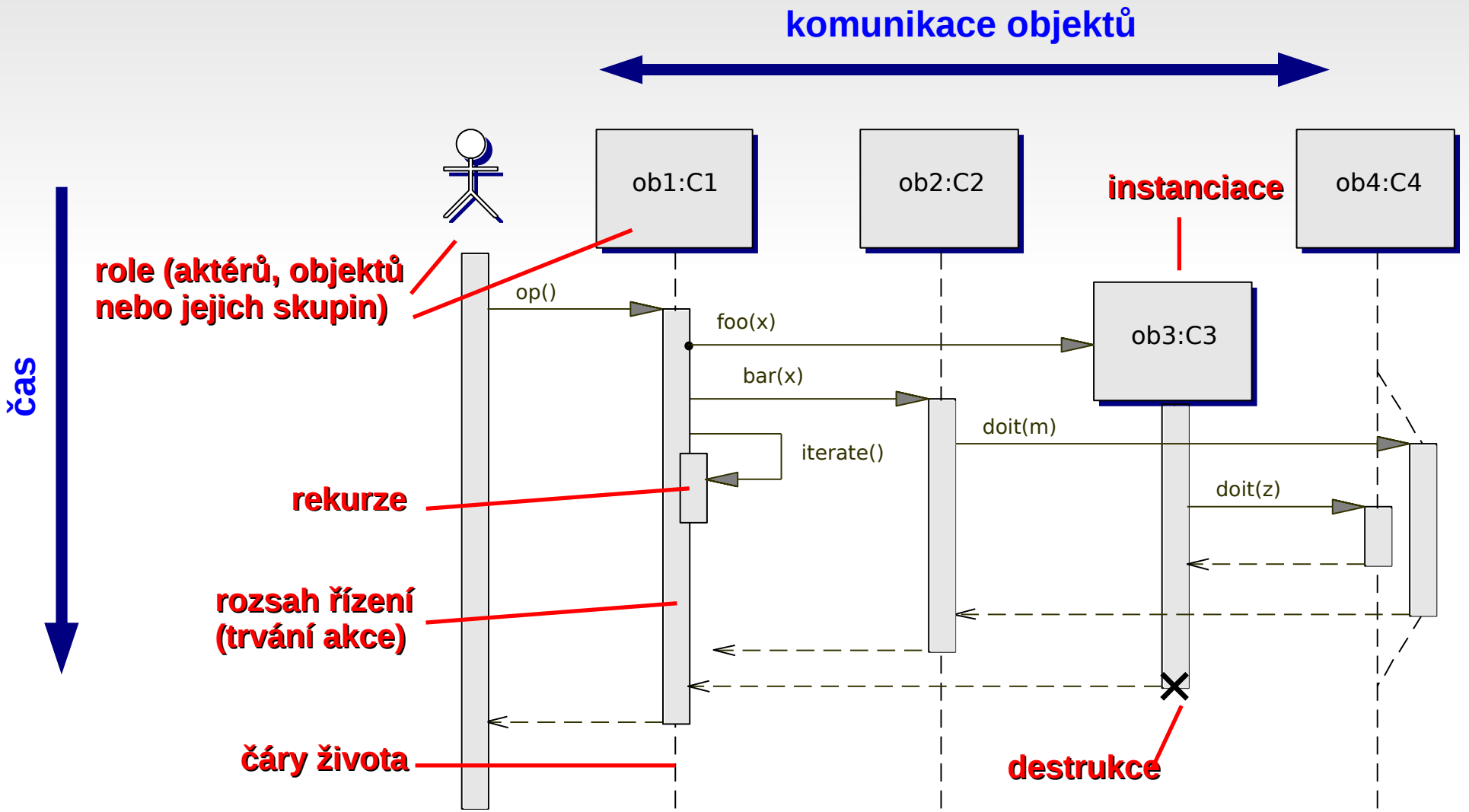
Diagramy interakcí

- Diagramy interakcí modelují chování systému v průběhu časového úseku.
- 4 různé diagramy nabízejí 4 různé pohledy na interakci objektů
- **Sekvenční diagramy** (sequence d.)
  - modelují výměnu zpráv s důrazem na časovou osu
- **Komunikační diagramy** (communication d.)
  - modelují interakce organizované podle interagujících objektů
- **Diagramy přehledů interakcí** (interaction overview d.)
  - ukazují realizaci složitého chování pomocí několika jednodušších interakcí (kombinuje sekvenční diagramy a digramy aktivit)
- **Časovací diagramy** (timing d.)
  - zaměřují se na “real-timové” aspekty interakcí, časová omezení a závislosti, ...

Sekvenční diagram
(Sequence Diagram, SD)

UML notace

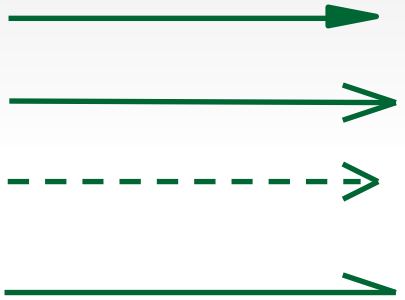
Sekvenční diagram reprezentuje **interakci**, kterou představuje množina **zpráv** vyměněných mezi objekty **během spolupráce** pro splnění požadované operace nebo získání výsledku.



Zprávy

Zpráva nese **informaci** a spouští **akci** v jiném (nebo stejném) objektu, je reprezentována pomocí **šipky volání**.

- **jméno zprávy**: signatura volané metody (parametry a návratové hodnoty mohou být vynechány) nebo jméno zaslaného signálu/události
- **synchronní volání** (vyvolání procedury):
- **asynchronní volání**
- **návrat** z volané procedury
  (může být vynechán v procedurálním toku řízení):
- **asynchronní zpráva** (signál):



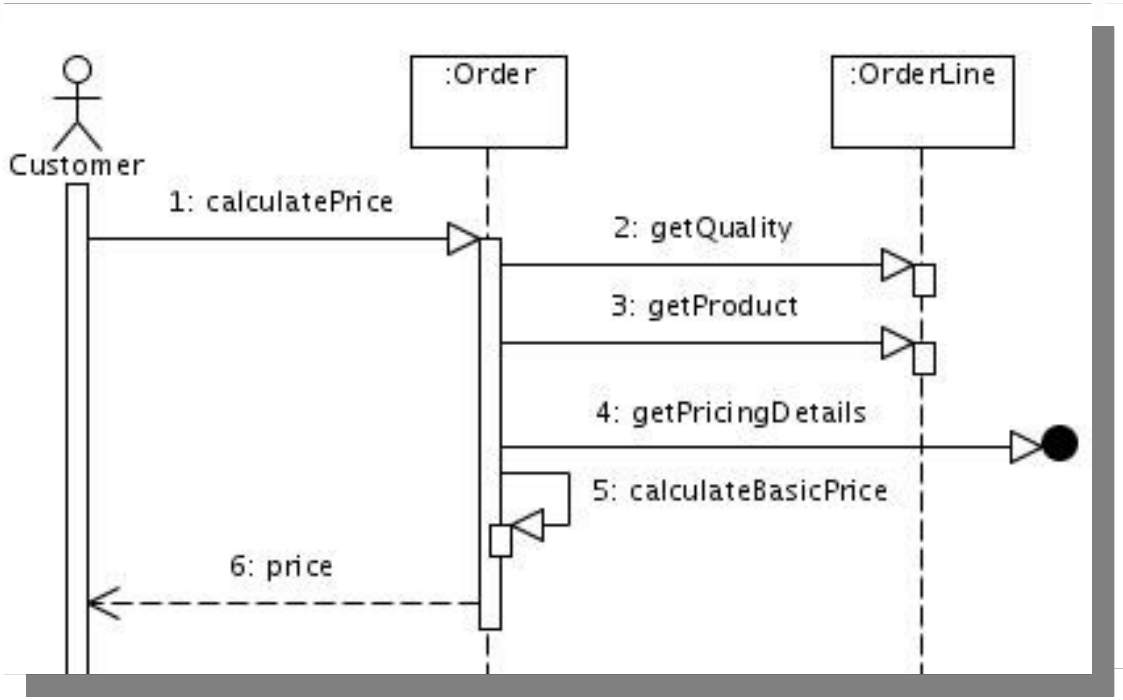
Další symboly: viz knihy o UML

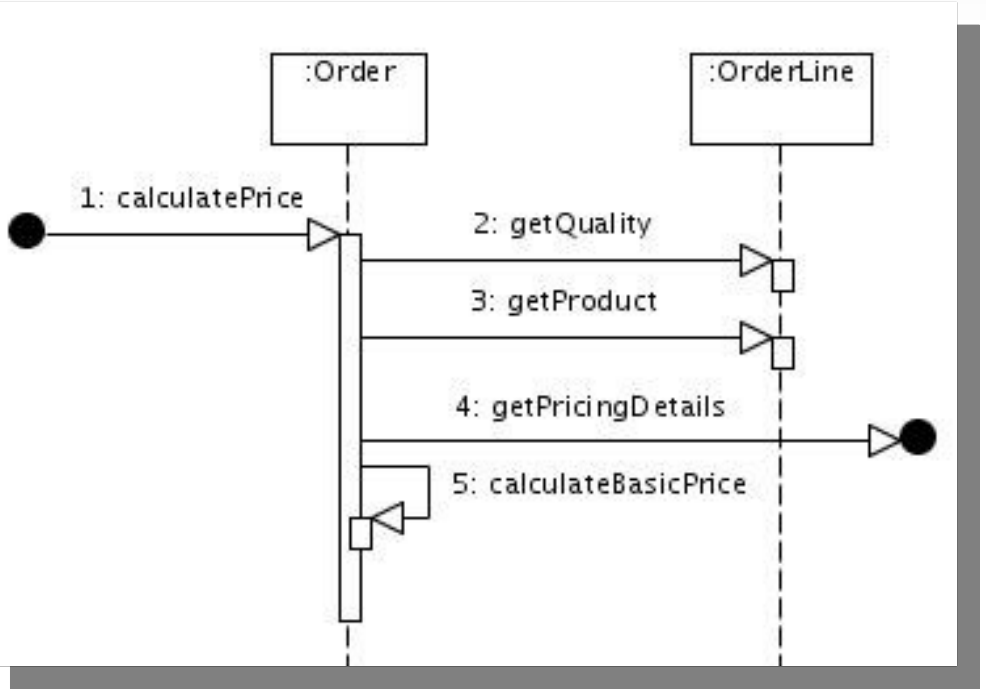
Účastníci interakce

Zahájení interakce

- Zahájení interakce
  - aktérem z diagramu případů užití
  - „nalezenou zprávou“ *(found message)* – nevíme (neřešíme), kdo zprávu poslal
- Ztracená zpráva (lost message)

  např. ztráta zprávy při chybě

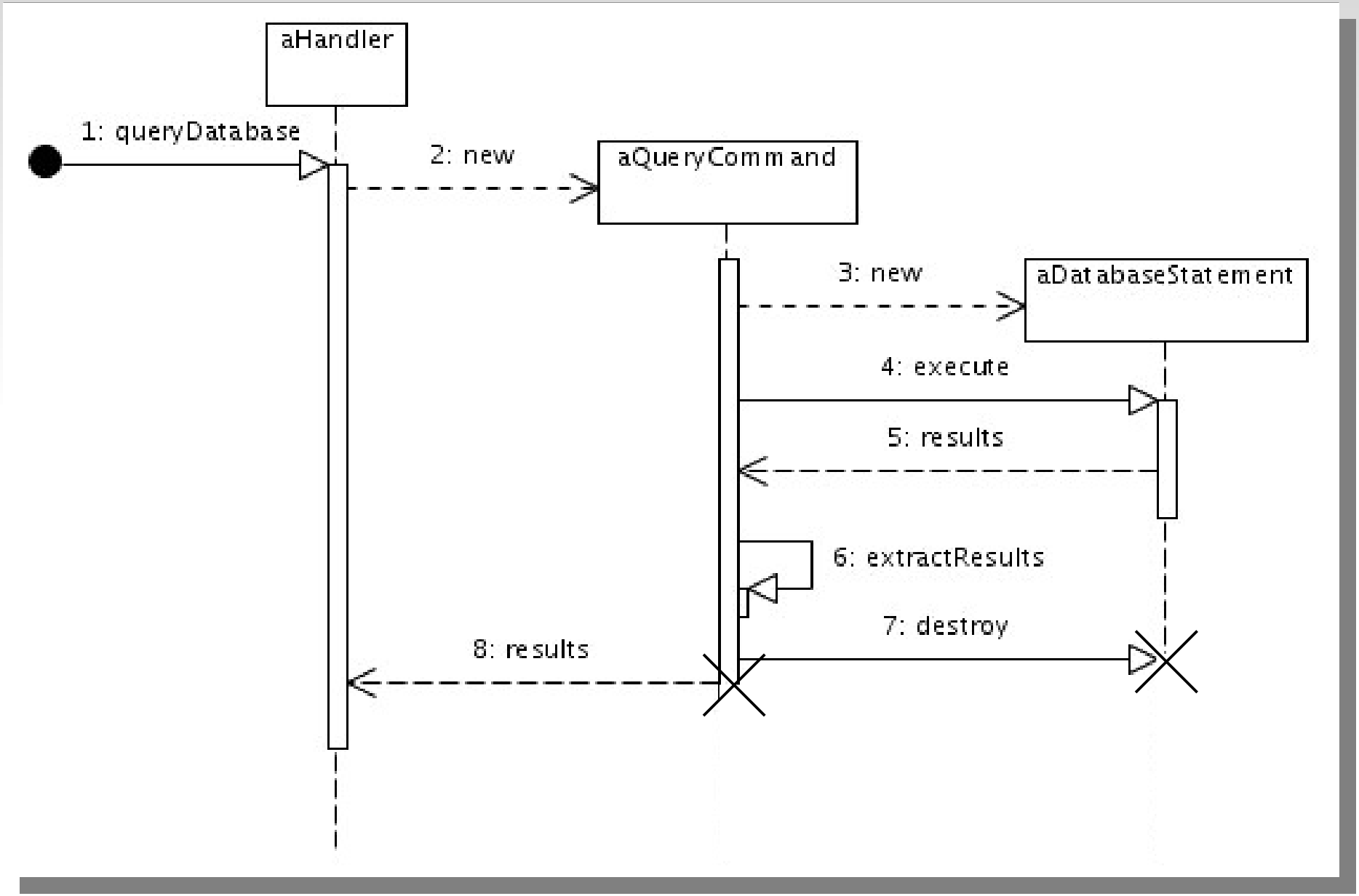




Vytváření a rušení účastníků

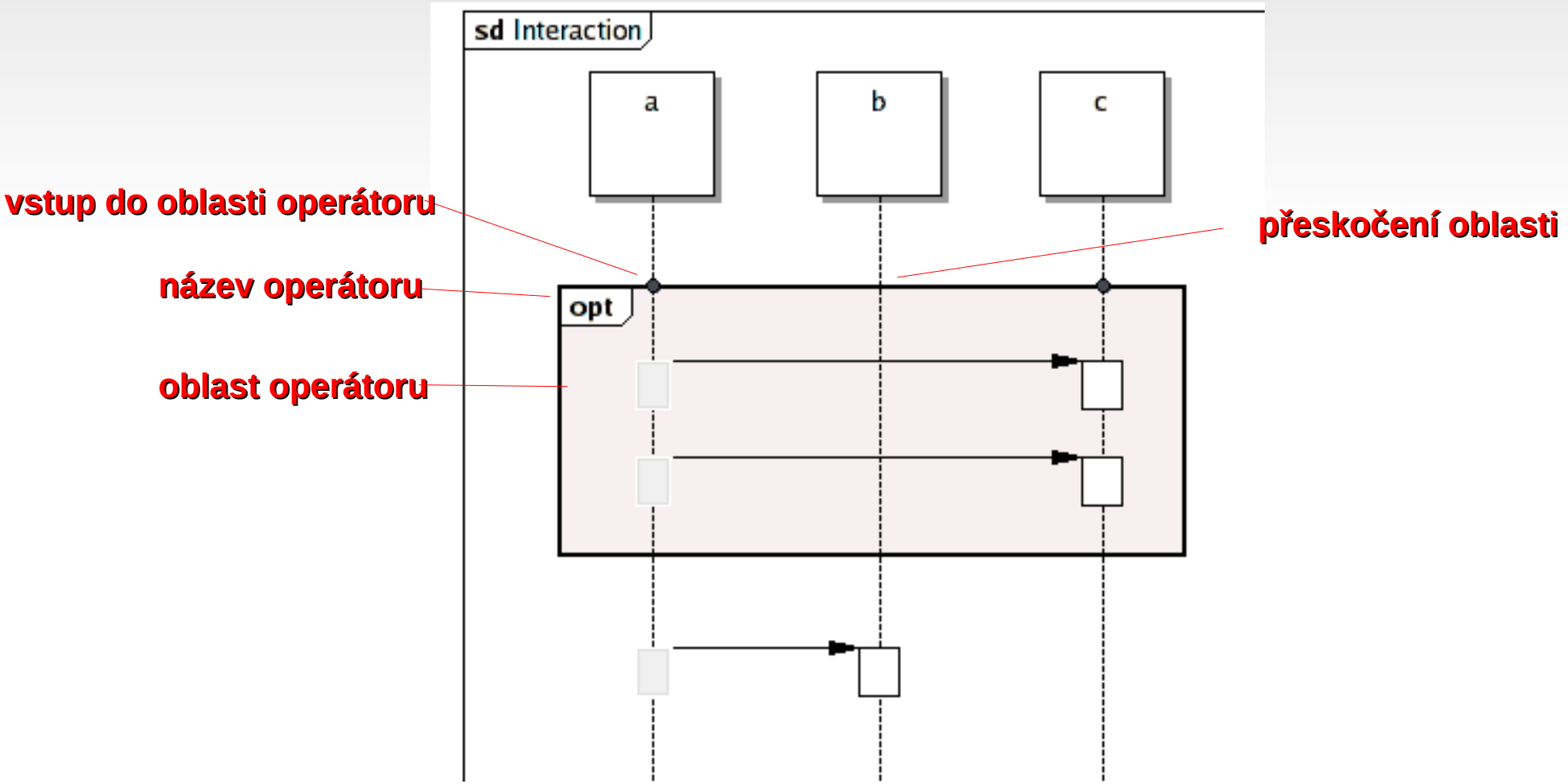
- Účastníci interakce většinou „žijí“ v průběhu celé interakce
  - nestaráme se o jejich vytvoření/rušení
  - jednoduše předpokládáme, že existují
- Explicitní vytvoření účastníka během komunikace
  - => instanciace objektu
  - komunikační šipka vede přímo do objektu, ne do jeho čáry života
- Explicitní zrušení
  - => destrukce objektu
  - křížek na konci čáry života
  - samozničení vs. zničení jiným objektem

Vytváření a rušení účastníků (II)



Strukturní dělení

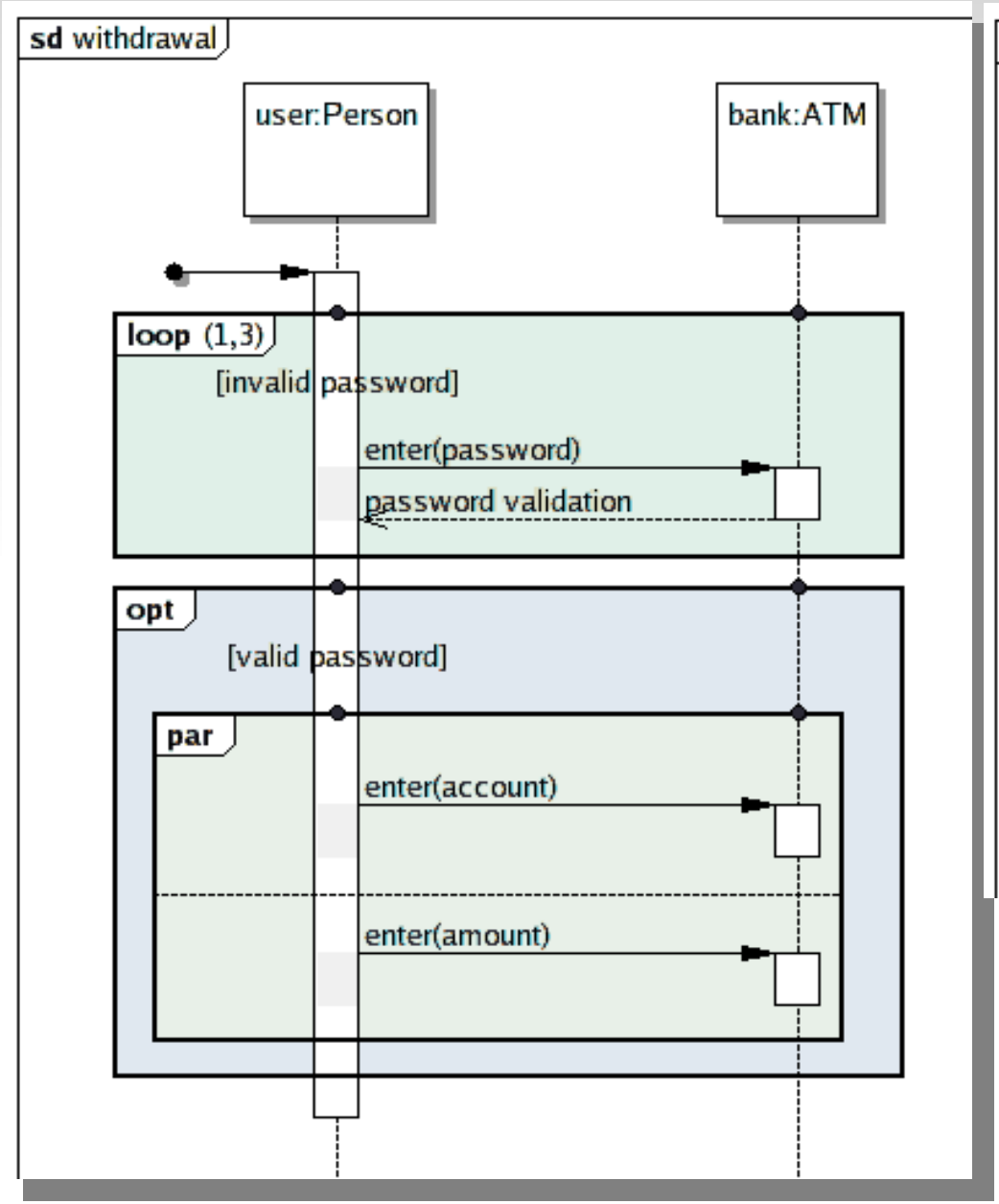
- Znázornění souběžných sekvencí, cyklických sekvencí apod.
  - vyznačíme oblasti a pojmenujeme operátor pro danou oblast
  - operátor se aplikuje na všechny čáry života, které do něj vstupují

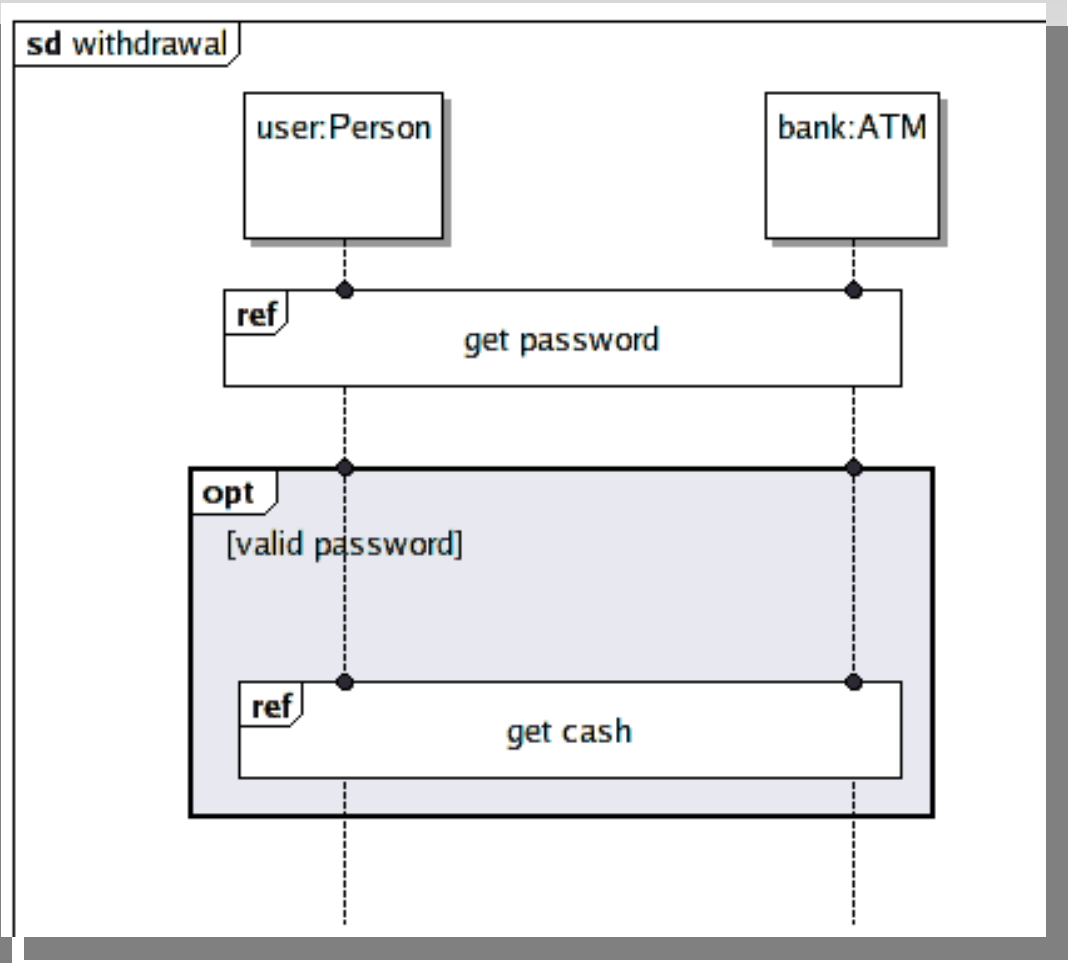


Nejčastější operátory interakcí

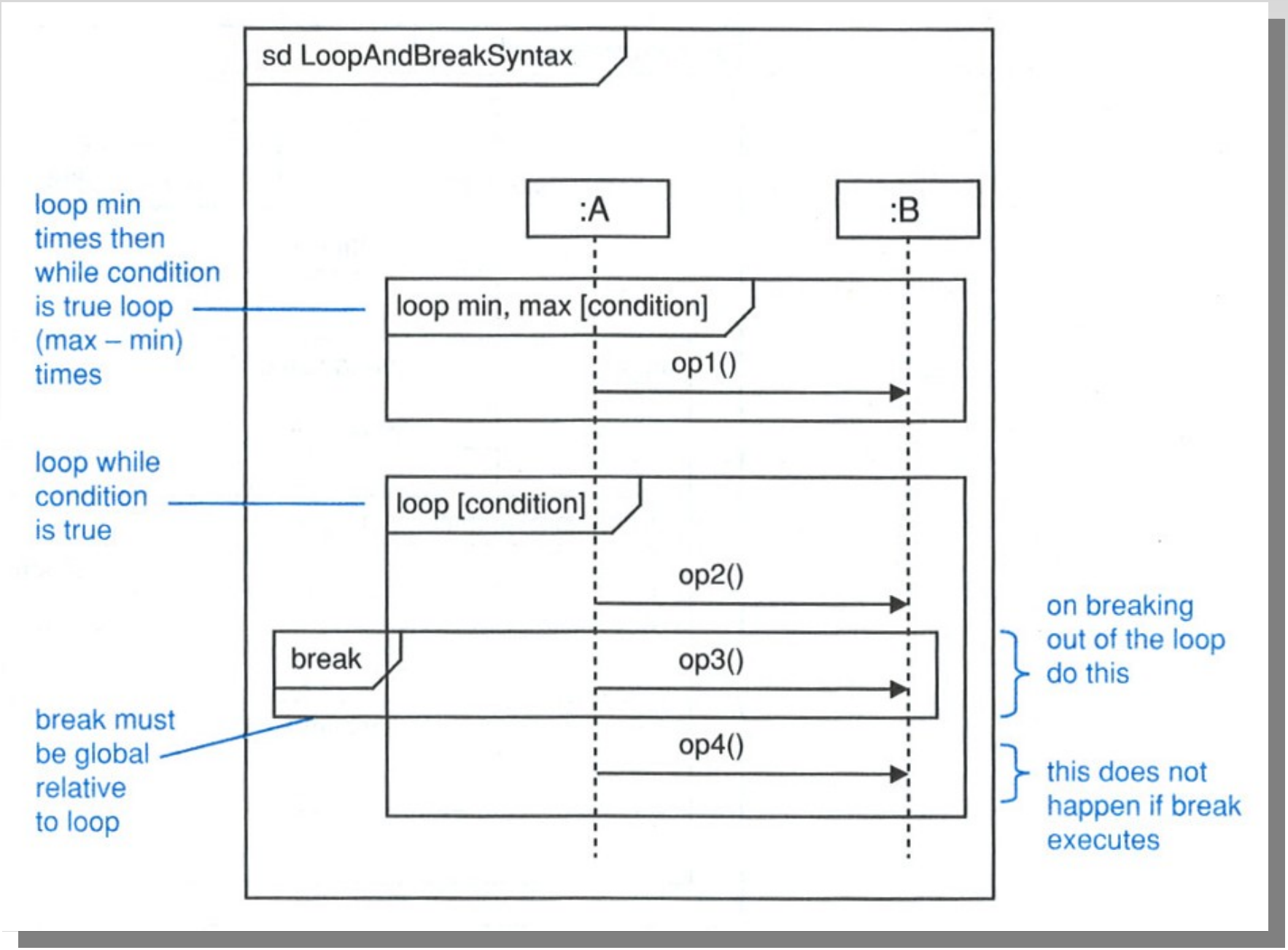
- **podmíněné spuštění** *(optional execution),* **opt**
  - operace uvnitř bloku jsou spuštěny jen pokud je splněna podmínka stráže (není to větvení!)
- **volitelné spuštění** *(conditional execution),* **alt**
  - blok je rozdělen na více horizontálních podoblastí reprezentujících jednotlivé větve podmínky; každá podoblast má vlastní stráž (= větvení)
- **paralelní spuštění** *(parallel execution),* **par**
  - blok je rozdělen na více horizontálních podoblastí; každá podoblast reprezentuje paralelní spuštění
- **cyklické spuštění** *(loop execution),* **loop**
  - blok se neustále spouští dokola, pokud je stráž vyhodnocena jako *true*
  - čísla za **loop** mohou udávat minimální a maximální počet cyklů
- **odkaz na jiný diagram** *(reference),* **ref**
  - blok obsahuje pouze název aktivity, která je popsaná separátním diagramem (diagram aktivit, stavový diagram, další sekvenční diagram apod.)
- **negace** *(negative),* **neg**
  - fragment ukazuje chybnou interakci
- **kritická sekce** *(critical region),* **region**
  - fragment může mít v daném okamžiku spuštěné pouze jedno vlákno

Strukturní dělení - příklad





Loop a Break


sd LoopAndBreakSyntax
:A
:B
loop min times then while condition is true loop (max – min) times
loop min, max [condition]
op1()
loop while condition is true
loop [condition]
op2()
break
op3()
op4()
break must be global relative to loop
on breaking out of the loop do this
this does not happen if break executes

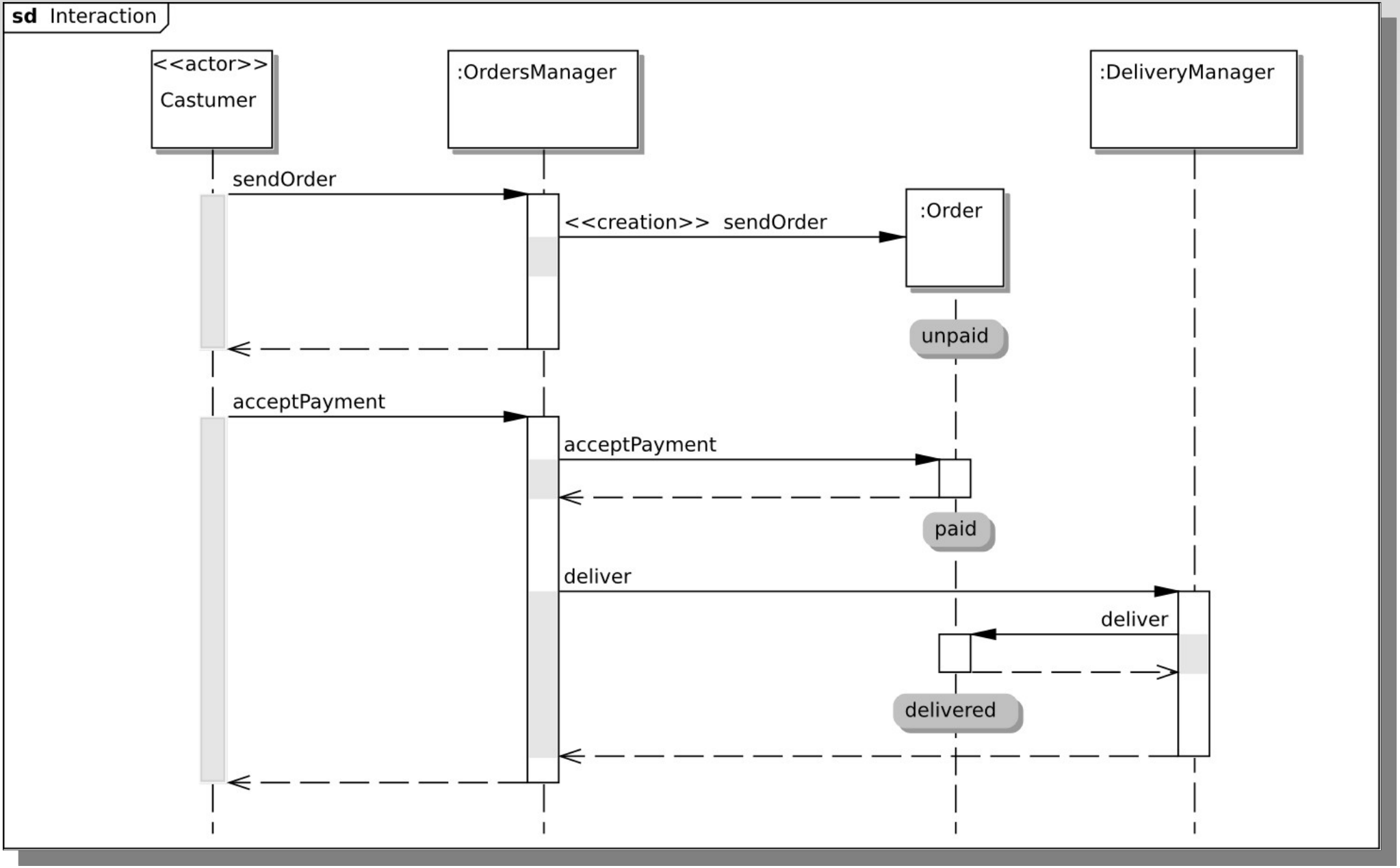
Varianty cyklů

Type of loop	Semantics	Loop expression
while(true) { body }	Keep looping forever	loop or loop *
for i = n to m { body }	Repeat (m – n) times	loop n, m
while(booleanExpression) { body }	Repeat while booleanExpression is true	loop [booleanExpression]

repeat
{ body }
while(booleanExpression)
	Execute once then repeat while booleanExpression is true	loop 1, * [booleanExpression]

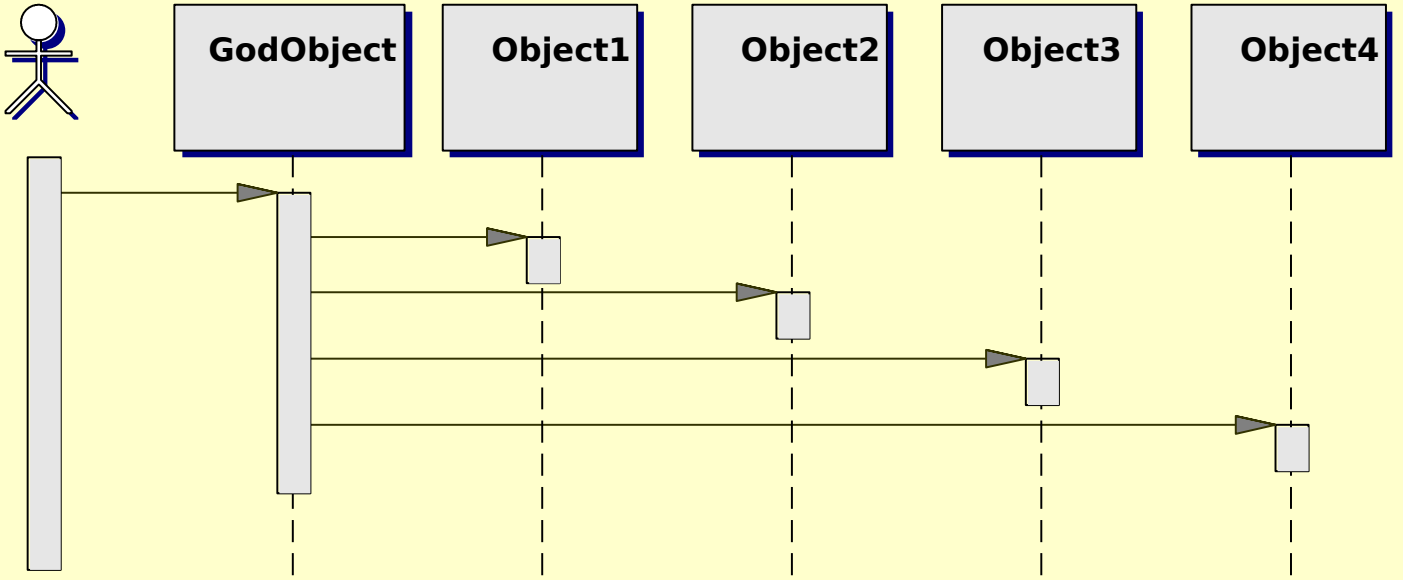
forEach object in collection { body }	Execute the body of the loop once for each object in a collection of objects	loop [for each object in collectionOfObjects]
forEach object of class { body }	Execute the body of the loop once for each object of a particular class	loop [for each object in ClassName]

Stav objektu

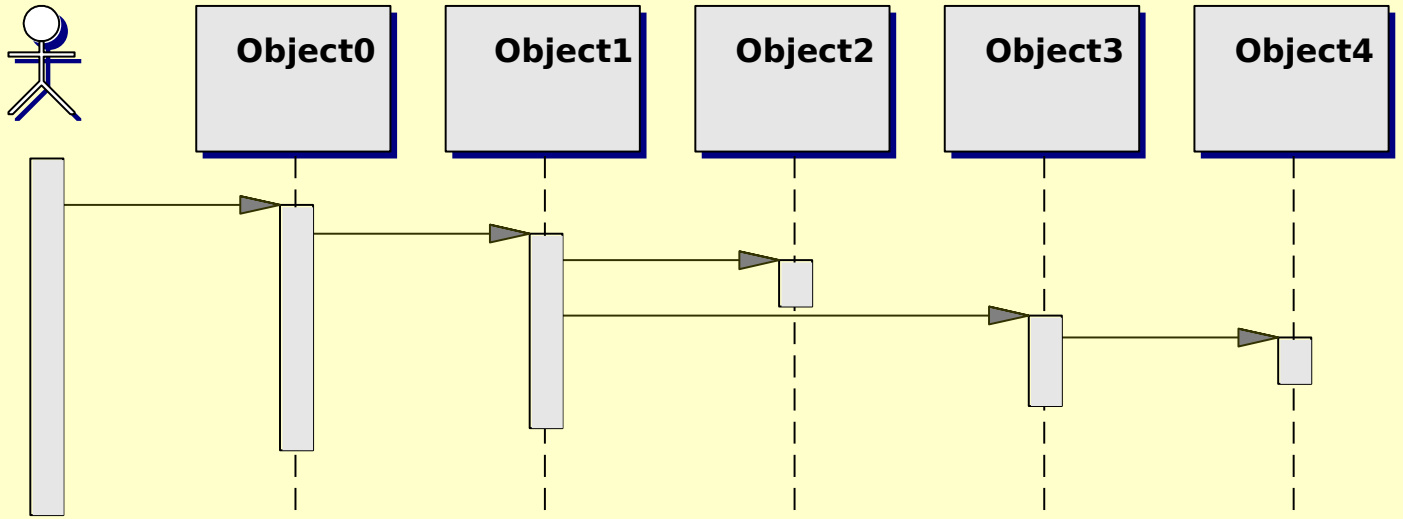


Distribuce “inteligence” objektů

vidlička:
GodObject musí znát mnoho rozhraní a je na nich závislý; koncentruje přílišnou inteligenci, je složitý.

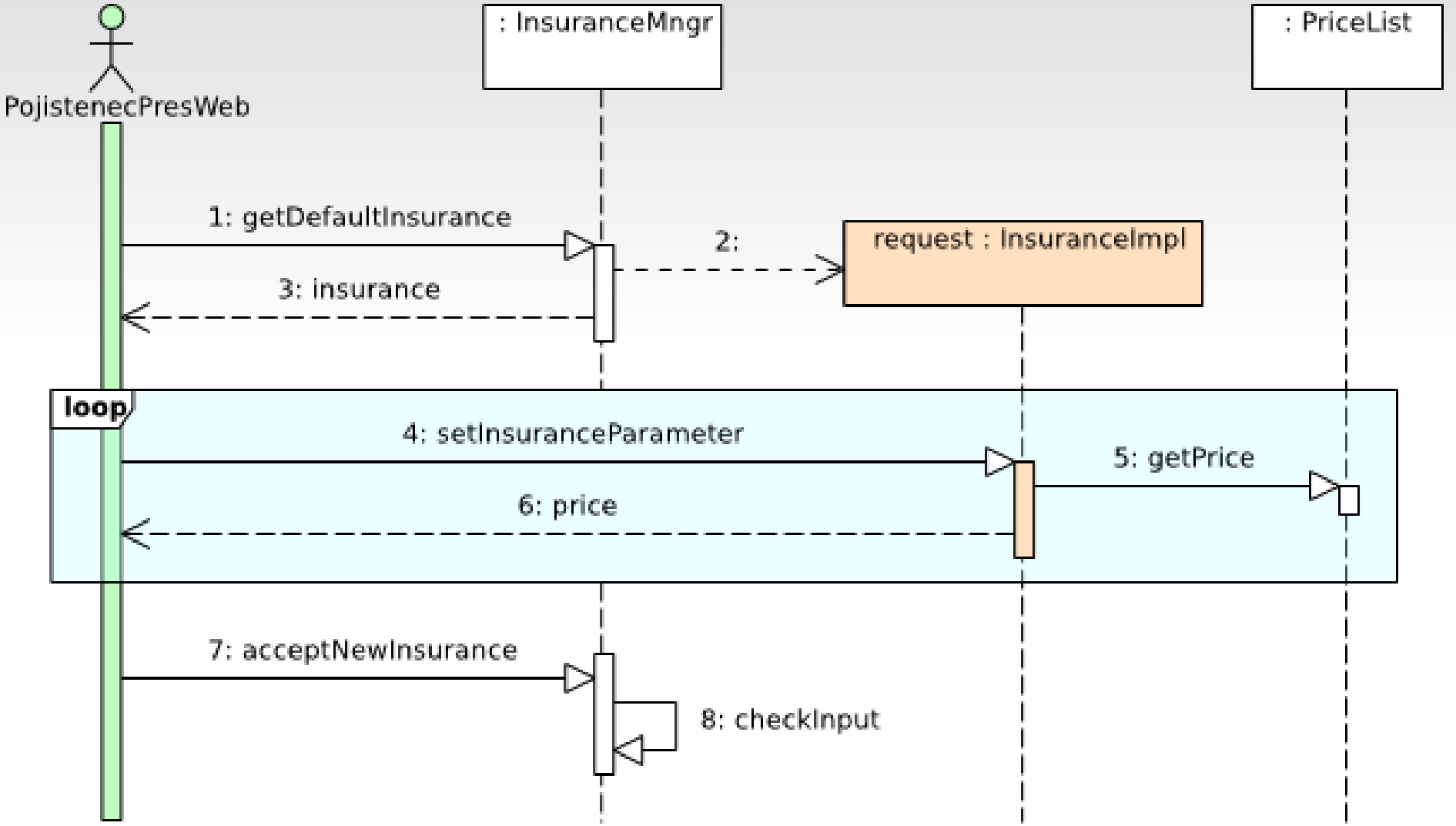


schody:
inteligence je rovnoměrněji rozprostřena mezi objekty, objekty jsou jednodušší



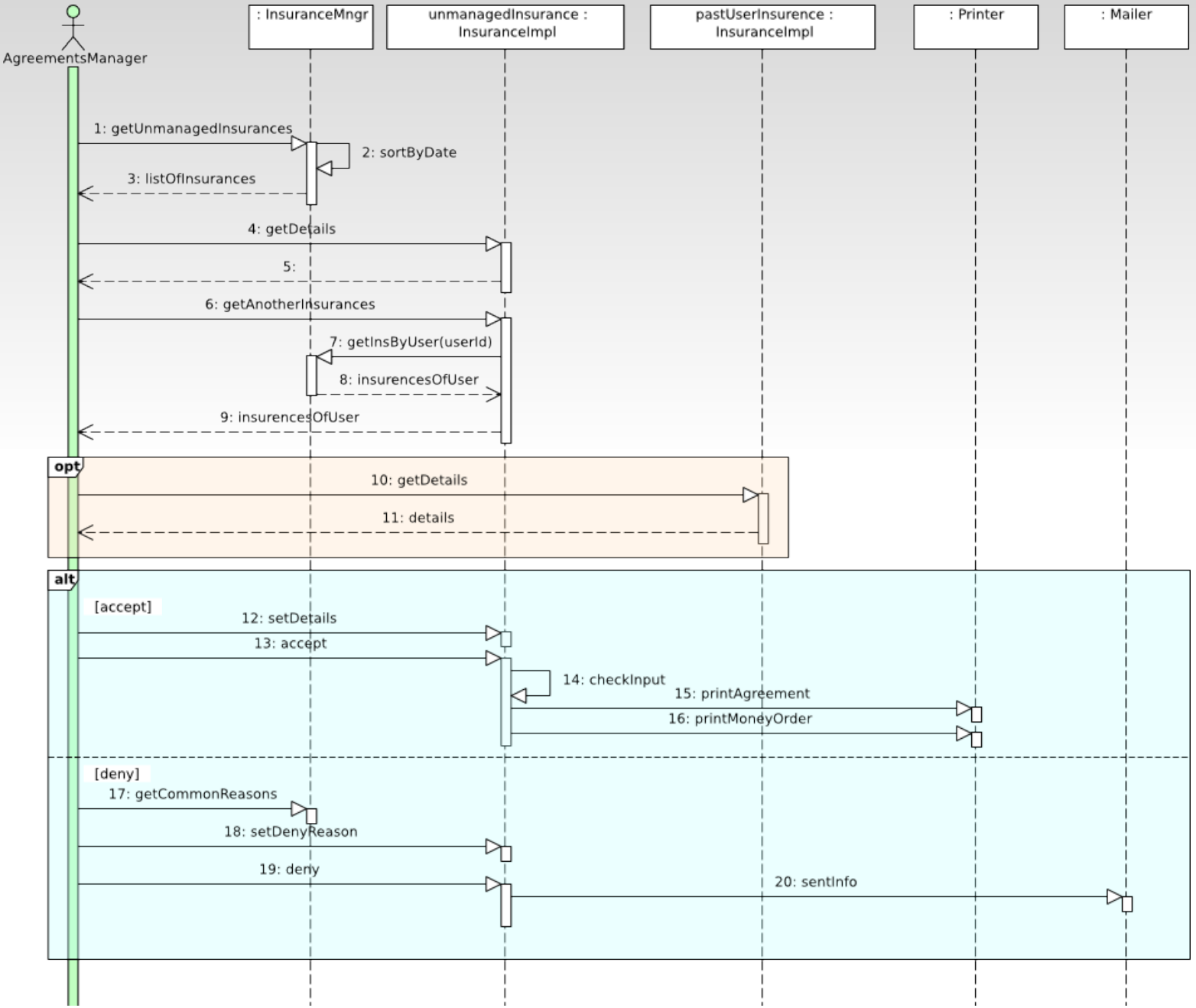
Pojišťovna: sekvenční diagram (I)

Demo



Pojišťovna: sekvenční diagram (II)

Demo



Komunikační diagram
(Communication Diagram)

Komunikační diagram

Dříve (UML < 2.0): *Object Collaboration Diagram*

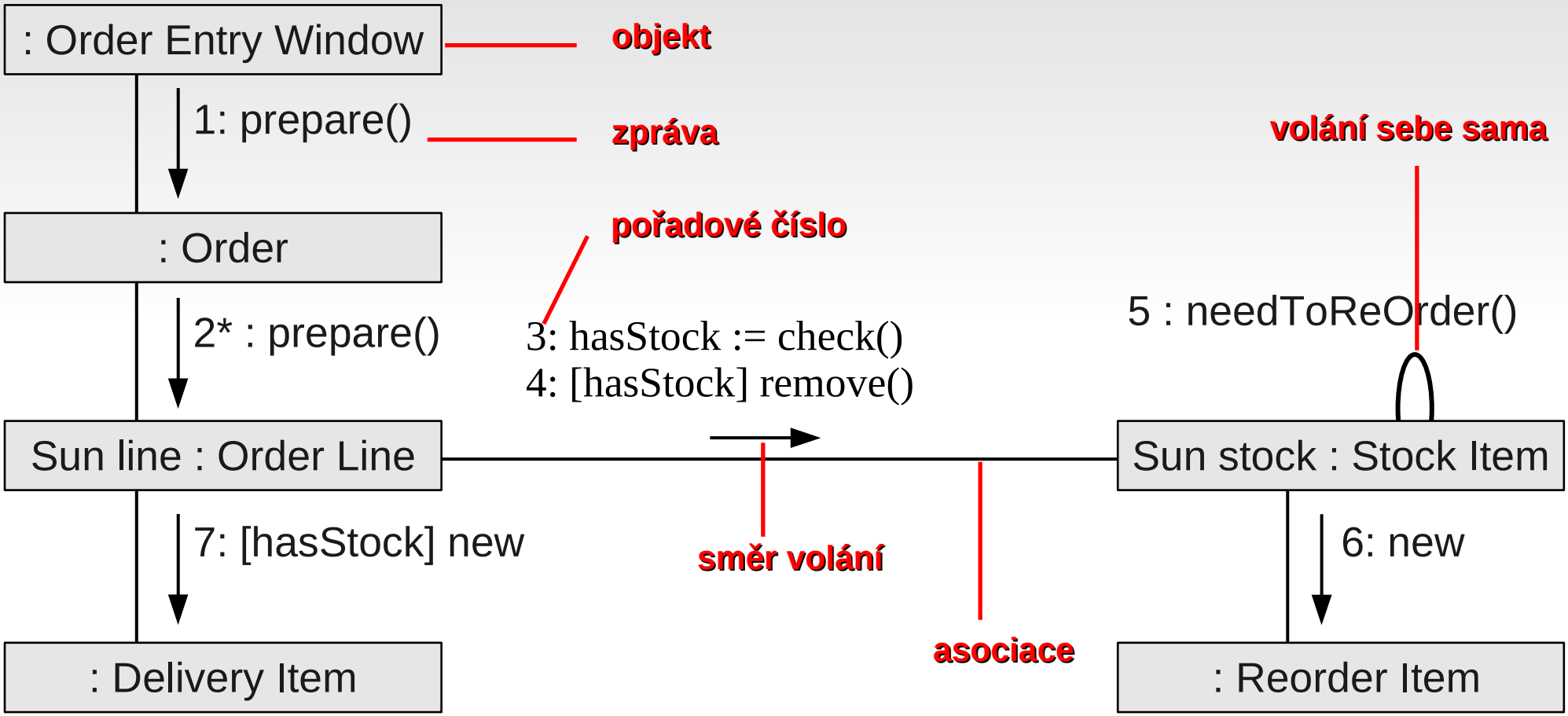
Komunikační diagram ukazuje **interakci** organizovanou podle interagujících objektů, a jejich **propojení** mezi sebou.

- ukazuje **vztahy** mezi rolemi objektů
- **časová dimenze** není ukázána

Bez časové osy ⇒ **sekvenční čísla** jsou použita pro uspořádání zpráv.

- **procedurální tok řízení**: vnořené číslování podle vnořeného volání
- **neprocedurální tok**: nevnořené číslování, které indikuje pořadí zpráv a synchronizaci mezi vlákny, detaily viz knihy o UML

UML notace

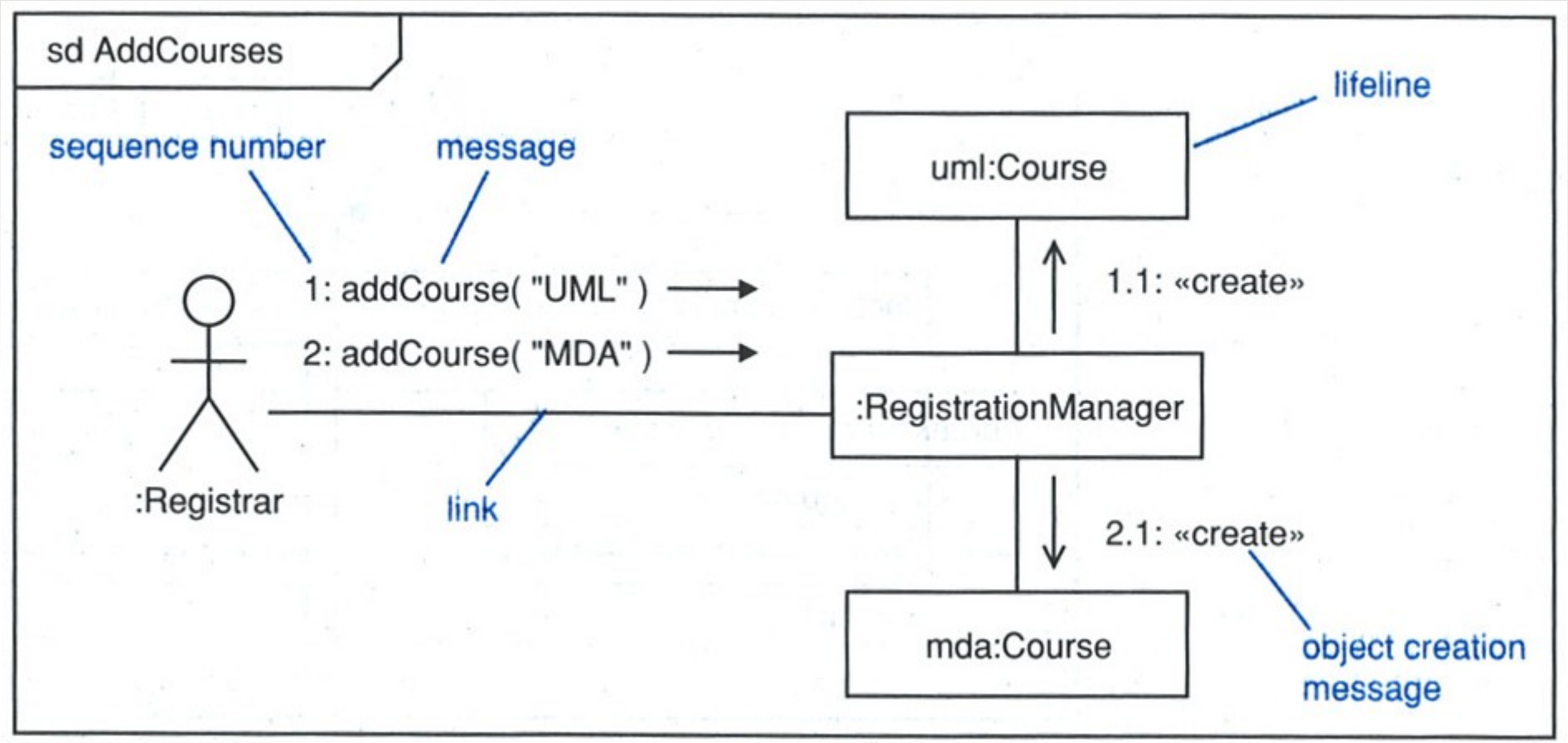

: Order Entry Window
objekt
1: prepare()
zpráva
volání sebe sama
: Order
pořadové číslo
2* : prepare()
3: hasStock := check()
4: [hasStock] remove()
5 : needToReOrder()
Sun line : Order Line
Sun stock : Stock Item
7: [hasStock] new
směr volání
6: new
asociace
: Delivery Item
: Reorder Item

Zprávy

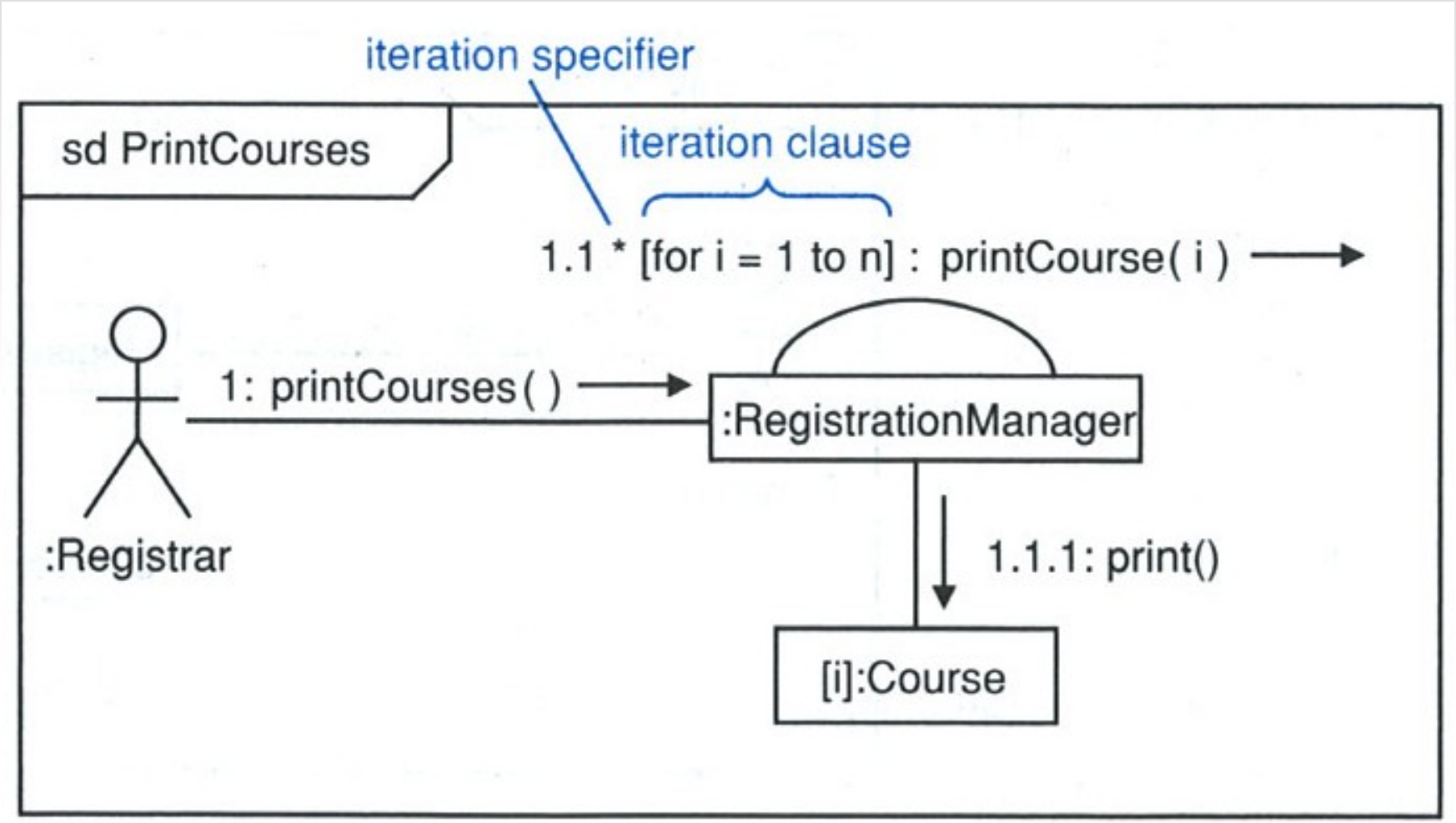
2.1b: *[i:=1..akt_rok] vsechny_predmety := vyhledej(i) // poznamka

- Číslo sekvence
  - lexikografické číslování
  - písmena označují paralelní zprávy
  - * indikuje iteraci
- Podmínka, cykly
  - uvnitř []
- Jména návratových hodnot
  - jsou uvedena před :=
- Jména návratových hodnot
  - jsou uvedena před :=
- Jméno zprávy
  - může být událost nebo operace
- Argumenty
  - v kulatých závorkách
- Poznámky
  - za dvěma lomítky

Příklad – vytváření instancí



Příklad – cykly (iterace)



Příklad – větvení

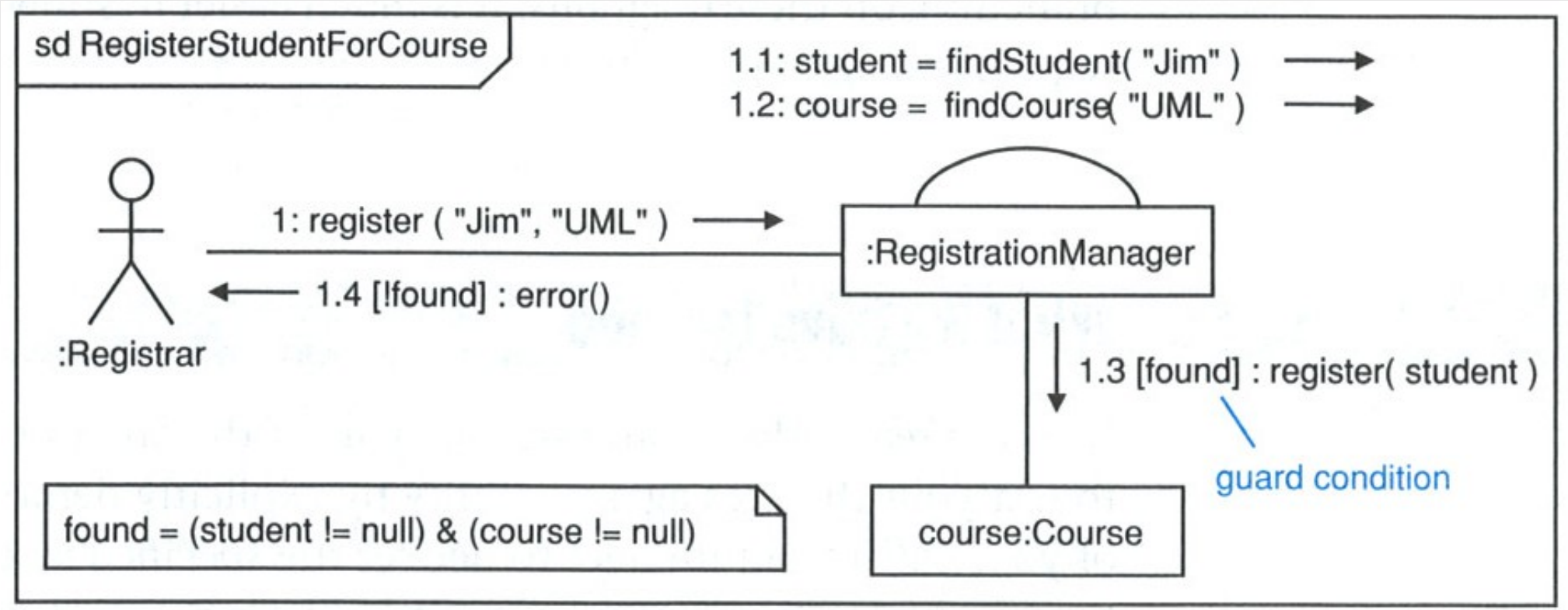
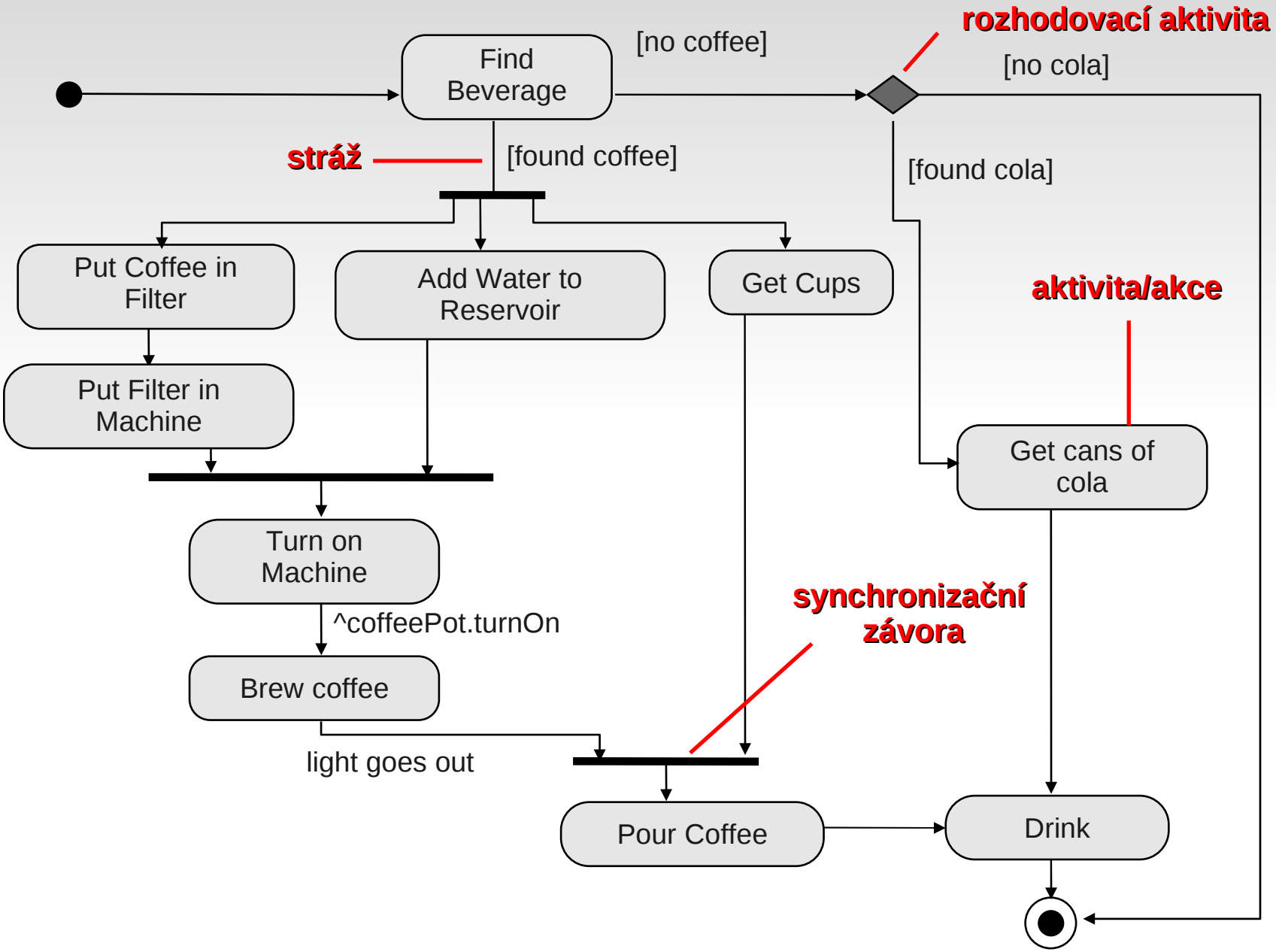


Diagram aktivit
(Activity diagram)

Diagramy aktivit

- „OO flowcharts“, sémantika založená na Petriho sítích.
- Aktivita = posloupnost akcí
- Používají se v mnoha tocích práce (workflows) UP
  - během „business modeling“ fáze na modelování business procesů
  - během analýzy na dokumentaci případů užití
  - běhen návrhu na modelování detailů operací a algoritmů
- Používají se pro:
  - popis akcí případu užití
  - modelování toků mezi případy užití
  - popis algoritmických aspektů operací nad objektem
  - popis toku dat systémem
  - modelování obchodních procesů (business processes)
  - popis chování komponent
  - ...

UML notace

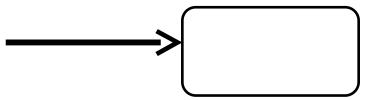

rozhodovací aktivita
Find Beverage
[no coffee]
[no cola]
stráž
[found coffee]
[found cola]
Put Coffee in Filter
Add Water to Reservoir
Get Cups
aktivita/akce
Put Filter in Machine
Get cans of cola
Turn on Machine
^coffeePot.turnOn
synchronizační závora
Brew coffee
light goes out
Pour Coffee
Drink

UML notace (II)

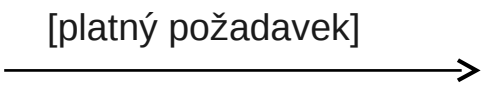
začátek aktivity

konec aktivity

akční uzel – krok v aktivitě



kontrolní tok – přechod mezi akcemi

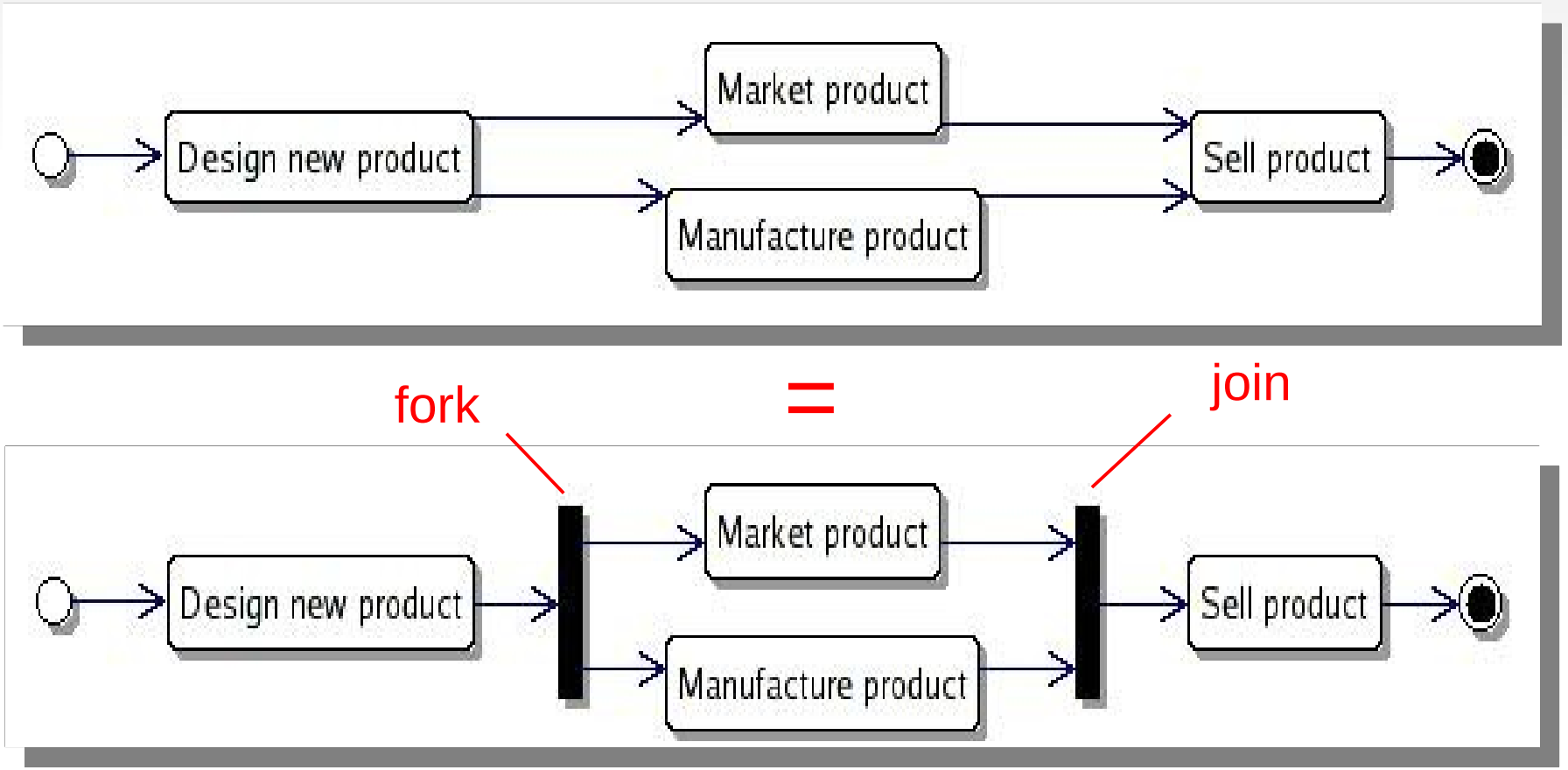


podmínka přechodu

Přechody a fork/join

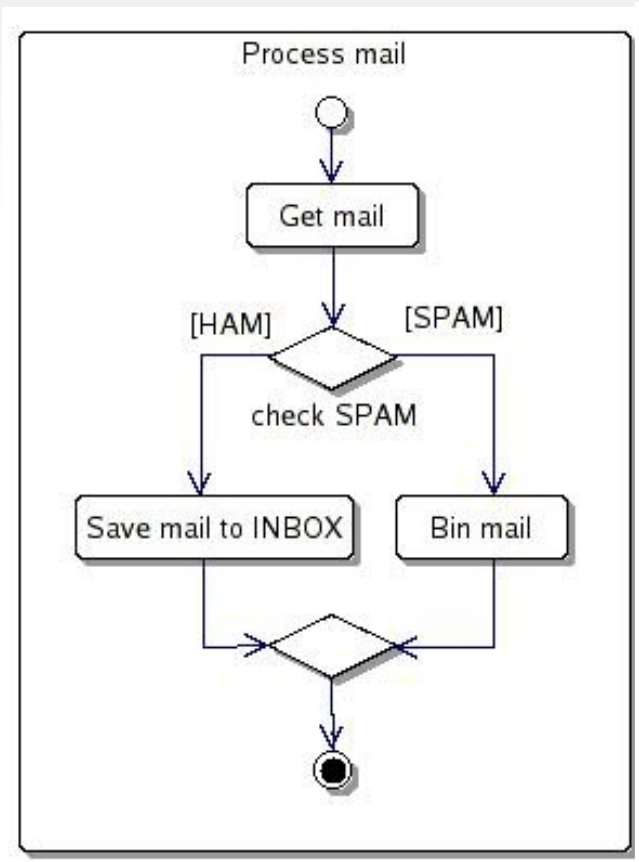
- Akční uzly čekají na všechny vstupy a pak pokračují na všech výstupech současně

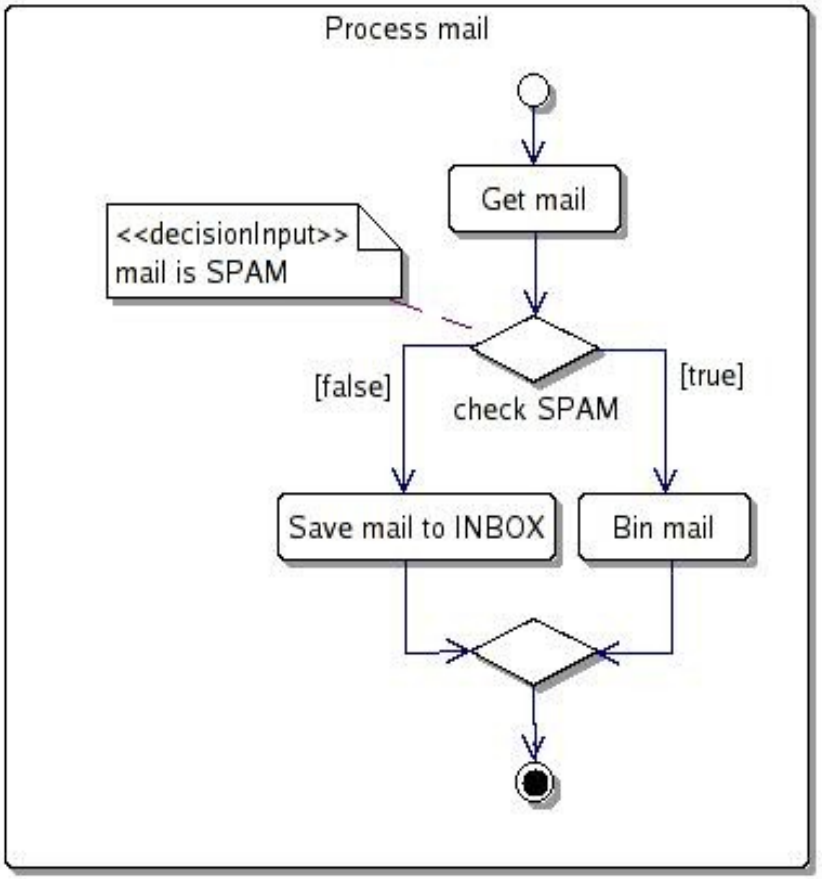
- => Implicitně se AD chová paralelně, tj. jako s *fork/join:*

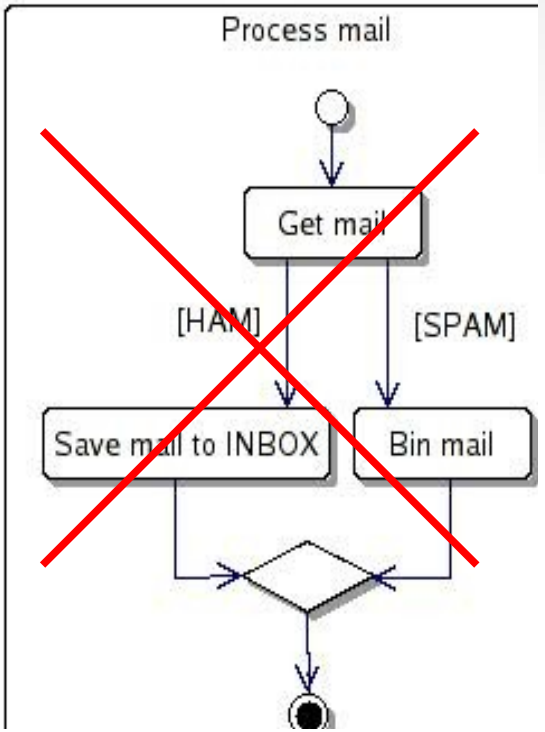


Rozhodovací a slučovací uzly

- Rozhodovací uzel pokračuje **právě jednou** cestou
  - stráže musí být vzájemně výlučné a úplné
- Slučovací uzel pokračuje při libovolném vstupu (OR)
- Ekvivalentní diagramy:



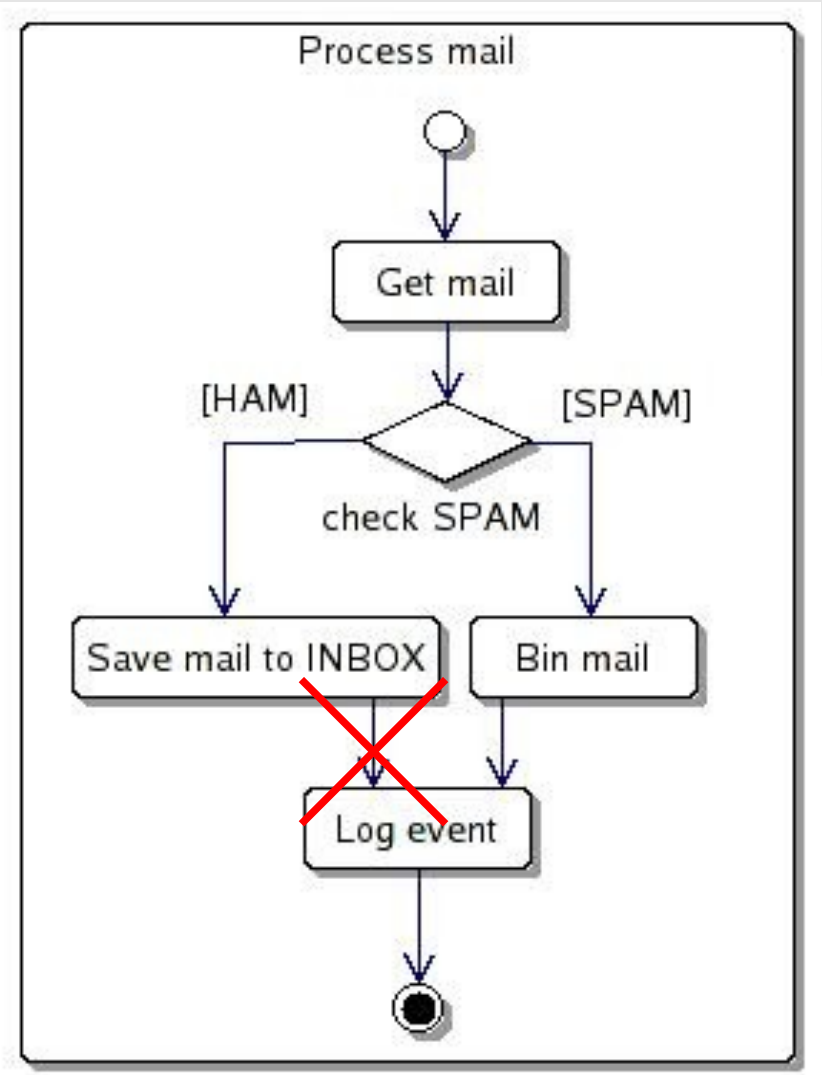




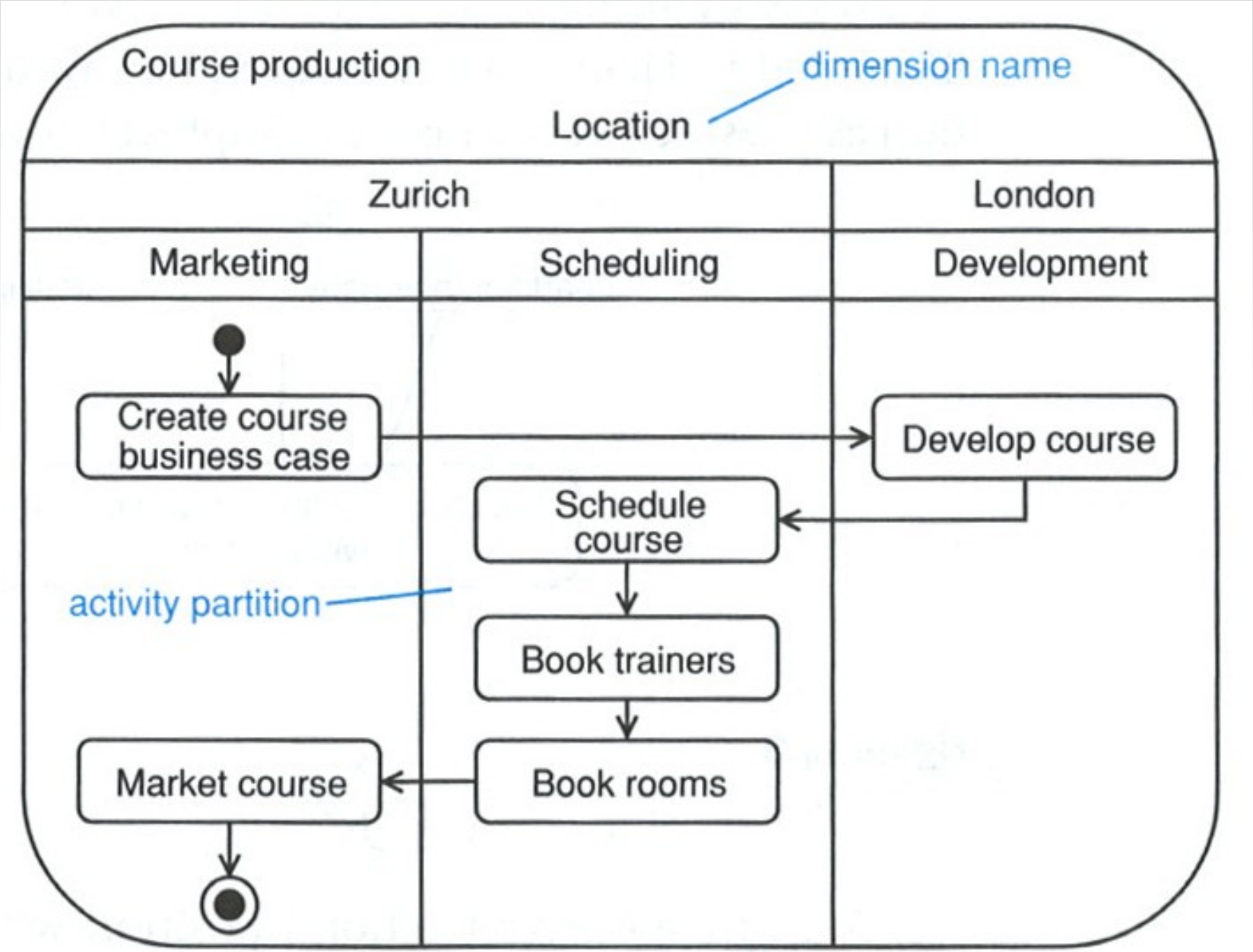
nedoporučuje se

Slučovací uzel

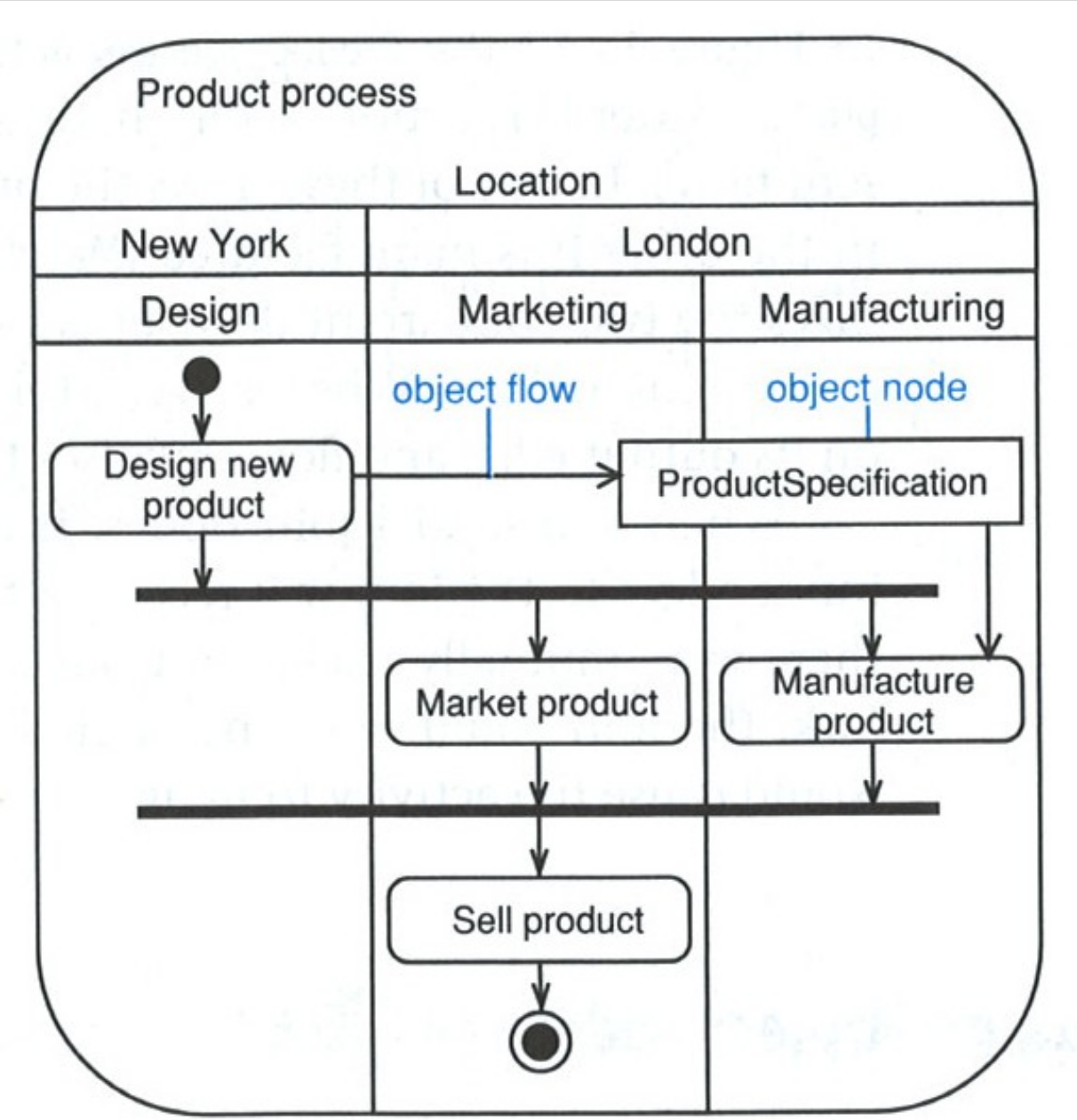
- Smysl použití slučovacího uzlu:
  - akční uzly čekají na všechny vstupy!



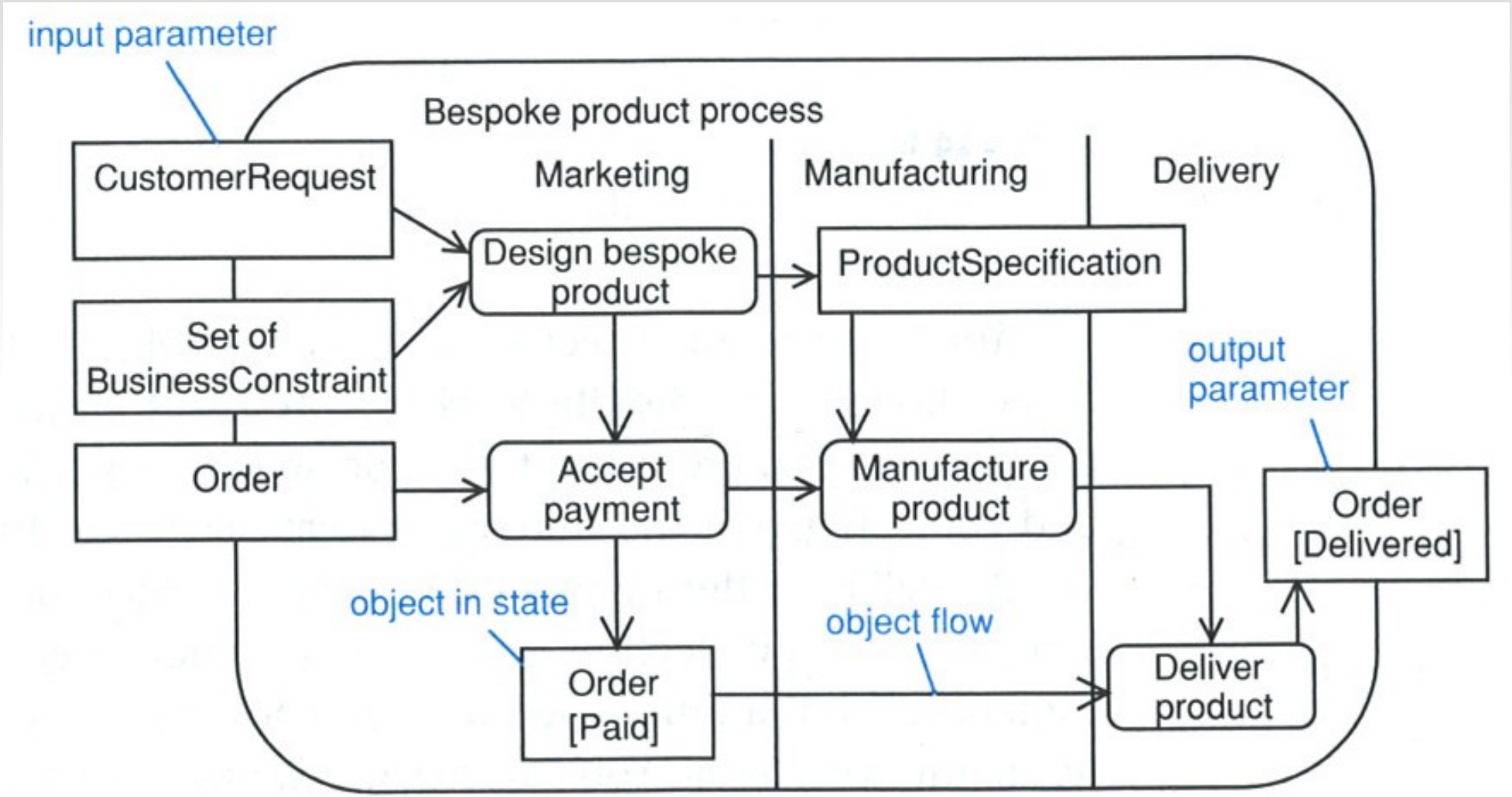
Segmenty (partitions)



Objektové uzly



Parametry aktivit



Signály

Příjem signálu a vyslání signálu

Signál není nic jiného, než instance třídy (stereotyp <<signal>>)

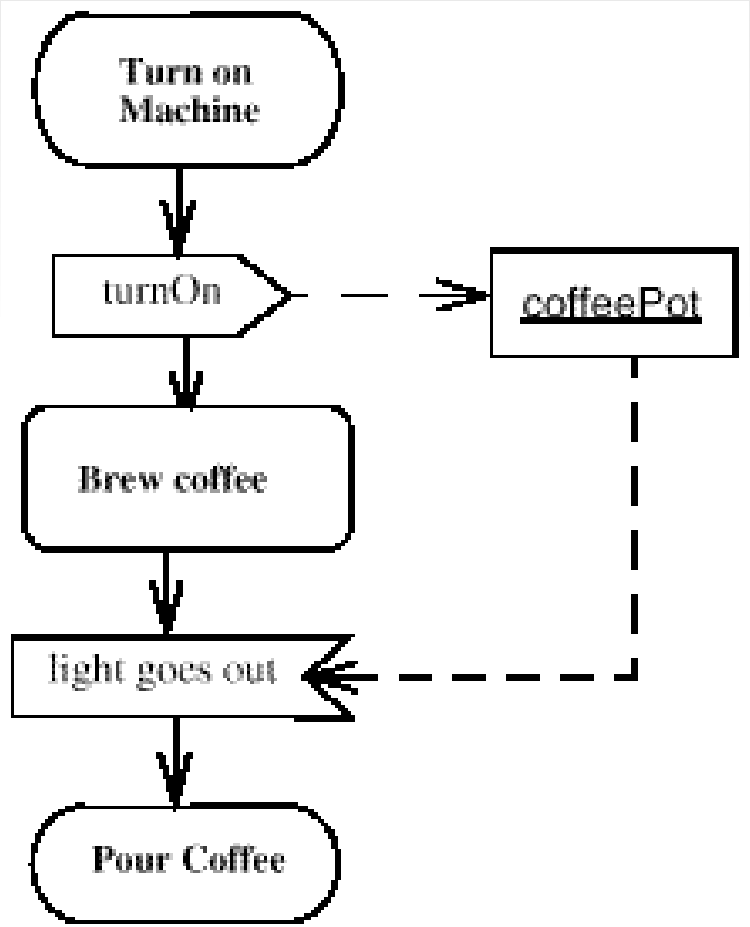


Diagram přehledu interakcí
(Interaction Overview Diagram)

Diagram přehledu interakcí