UML (3)

dynamické modely, diagramy interakcí

© 2008 Radek Ošlejšek
FI MU, Brno

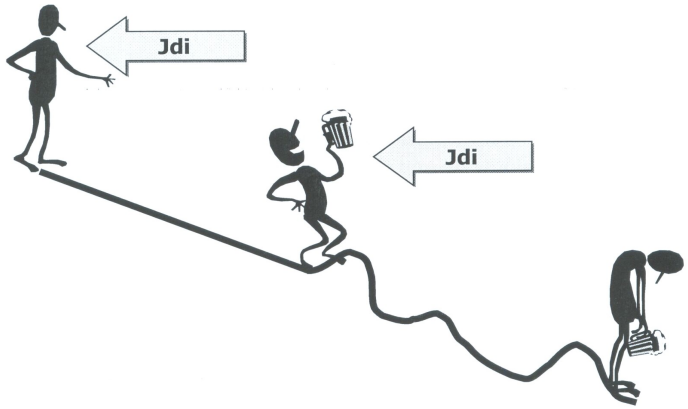
oslejsek@fi.muni.cz
http://www.fi.muni.cz/~oslejsek/PA103

Stavový diagram
(State Transition Diagram)

Stavové diagramy

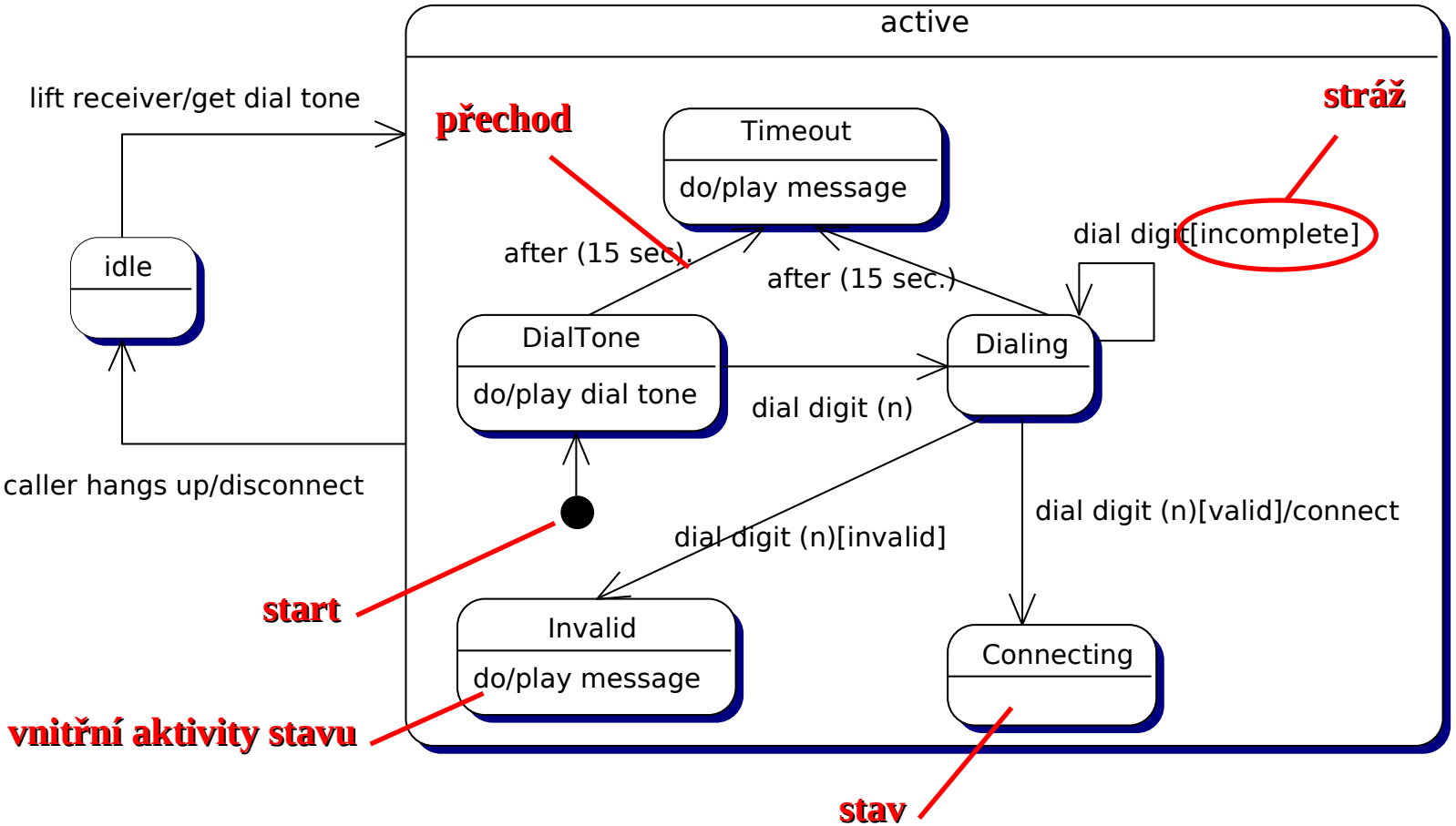
State Transition Diagram – STD

Stavový diagram reprezentuje **stavový automat,** který je grafem **stavů** a **přechodů** a popisuje **odezvu objektu** dané třídy na přijetí **vnějšího stimulu**.



- jeden diagram pro každou třídu vykazující zvláštní (důležité) chování
- zpráva je stimul odpovídající události na kterou má objekt reagovat
- objekt reaguje změnou svého stavu a/nebo provedením operace
  - objekt může reagovat na stejnou zprávu různě v závislosti na stavu
- stavový diagram reprezentuje šablonu pro všechny objekty třídy

STD: notace



přechod: událost [stráž] / akce

STD: přechody

- **Notace:**

  událost (seznam_argumentů) [stráž] / akce_výraz ^ vysílací_doložky

- **Událost**
  - vnější stimul, který může vést ke změně stavu
  - libovolné jméno, kromě ***entry***, ***exit*** a ***do***
  - Př: *stisknuté tlačítko (n)*
- **Stráž** (nepovinná)
  - podmínka je platná v určitém časovém rozmezí a tudíž **není stimulem**
  - booleovský výraz používající parametry spouštěcí události a atributy, propojení nebo stavy objektu, kterému patří stavový diagram, nebo objektu, který je dosažitelný (pomocí propojení)
  - Př. špatně: *teplota > 100*
  - Př. správně: s*ignál z čidla (teplota) [teplota > 100]*
- **Akce_výraz** (nepovinná)
  - reakce na událost
  - atomická nepřerušitelná operace, operace jsou prováděny sekvenčně
  - mohou přiřazovat hodnoty atributům a spojením
- **Vysílací_doložka** (nepovinná)
  - Př: *cílový-výraz . jméno-operace-nebo-signálu (seznam-argumentů)*

STD: příklady přechodů

Příklad komplexního přechodu mezi stavy:

right-mouse-down(location) [location in window] /
object := pick-object(location) ^ object.highlight()

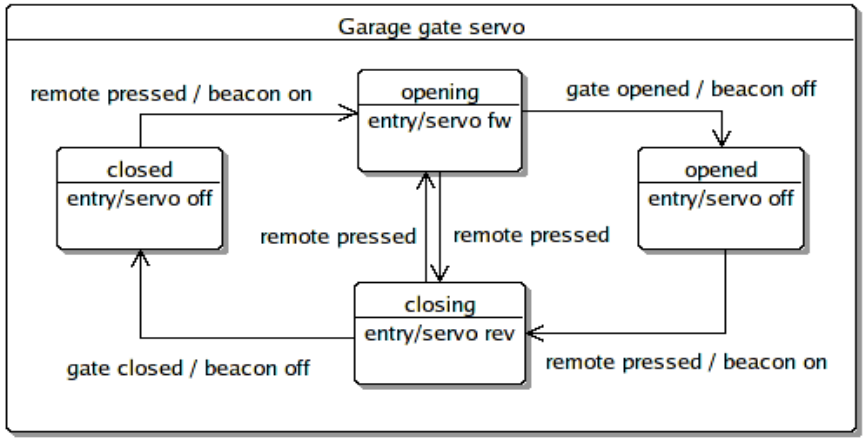
Příklady událostí:

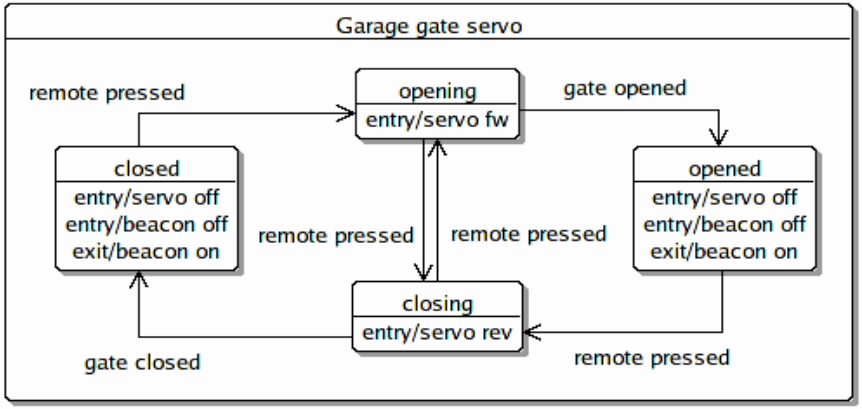
- objekt přijme volání operace (*CallEvent*)
  Př: *jméno-operace (seznam parametrů), tj. startAutopilot(normal)*
- explicitní signál od jednoho objektu nebo systému k jinému (*SignalEvent*); signály jsou asynchronní, vznikají, vznikají a zanikají, mají parametry a hierarchie dědičnosti => jsou modelovány jako třídy se stereotypem *<<signal>>*
  Př: *jméno-třídy (seznam parametrů), tj. Collision(5.3)*
- vypršení stanovené doby od výskytu jiné události (*TimeEvent*)
  Př: *after (2 seconds), at (11:49PM)*
- podmínka je splněna (*ChangeEvent*)
  Př: *when (altitude < 1000)*

STD: Vnitřní události stavu

1) Vyvolání vnořeného stavového automatu (událost ***do***):
- **do *l jméno_vnořeného_stavového_automatu (seznam argumentů)***
- vyvolání stavového automatu zakresleného v jiném diagramu
  ==> jeden ze způsobů rozdělení diagramu
- ostatní aktivity považujte za stavové automaty „bez diagramu“

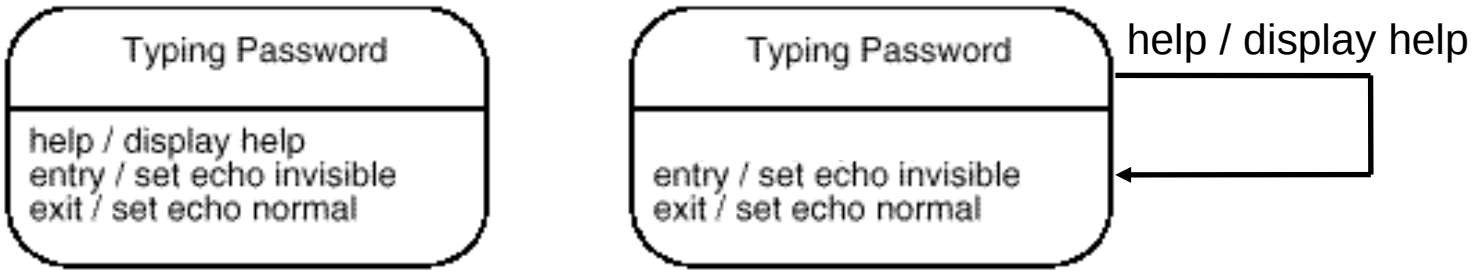
2) Vstupní a výstupní akce (událost ***entry*** a ***exit***):
- alternativa k zobrazování akcí v přechodech
- používá se pokud všechny přechody z/do stavu vykonávají stejnou akci
- nepoužívat na úrovni analýzy, ale až v návrhu





STD: Vnitřní události stavu (II)

3) Ostatní vnitřní události:
- události, které nastanou v daném stavu a *ponechávají původní stav*
- alternativa k externí akci na lokálním přechodu
- Př: když nastane událost ‘help’, je provedena akce ‘display help’:

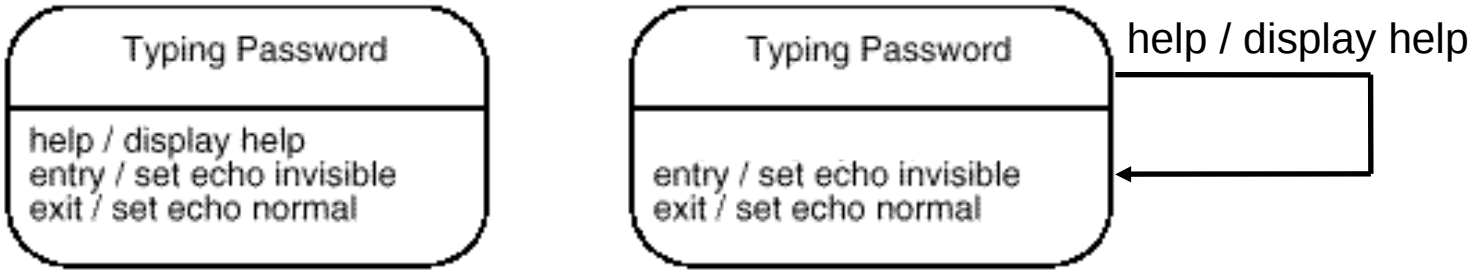


rozdíl mezi těmito diagramy?

STD: Vnitřní události stavu (III)

3) Ostatní vnitřní události:

- události, které nastanou v daném stavu a *ponechávají původní stav*
- alternativa k externí akci na lokálním přechodu
- Př: když nastane událost ‘help’, je provedena akce ‘display help’:



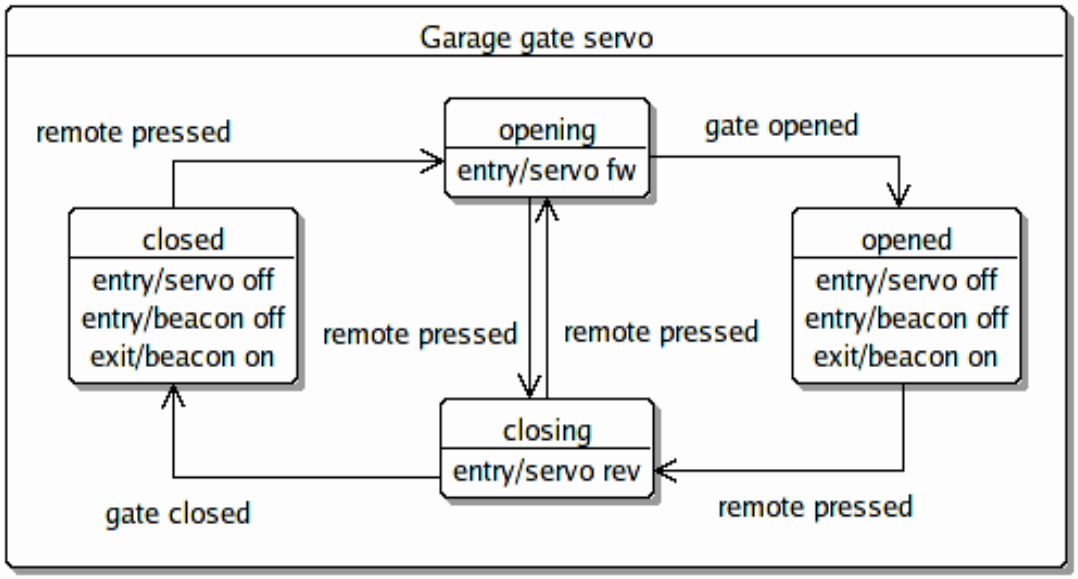
rozdíl mezi těmito diagramy?

- ***externí akce*** opouští stav a znovu do něj vstupuje
  => spustí se ***enty*** a ***exit*** akce
- ***entry***, ***exit*** a ***interní akce*** často vedou na privátní operace dané třídy,
  externí akce často vedou na veřejné operace

STD: Pořadí vykonávání akcí

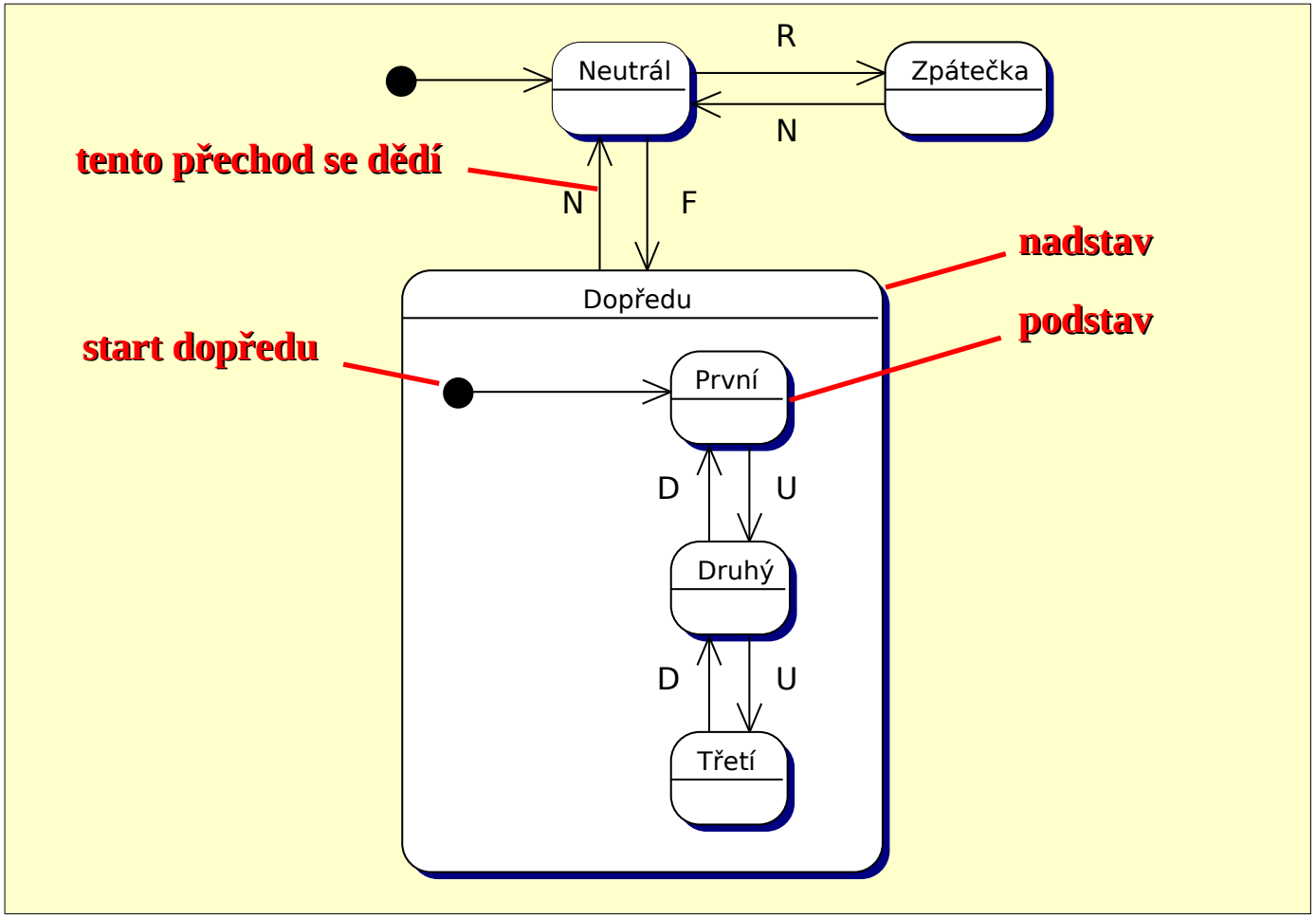
Pořadí vykonávání akcí:

1. akce vstupních přechodů
2. *entry* akce
3. Vnitřní akce a *do* (vnořené STD)
4. *exit* akce
5. akce výstupních přechodů



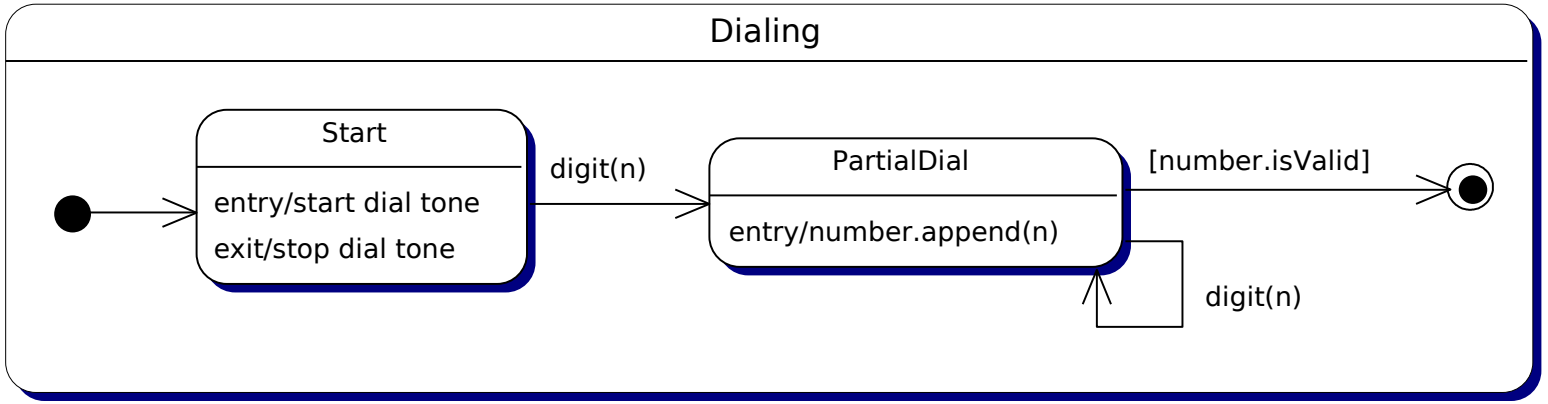
STD: vnořené stavy = generalizace

- Podstavy dědí všechny přechody nadstavů, mohou je ale předefinovat
- Použití generalizace je velmi vhodný postup pro zpřehlednění složitých STD



STD: vnořené stavy (*or*-vztah)

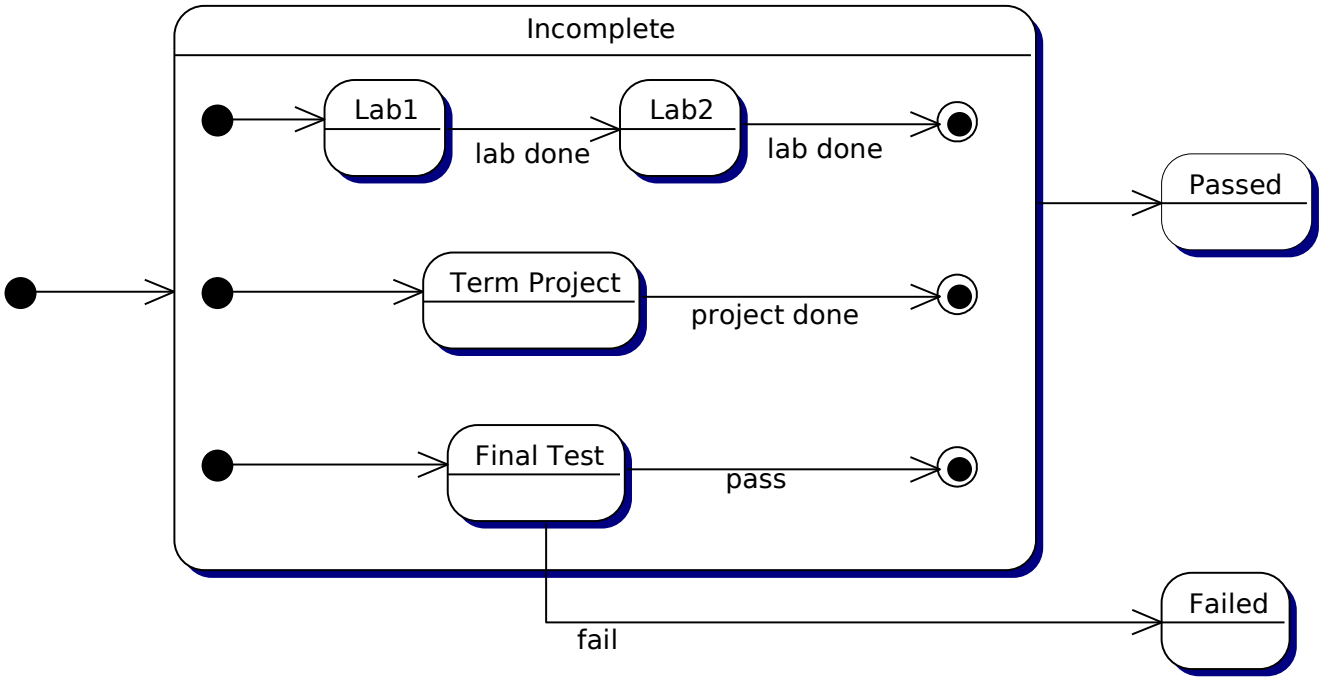
- stav ‘Vytáčení’ je rozložen s použitím ***or*-vztahu** na **vzájemně výlučné dílčí stavy**



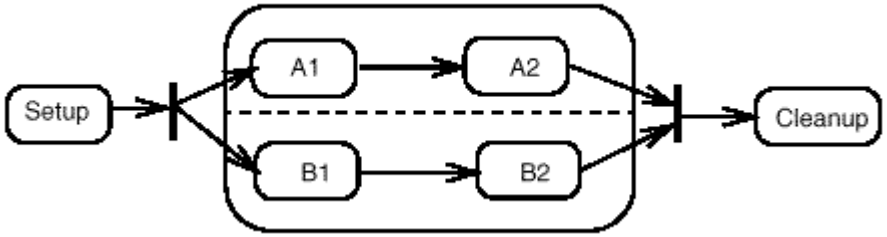
- **počáteční** stav - **koncový** stav během ‘Vytáčení’
- **aktivity** dány **pomocí vstupní a výstupní akce**
- **události** mohou být „metodami“ určitých tříd
- co **popisuje** stavový diagram ? metodu, třídu, …?

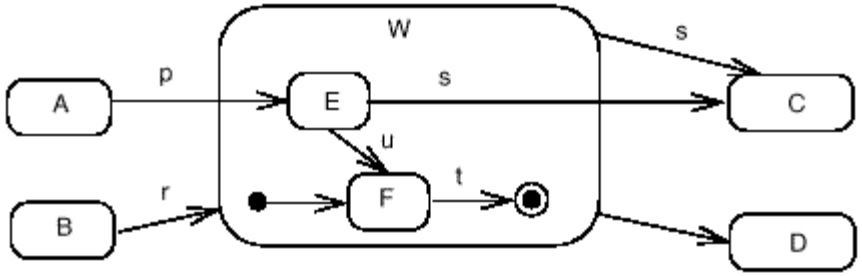
STD: vnořené stavy (*and*-vztah)

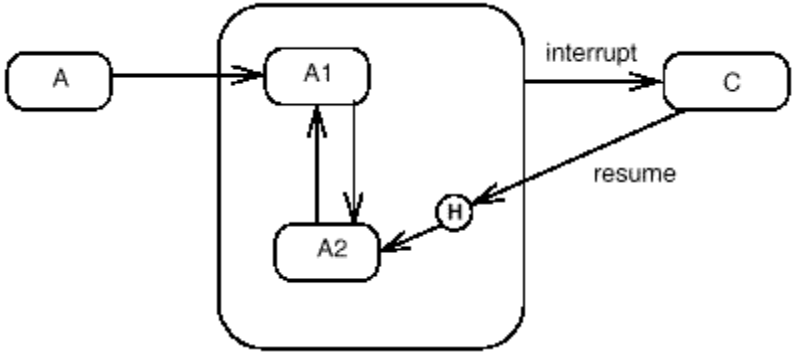
- stav ‘Incomplete’ je rozložen s použitím ***and*-vztahu** na **souběžné dílčí stavy**
- paralelní stavy uvnitř jedné třídy => třída má více zodpovědností => je dobré třídu přezkoumat



STD: komplexní přechody

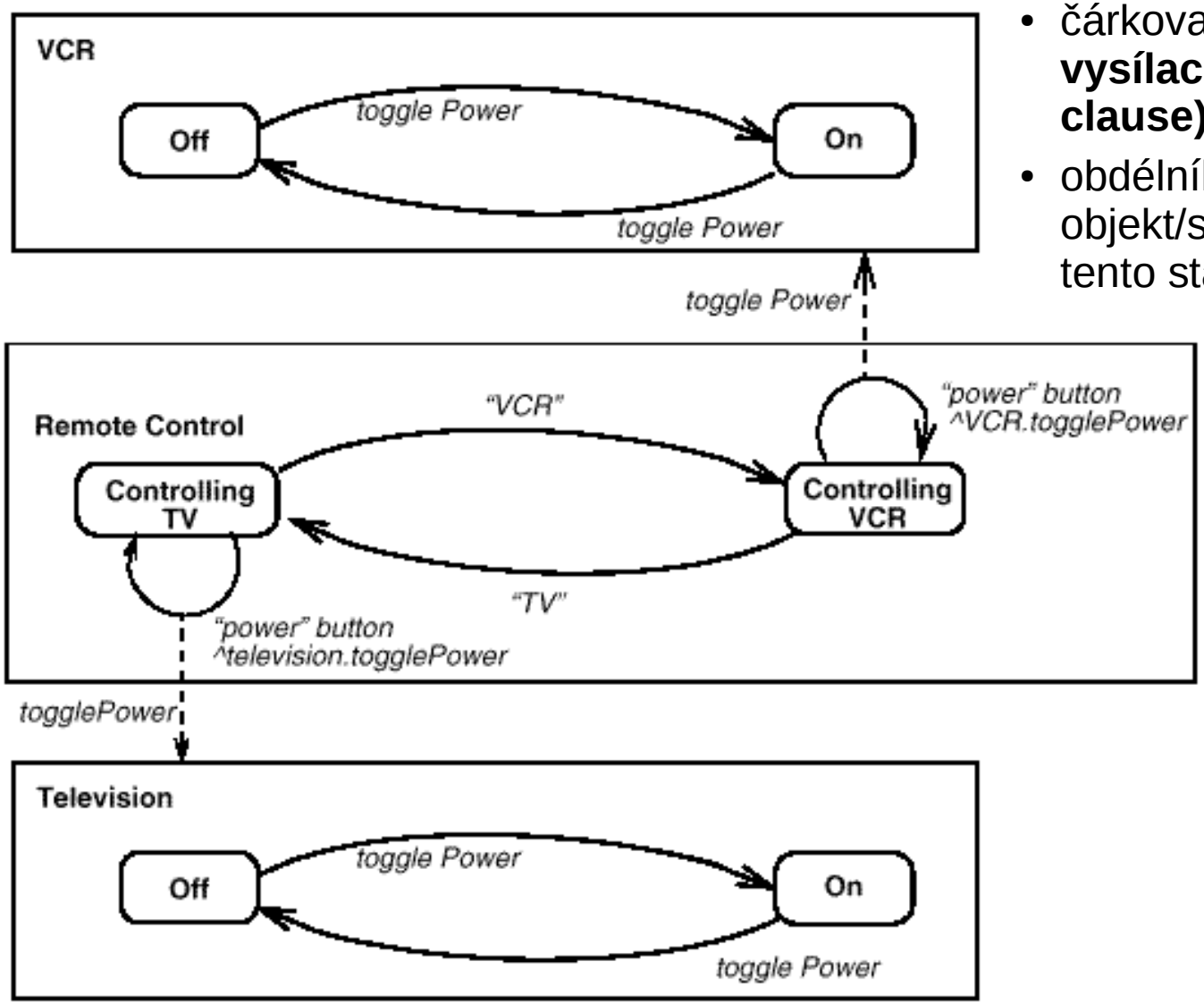






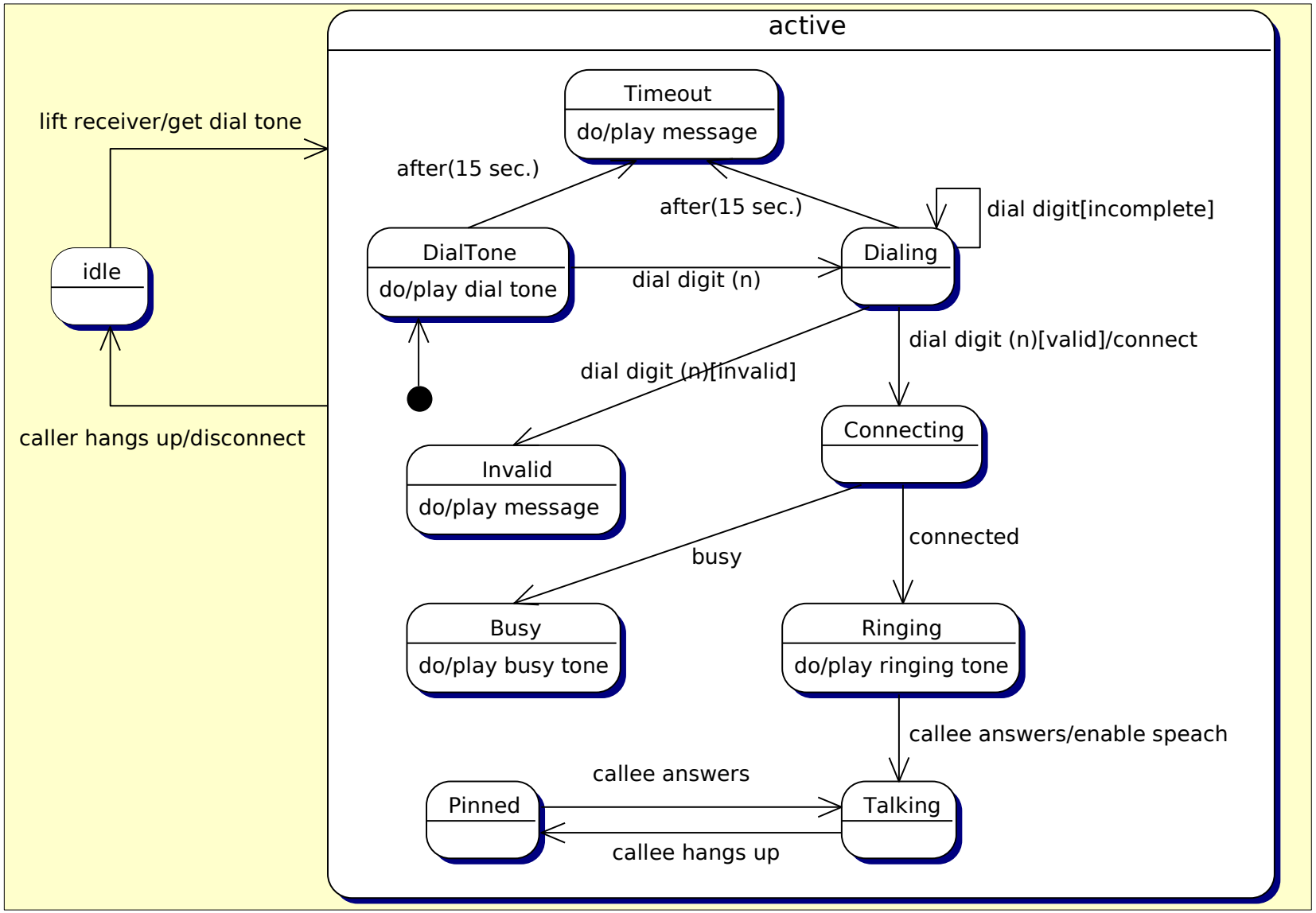
- **komplexní přechody**: všechny události před závorou musí nastat, než dojde k přechodu do stavů za závorou
- **dědění přechodů**: podstavy E a F dědí přechod s událostí 's', podstav E tento zděděný přechod navíc předefinovává
- **indikátor historie**: A2 je vybráno poprvé, později po *resume* následuje přechod do stavu, který byl aktivní předtím, než došlo k přerušení

Interakce ve stavových diagramech



- čárkovaná šipka označuje **vysílací-doložku (send-clause)**
- obdélníky označují objekt/subsystém, k němuž tento stav přináleží

STD: příklad


active
lift receiver/get dial tone
idle
caller hangs up/disconnect
Timeout
do/play message
after(15 sec.)
after(15 sec.)
dial digit[incomplete]
DialTone
do/play dial tone
dial digit (n)
Dialing
dial digit (n)[valid]/connect
dial digit (n)[invalid]
Connecting
Invalid
do/play message
connected
busy
Busy
do/play busy tone
Ringing
do/play ringing tone
callee answers/enable speach
callee answers
Pinned
Talking
callee hangs up

Proč vytváříme STD

- Pochopení posloupnosti chování objektu v průběhu času
- Vyjasňuje závislost chování na stavu
- Odhaluje skryté atributy
- Pomáhá rozpoznat chybějící a skryté operace
- Definuje sekvence operací a zpráv

STD: kontrola konzistence

- Kontrola konzistence STD
  - Každý stav musí mít předchůdce (kromě iniciálního stavu)
  - Stavy bez následníka (kromě koncového stavu) jsou podezřelé
  - Projděte efekty vstupních událostí systému a zkontrolujte soudržnost
  - Ověřte, že stejná událost na různých STD je modelována konzistentně
- Konzistence mezi STD a diagramy tříd
  - Vyskytují se všechny aktivity a akce jako operace na diagramu tříd?
  - Podporuje diagram tříd plně stavové diagramy?
    - existují atributy pro modelované stavy?
    - existují atributy pro vyhodnocení podmínek?
    - jsou pro všechny přechodové akce veřejné operace?
    - jsou pro ***entry***, ***exit*** a ***interní akce*** privátní operace?

Diagram aktivit
(Activity Diagram)

Diagramy aktivit

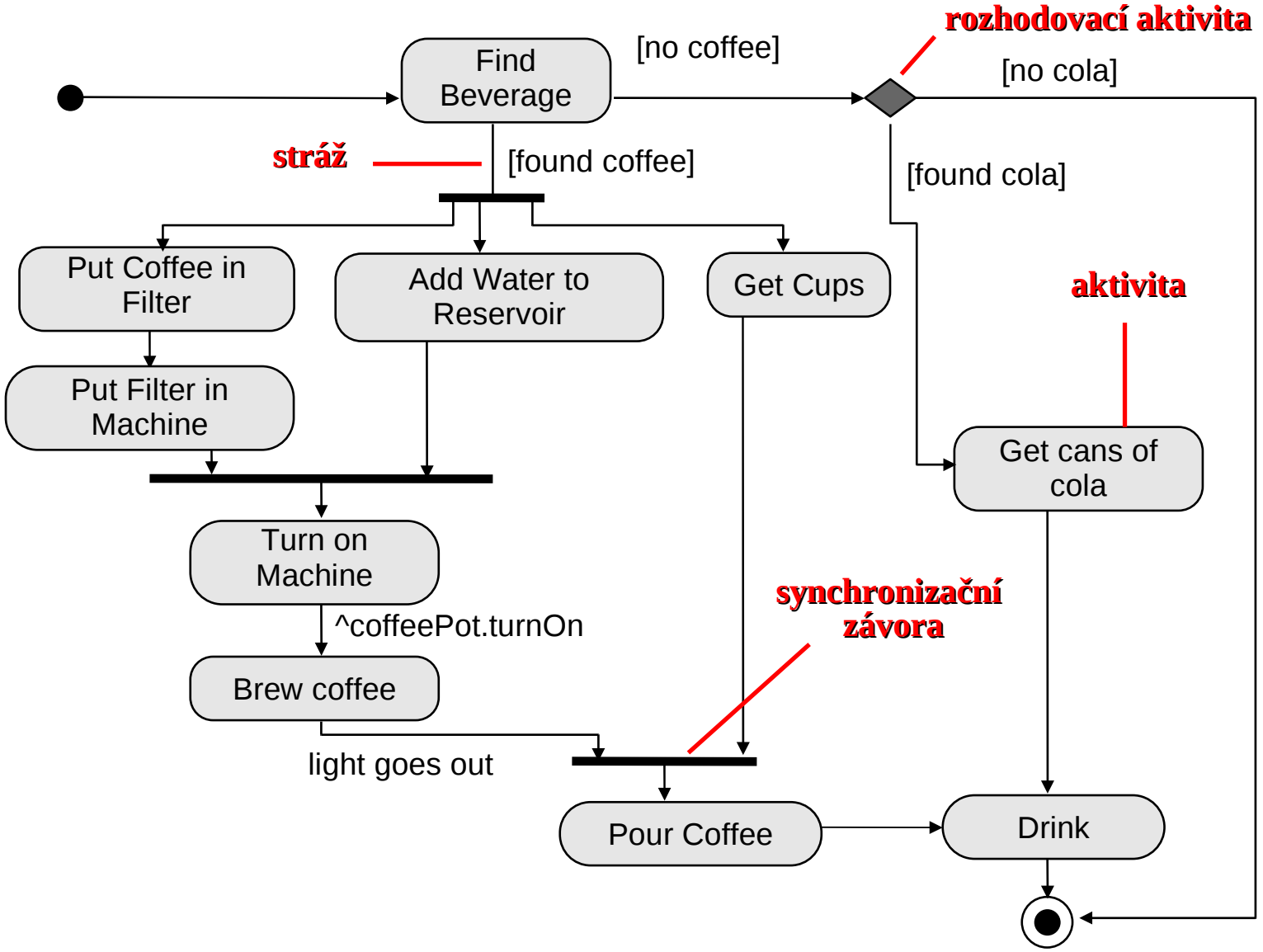
Angl.: *Activity Diagram, AD*

Sémantika založená na Petriho sítích.

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)
- ...

AD: Motivace


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

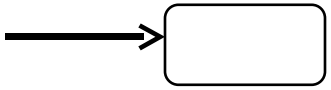
AD: Notace

začátek aktivity

konec aktivity

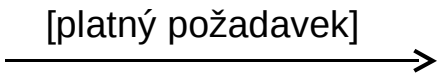
Ověření platnosti požadavku

akční uzel – krok v aktivitě



kontrolní tok – přechod mezi akcemi

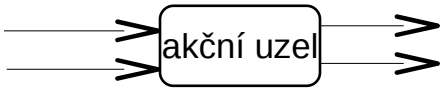
[platný požadavek]



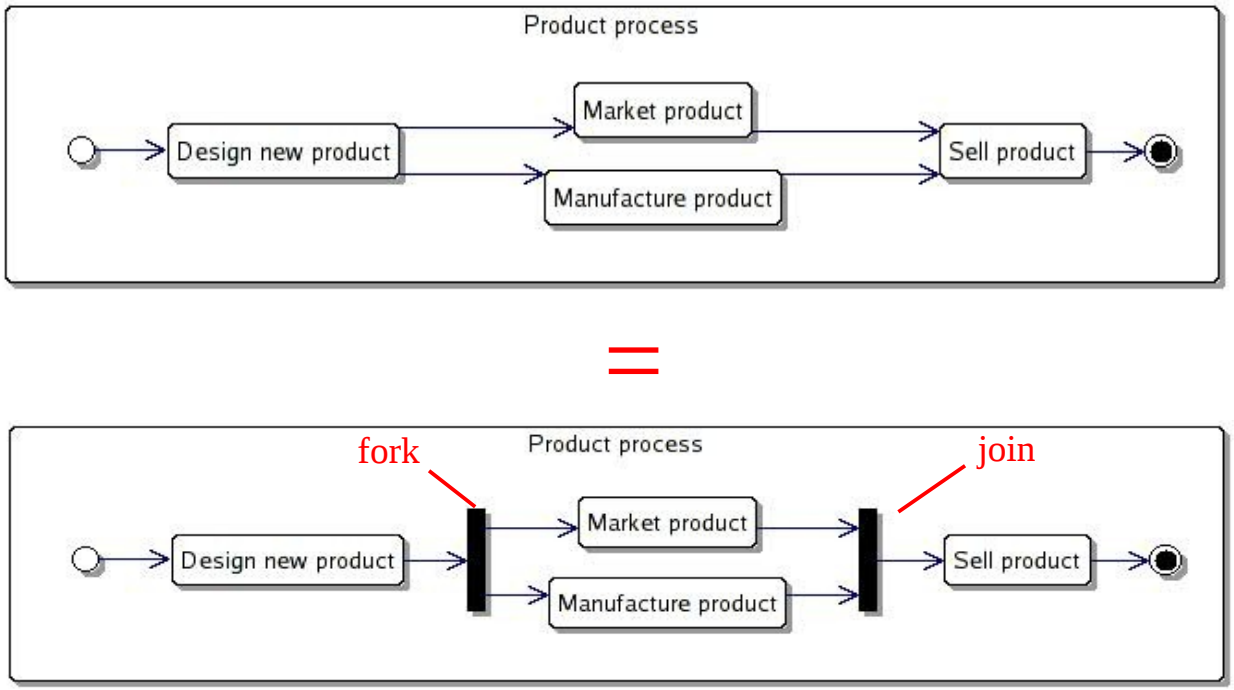
podmínka přechodu

AD: přechody a fork/join

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

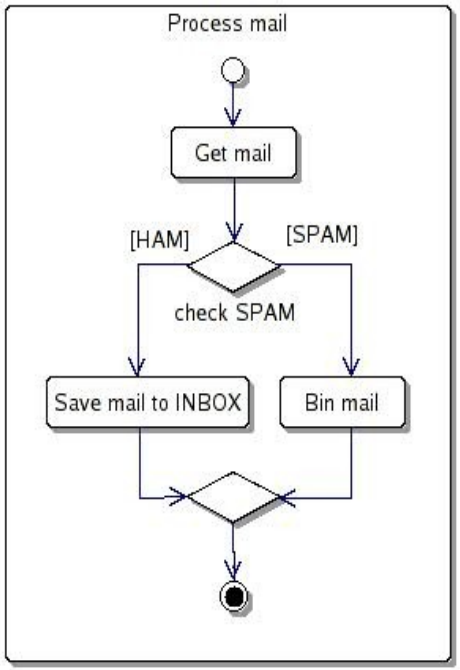


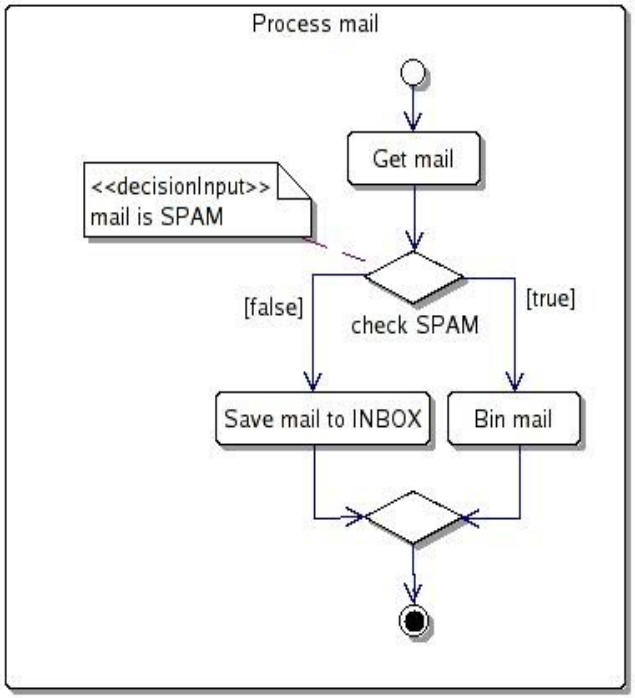
- => Implicitně se AD chová paralelně, tj. jako s *fork/join:*

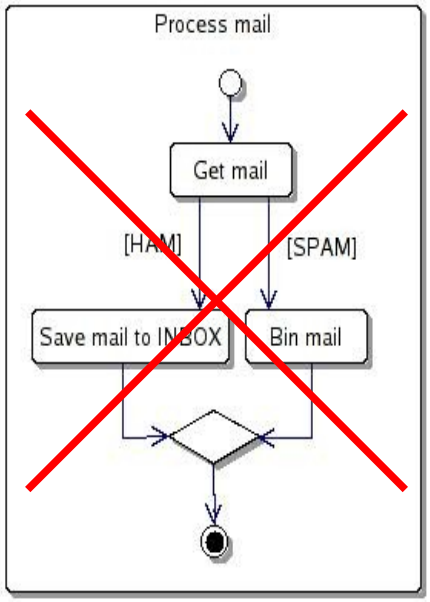


AD: Rozhodovací a slučovací uzly

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



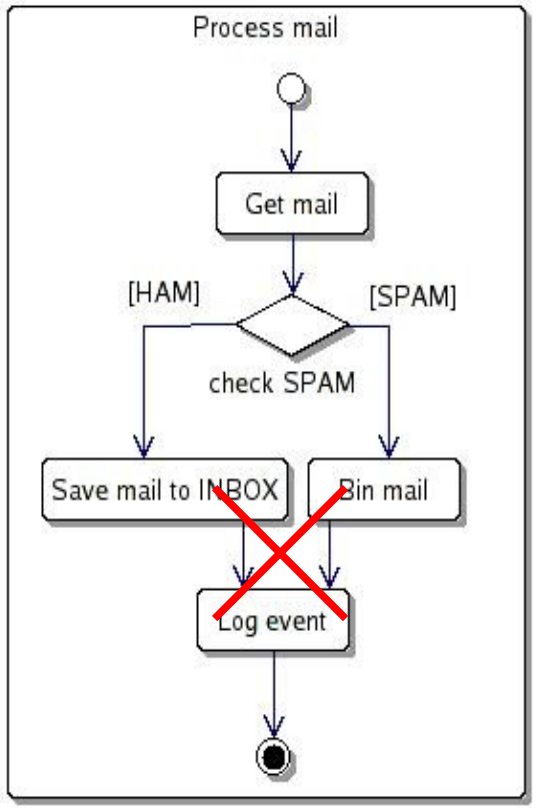




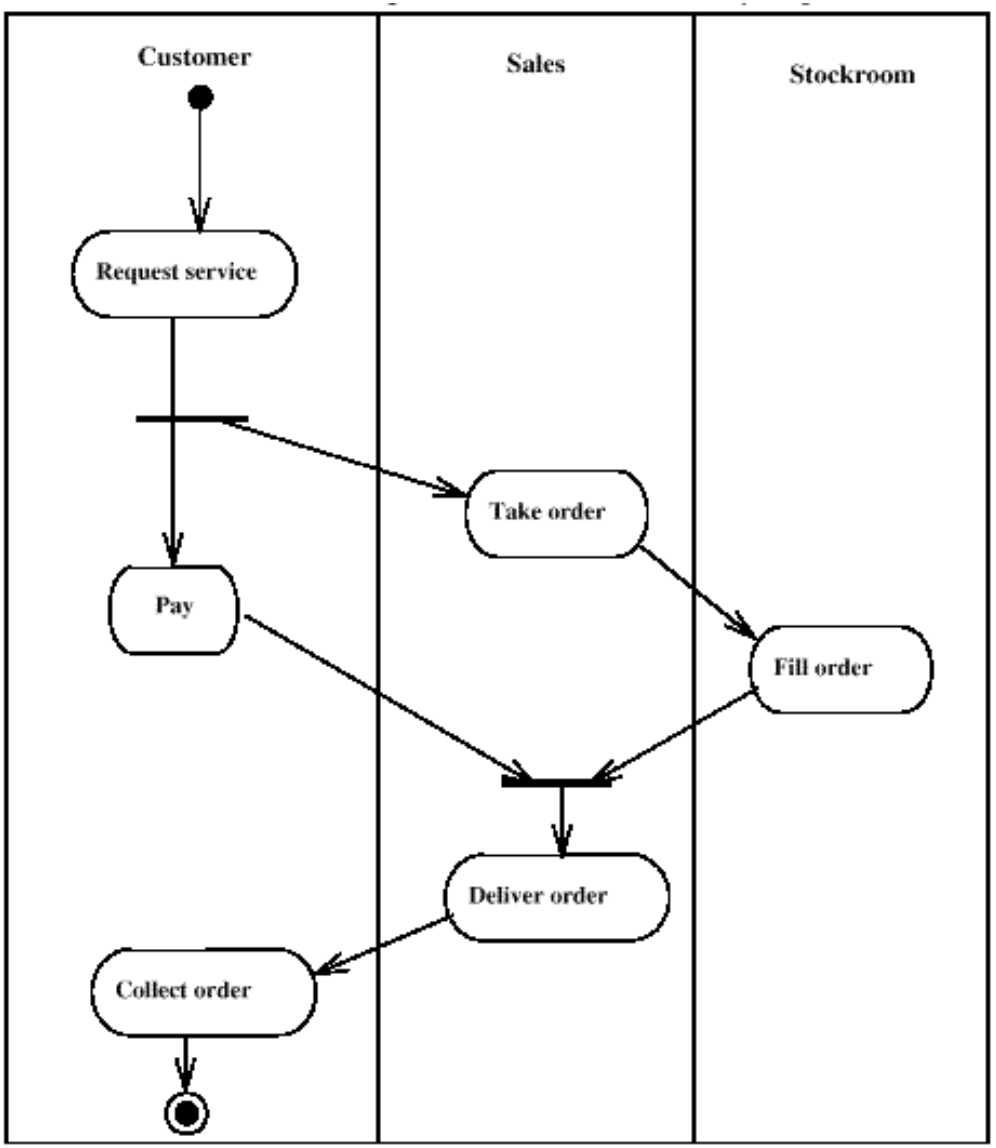
nedoporučuje se

AD: Slučovací uzel

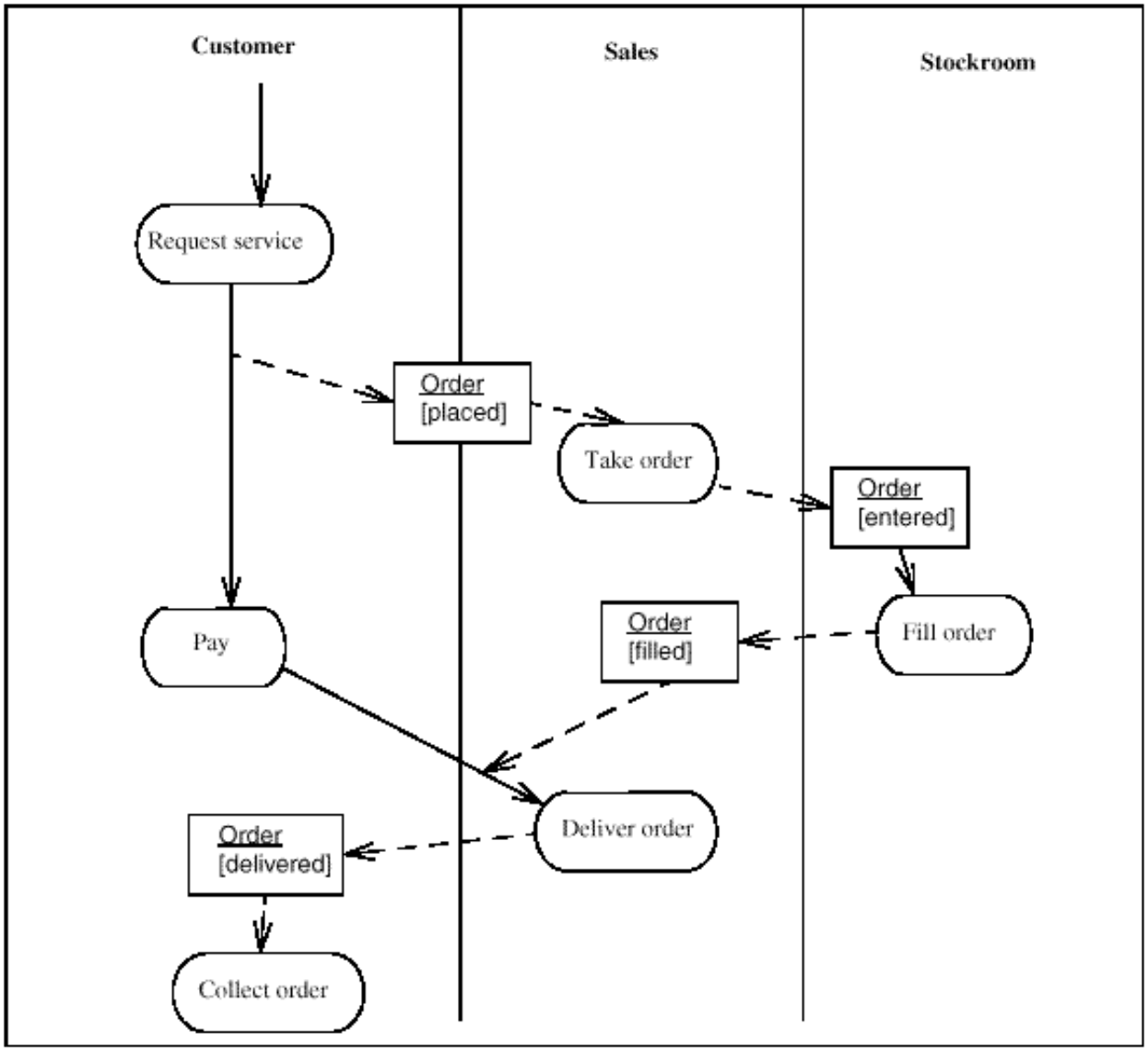
- Smysl použití slučovacího uzlu:
  - akční uzly čekají na všechny vstupy!



AD: segmenty (plavecké dráhy)



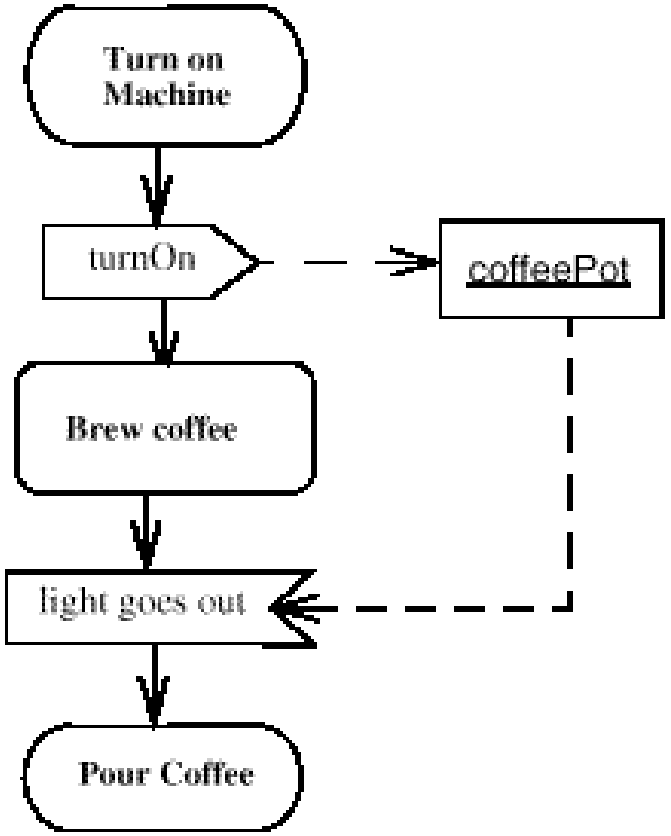
AD: Tok dat


Customer
Sales
Stockroom
Request service
Order
[placed]
Take order
Order
[entered]
Pay
Order
[filled]
Fill order
Deliver order
Order
[delivered]
Collect order

AD: Signály

Příjem signálu a vyslání signálu

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



AD: srovnání se STD

V čem jsou společné:

- Diagramy aktivit jsou zvláštní případ stavových diagramů, kde jsou stavy vyjádřeny jako akce a přechody jsou volány automaticky po ukončení akce

V čem jsou rozdílné:

- Diagramy aktivit se používají v situacích, kdy všechny nebo většina událostí reprezentuje dokončení vnitřně generovaných akcí
- Diagramy aktivit se používají pro modelování obchodních procesů, jichž se účastní několik objektů
- Stavové diagramy se používají v situacích, kdy nastávají asynchronní události
- Stavové diagramy se používají k modelování životního cyklu jedné třídy

Diagramy interakcí
(Interaction Diagrams)

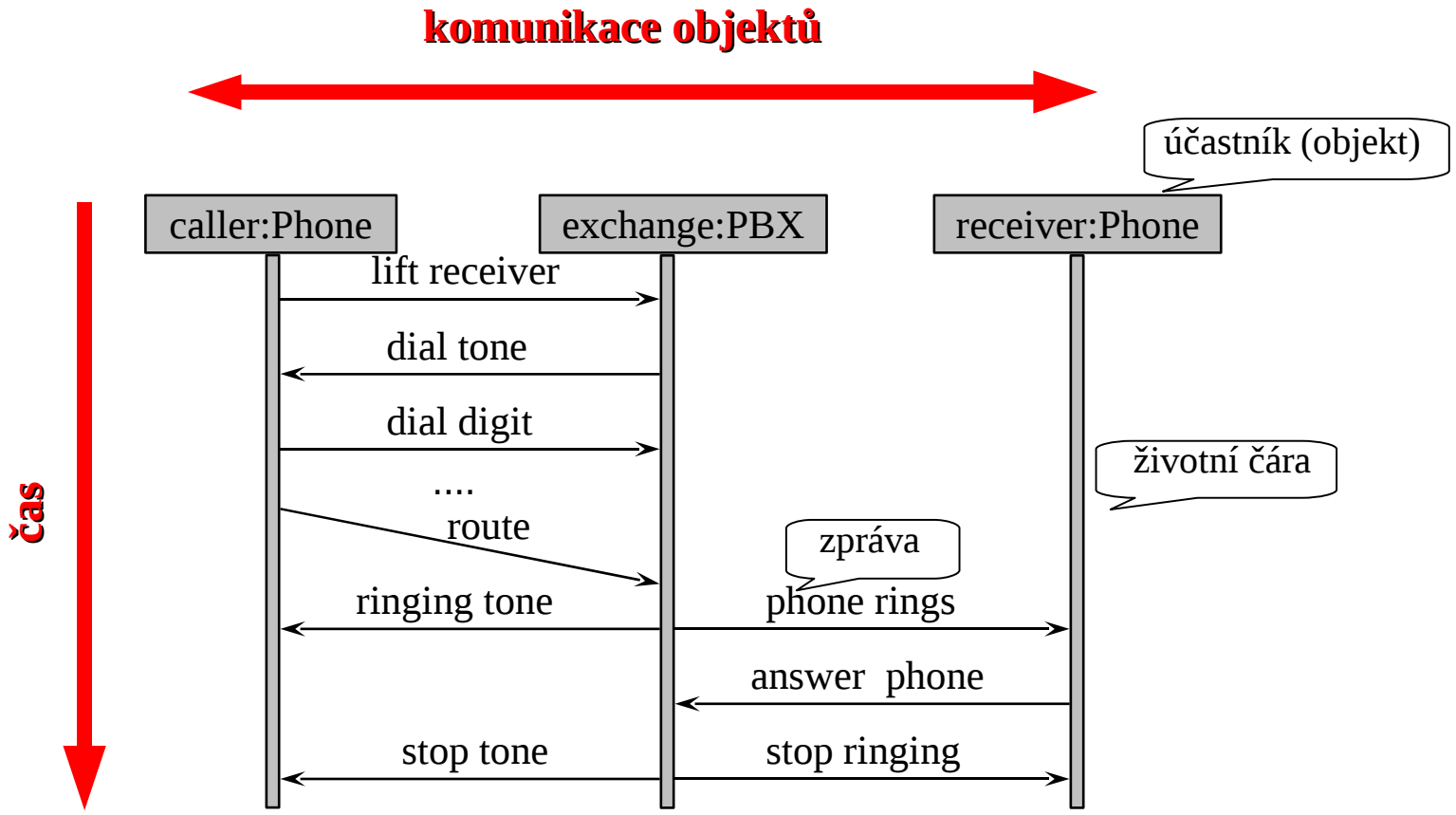
Diagramy interakcí

- Diagramy interakcí:
  - Sekvenční diagram
    - ukazuje interakce uspořádané v časové posloupnosti
    - ukazuje explicitní sekvenci zpráv
    - vhodné pro RT specifikace a pro složitější scénáře
  - Diagram spolupráce
    - ukazuje interakce organizované podle interagujících objektů
    - časová dimenze není znázorněna
  - Diagram časování
    - zaměřené na modelování časových omezení a závislostí
    - novinka od UML 2.0
  - Interaction Overview diagram
    - kombinuje sekvenční diagramy a diagramy aktivit
    - Novinka od UML 2.0

Sekvenční diagram
(Sequence Diagram)

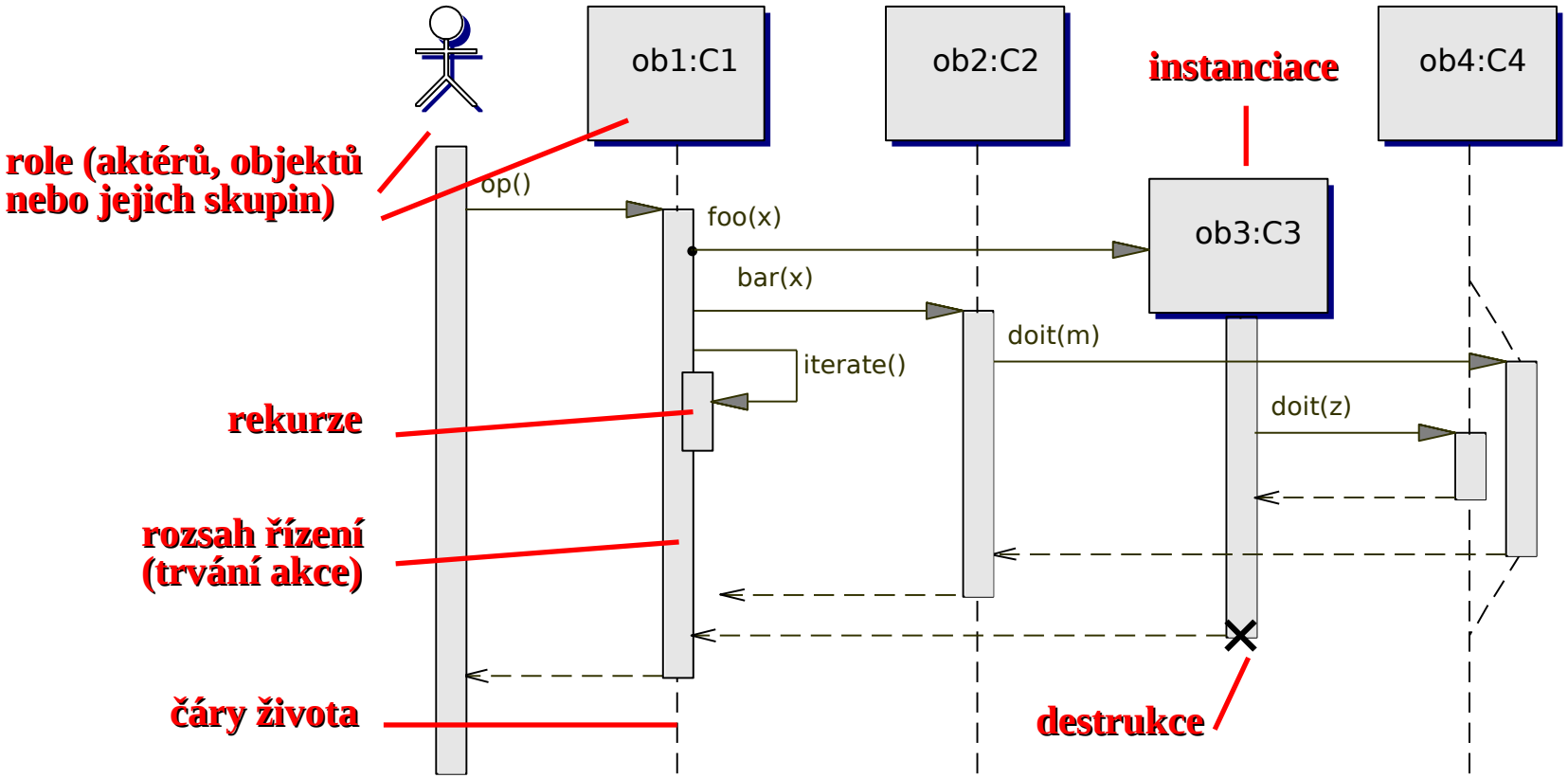
Sekvenční diagram: princip

- Angl: *Sequence Diagram – SD*
- UML 2 přináší dost změn oproti UML 1



SD: notace

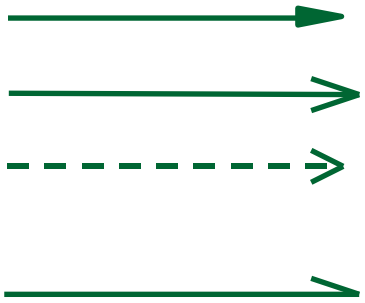
Sekvenční diagram reprezentuje **interakci**, kterou představuje množina **zpráv** vyměněných mezi objekty **během spolupráce** pro splnění požadované operace nebo získání výsledku.



SD: 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):

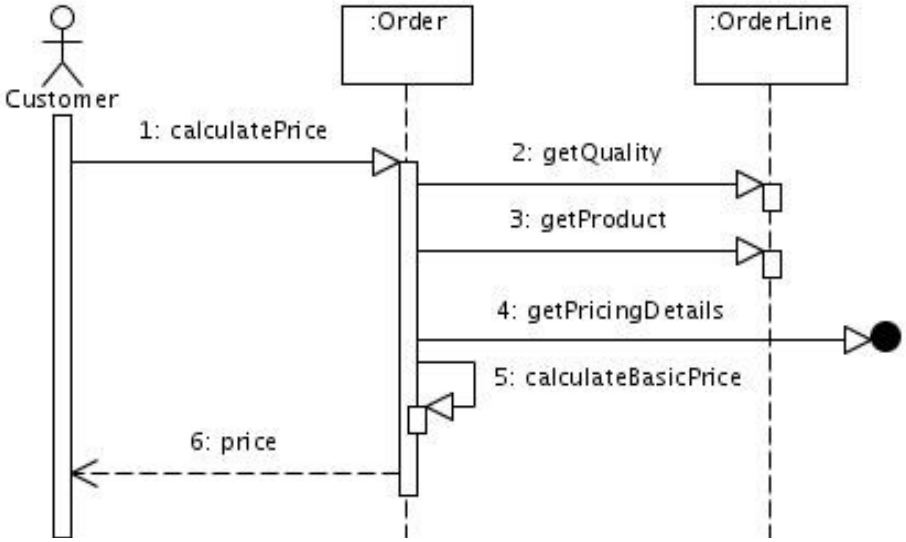


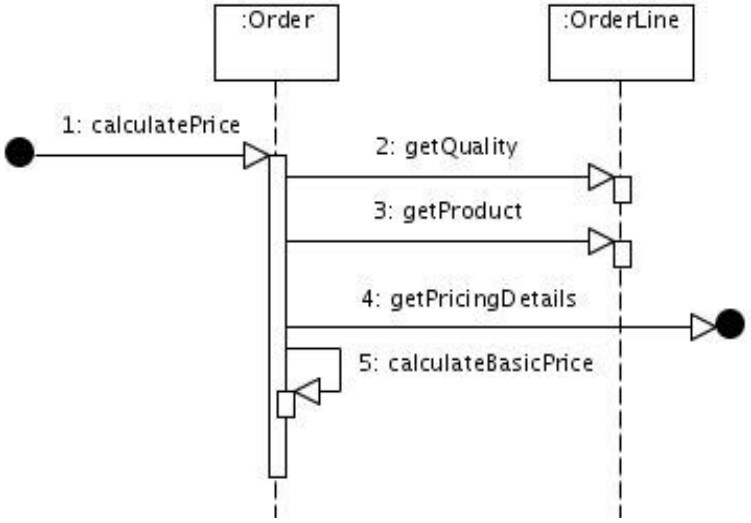
Další symboly: viz knihy o UML

SD: účastníci interakce

SD: provázání interakcí

- Zahájení interakce
  - aktérem z diagramu případů užití
  - „nalezenou zprávou“ *(found message)*
- Odkaz na jinou interakci
  - „ztracenou zprávou“ *(lost message)*

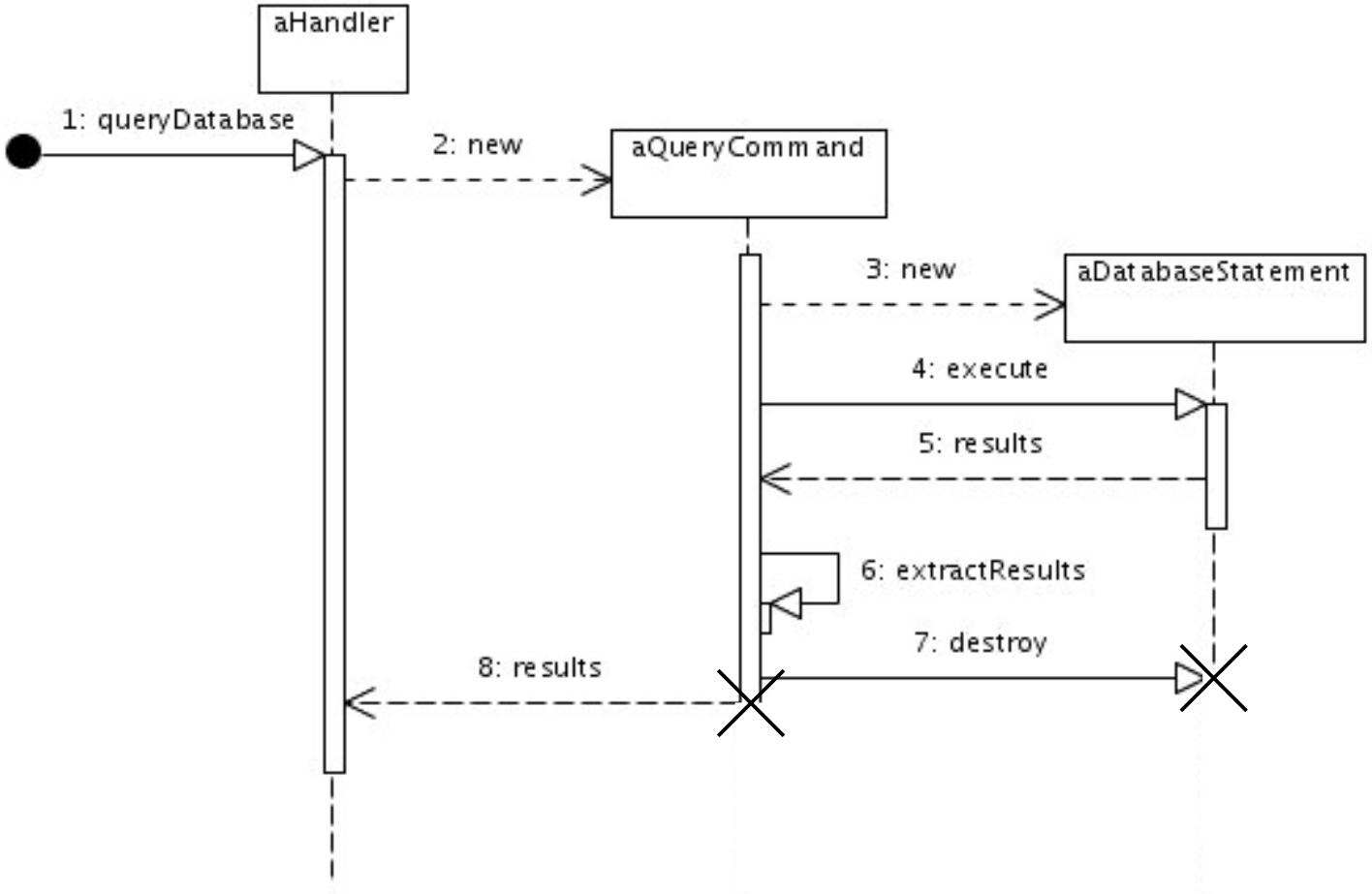




SD: vytváření a rušení účastníků

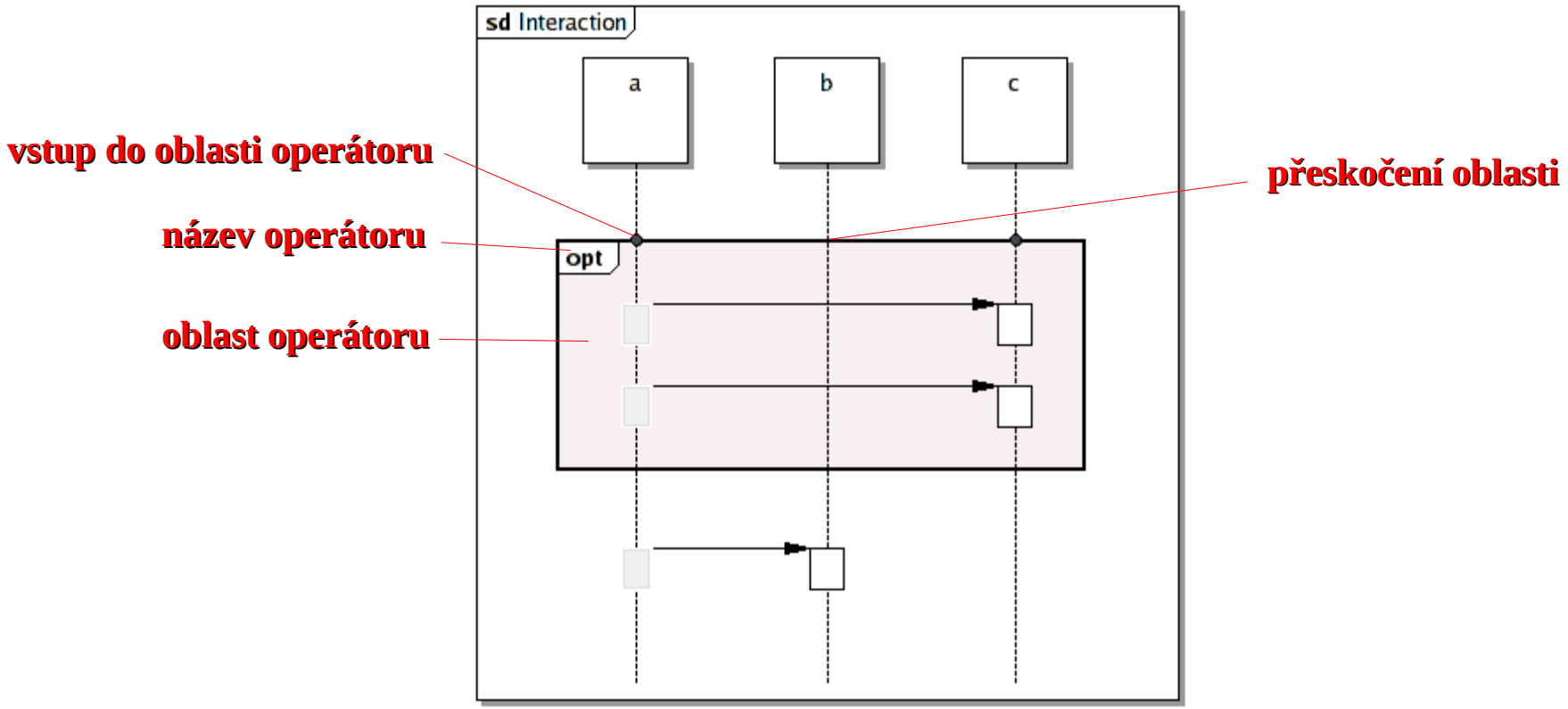
- Účastníci interakce většinou „žijí“ v průběhu celé interakce
  - nestaráme se o jejich vytvoření/rušení
  - jednoduše předpokládáme, že existují
- Explicitní vytvoření účastníka během komunikace
  - => instanciace objektu
  - komunikační šipka vede přímo do objektu, ne do jeho čáry života
- Explicitní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

SD: vytváření a rušení účastníků (II)



SD: Strukturní dělení

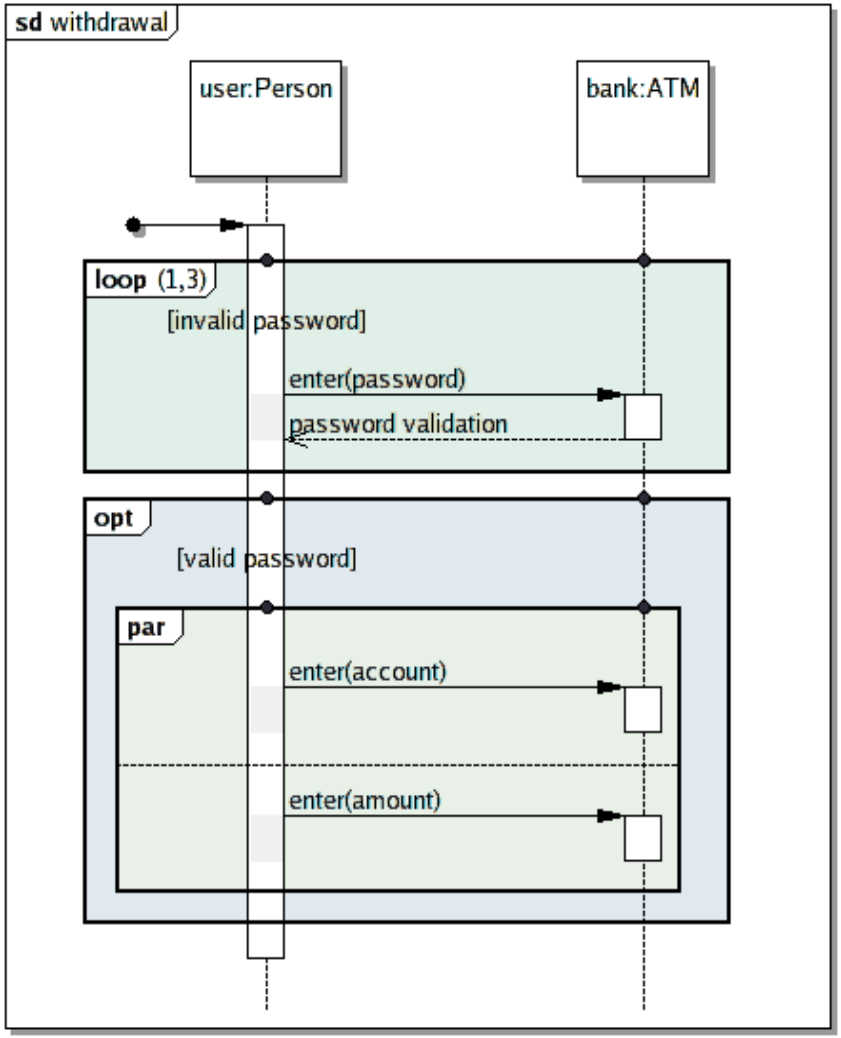
- Znázornění souběžných sekvencí, cyklických sekvencí apod.
  - vyznačíme oblasti a pojmenujeme operátor pro danou oblast
  - operátor se aplikuje na všechny čáry života, které do něj vstupují

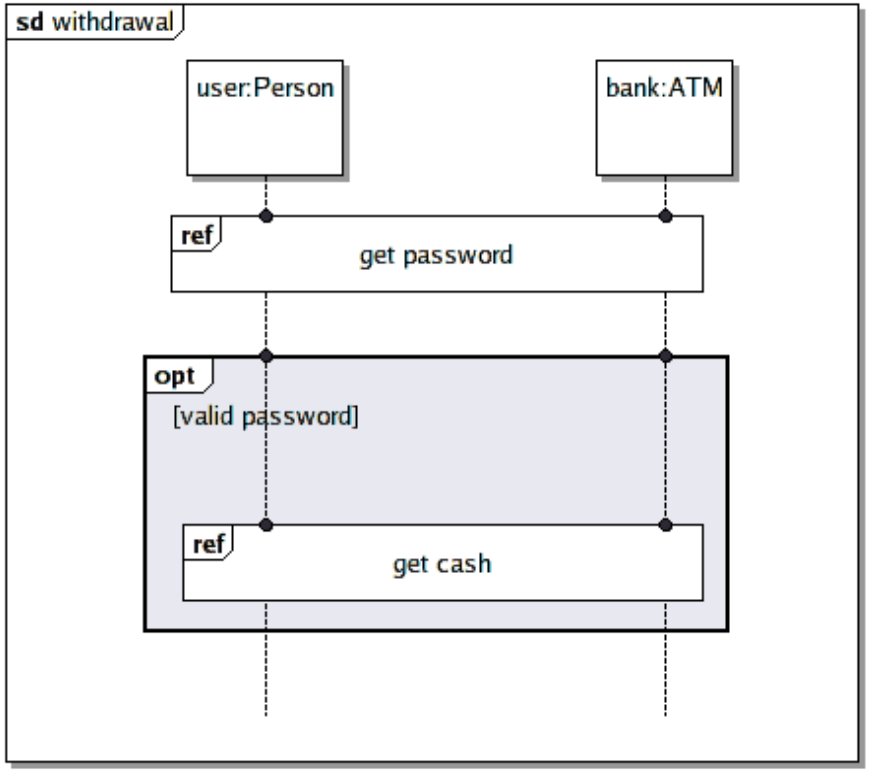


SD: Nejčastější operátory interakcí

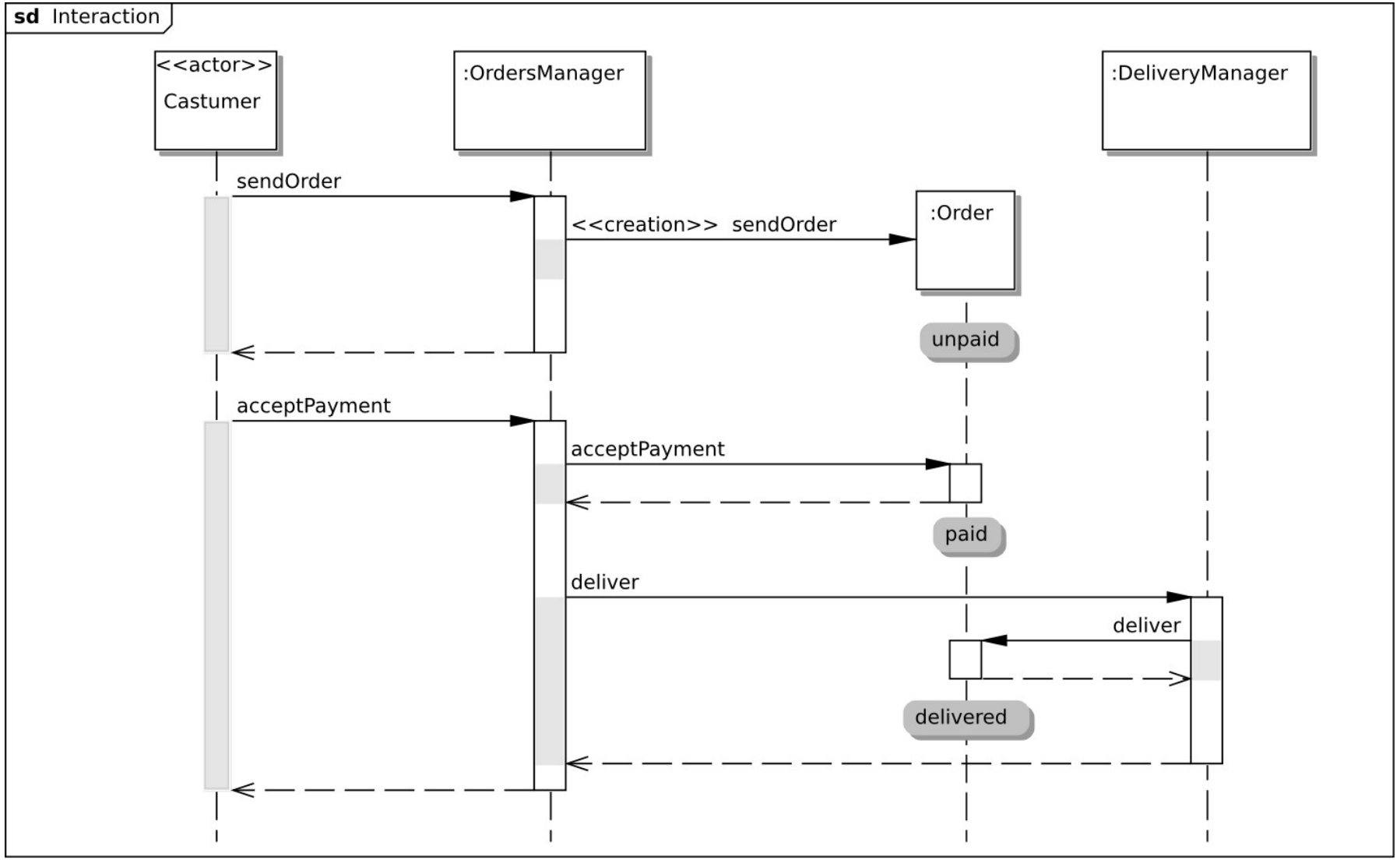
- **podmíněné spuštění** *(optional execution),* **opt**
  - operace uvnitř bloku jsou spuštěny jen pokud je splněna podmínka stráže.
- **volitelné spuštění** *(conditional execution),* **alt**
  - blok je rozdělen na více horizontálních podoblastí reprezentujících jednotlivé větve podmínky; každá podoblast má vlastní stráž
- **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

SD: Strukturní dělení - příklad

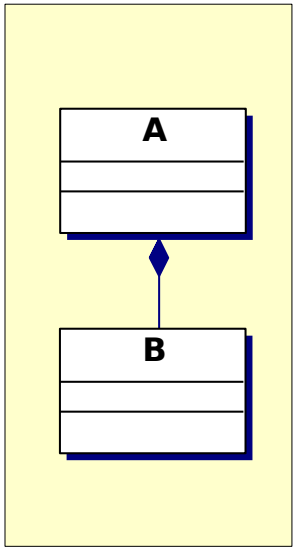


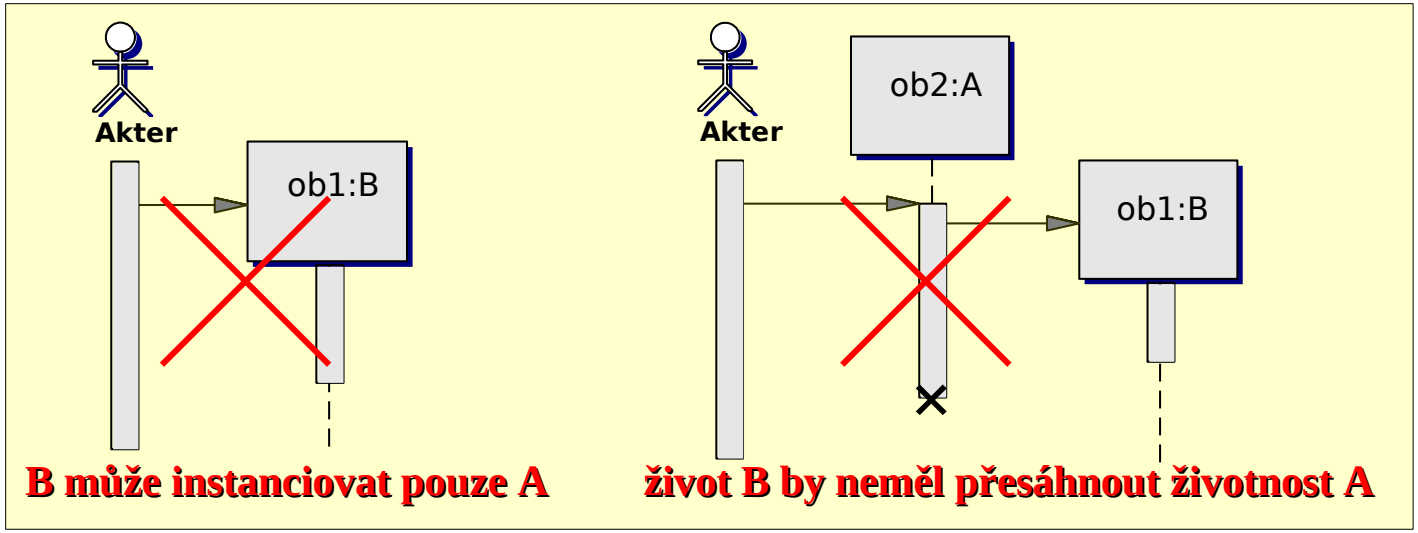


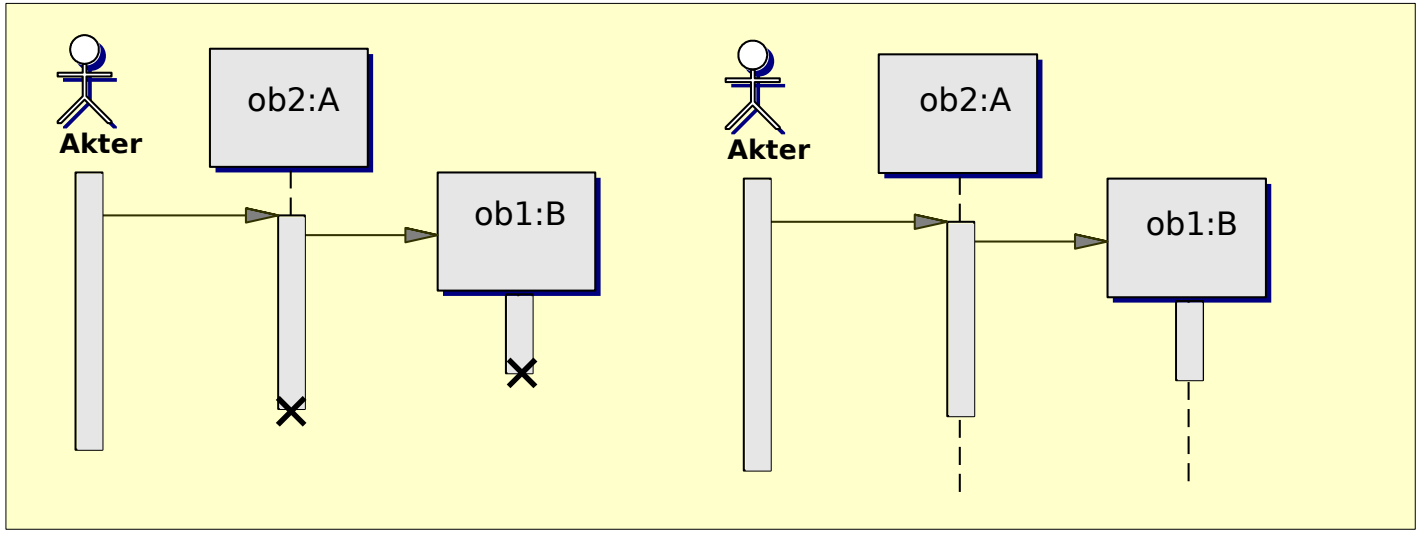
SD: Stav objektu



OSD: kompozice a instanciace

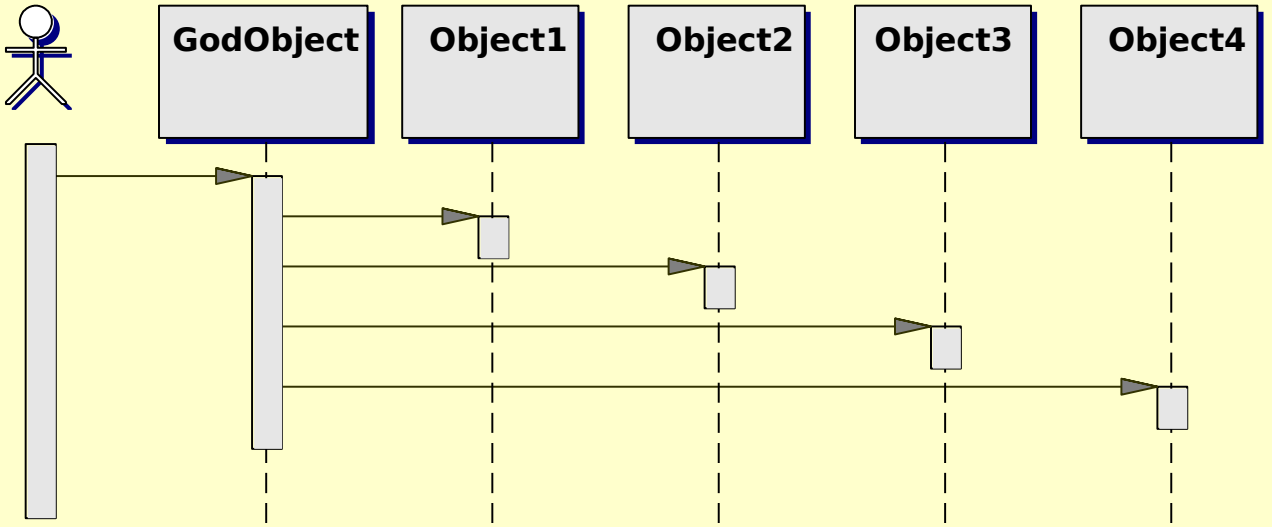




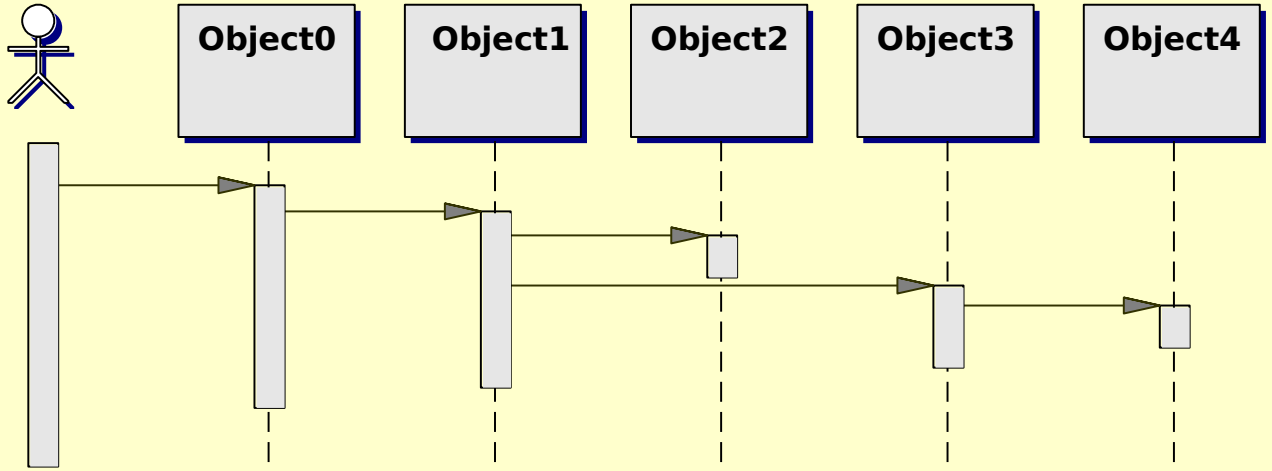


OSD: distribuce “inteligence” objektů

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



Komunikační diagram
(Communication Diagram)

Komunikační Diagramy

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

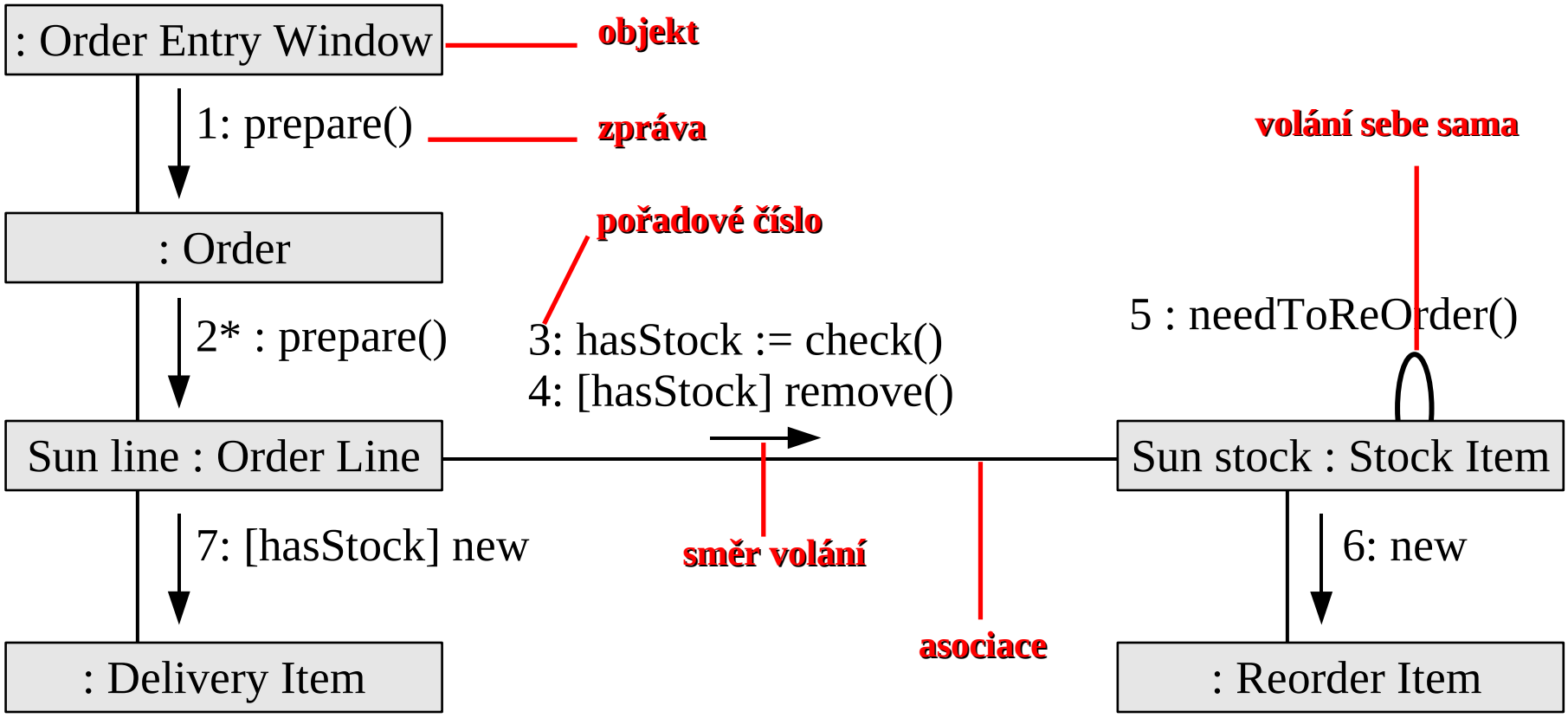
Komunikační diagram ukazuje **interakci** organizovanou podle interagujících objektů, a jejich **propojení** mezi sebou.

- 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

Princip


: 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

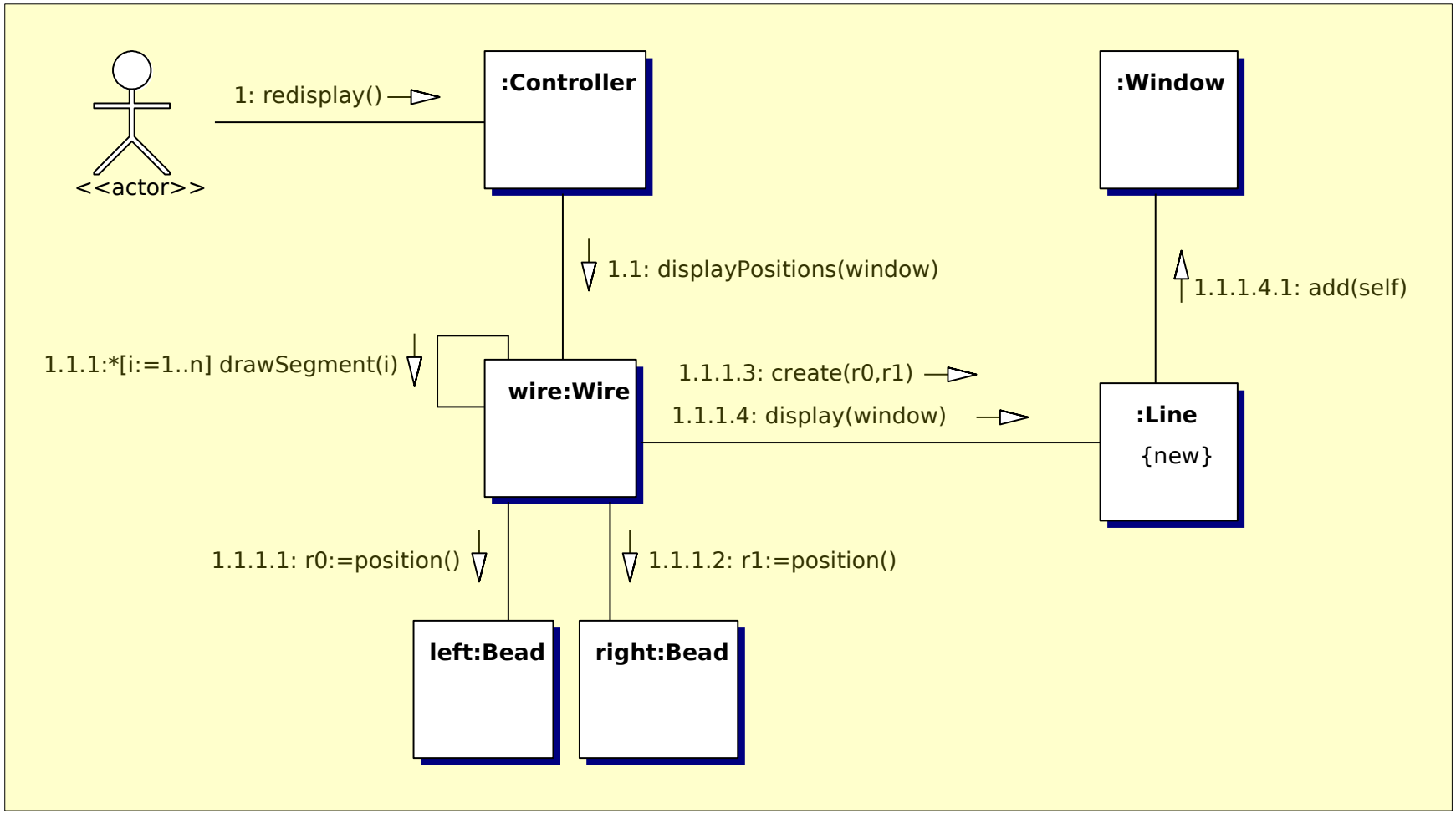

<<actor>>
1: redisplay()
:Controller
:Window
1.1: displayPositions(window)
1.1.1.4.1: add(self)
1.1.1:*[i:=1..n] drawSegment(i)
wire:Wire
1.1.1.3: create(r0,r1)
1.1.1.4: display(window)
:Line
{new}
1.1.1.1: r0:=position()
1.1.1.2: r1:=position()
left:Bead
right:Bead

Diagram časování
(Timing Diagram)

Diagramy časování

- Angl.: *Timing Diagrams*
- Modelují časová omezení mezi změnami stavů různých objektů

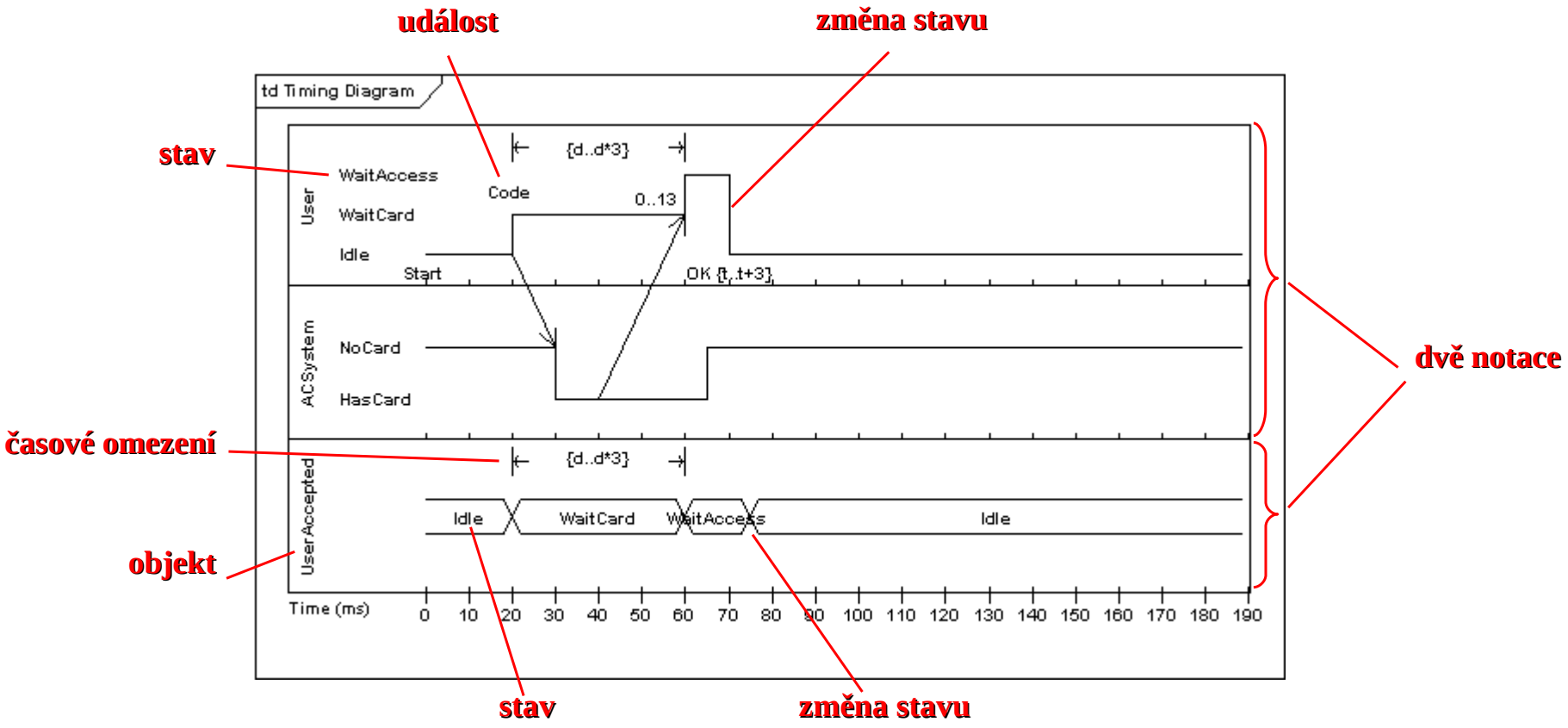
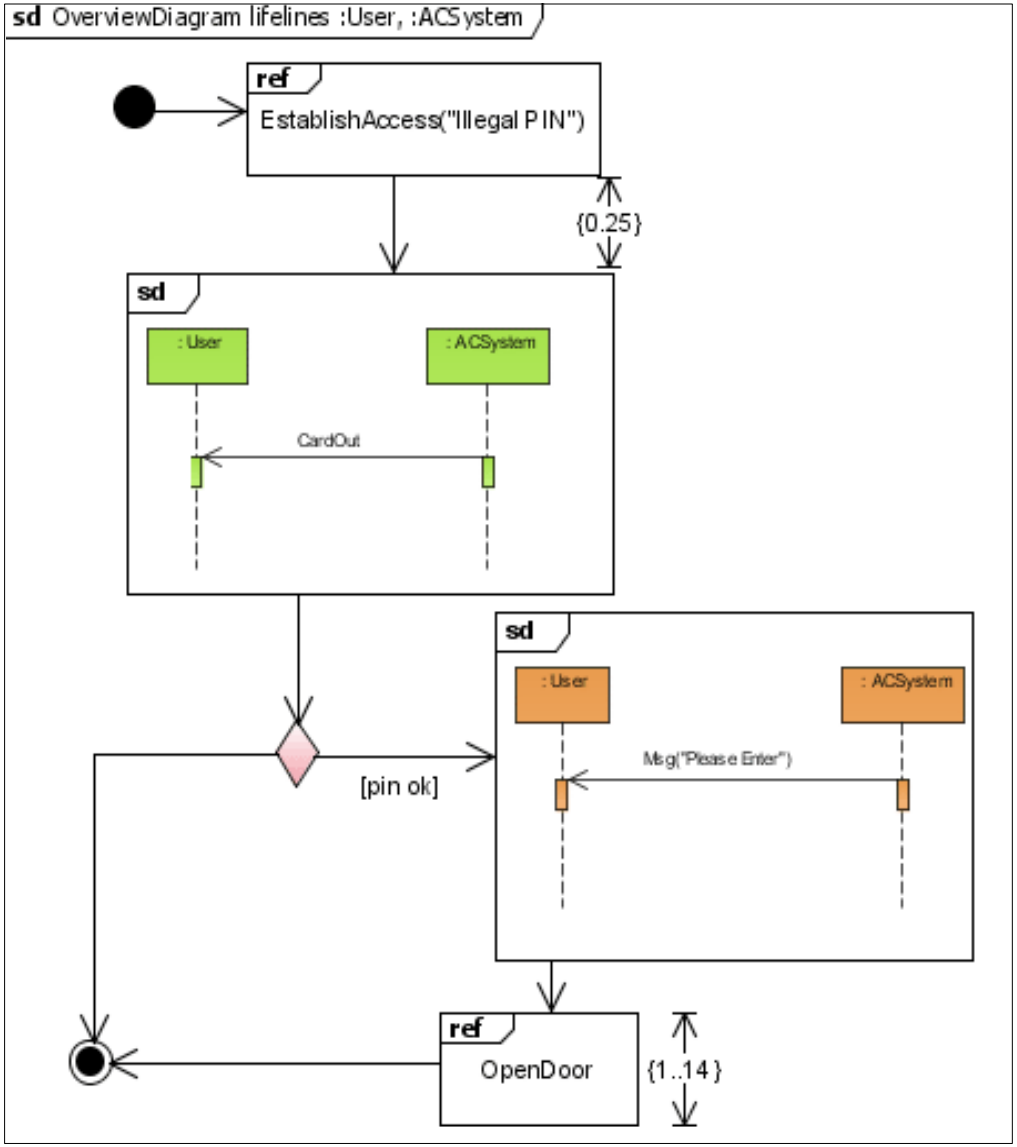


Diagram přehledu interakcí
(Interaction Overview Diagram)

Diagramy přehledu interakcí

- Anlg.: *Interaction Overview Diagram*
- Kombinuje sekvenční diagramy a digramy aktivit



Použití interakčních diagramů

Interakční diagramy jsou používány k mnoha účelům, např.

- dokumentace toku zpráv uvnitř případů užití
- nalezení tříd a operací
- dokumentace opakovaně se objevujících návrhových vzorů
- vysvětlení a předvedení určitých operací
- ukázání všech spolupracovníků specifické třídy
- dokumentace rozhodnutí o architektuře
  - rozdělení do balíků na základě závislostí na ostatních třídách
  - směry asociací
  - ...
- ukázání různých rolí objektu

Interakční diagramy mohou být na různých úrovních granularity

Shrnutí UML diagramů

- Diagram
  - Structure Diagram
    - Class Diagram
    - Component Diagram
    - Composite Structure Diagram
    - Deployment Diagram
    - Object Diagram
    - Package Diagram
  - Behavior Diagram
    - Activity Diagram
    - Use Case Diagram
    - State Machine Diagram
    - Interaction Diagram
      - Sequence Diagram
      - Communication Diagram
      - Interaction Overview Diagram
      - Timing Diagram

Shrnutí UML diagramů (II)

Verze UML	Diagram	Účel
UML 1	Activity	Procedurální a paralelní chování
UML 1	Class	Třídy, vlastnosti a vztahy
UML 1 (d. spolupráce)	Communication	Interakce mezi objekty; zdůrazňuje vazby mezi objekty
UML 1	Component	Struktura a propojení komponent
UML 2	Composite structure	Běhová (runtime) dekompozice tříd
UML 1	Deployment	Rozmístění artefaktů na uzly
UML 2	Interaction overview	Kombinace sekvenční a aktivity diagramů
UML 1 (neoficiálně)	Object	Příklad sestavy instancí tříd
UML 1 (neoficiálně)	Package	Hierarchická struktura pro kompilaci
UML 1	Sequence	Interakce mezi objekty; zdůrazňuje posloupnost zpráv
UML 1	State machine	Jak události mění objekt v čase
UML 2	Timing	Interakce mezi objekty; zdůrazňuje časová omezení
UML 1	Use case	Jak uživatelé interagují se systémem

Shrnutí a hodnocení UML

- UML je efektivní při modelování velkých a složitých systémů
- Základy se naučí běžní vývojáři snadno, poskytuje také pokročilé rysy pro expertní analytiky, návrháře a architekty
- Může specifikovat systém implementačně nezávislým způsobem
- 10-20% modelovacích konstrukcí se používá 80-90% doby
- Strukturální modelování specifikuje kostru, kterou lze zpřesňovat a rozšiřovat dodatečnými strukturami a chováním
- Modelování případů užití specifikuje funkční požadavky na systém objektově-orientovaným způsobem
- Modelování chování zachycuje interakce mezi objekty, stavové přechody mezi objekty a zvýrazňuje tok řízení.