UML

Stavový diagram

Úvod

- Stavový diagram patří mezi klasické a osvědčené nástroje objektového modelování. V UML je tento prostředek přítomen v lehce modifikované podobě Harelových diagramů. Diagram stavů a přechodů, jak je někdy tento prostředek nazýván, slouží pro modelování životního cyklu části systému svým rozsahem odpovídající jednomu objektu.

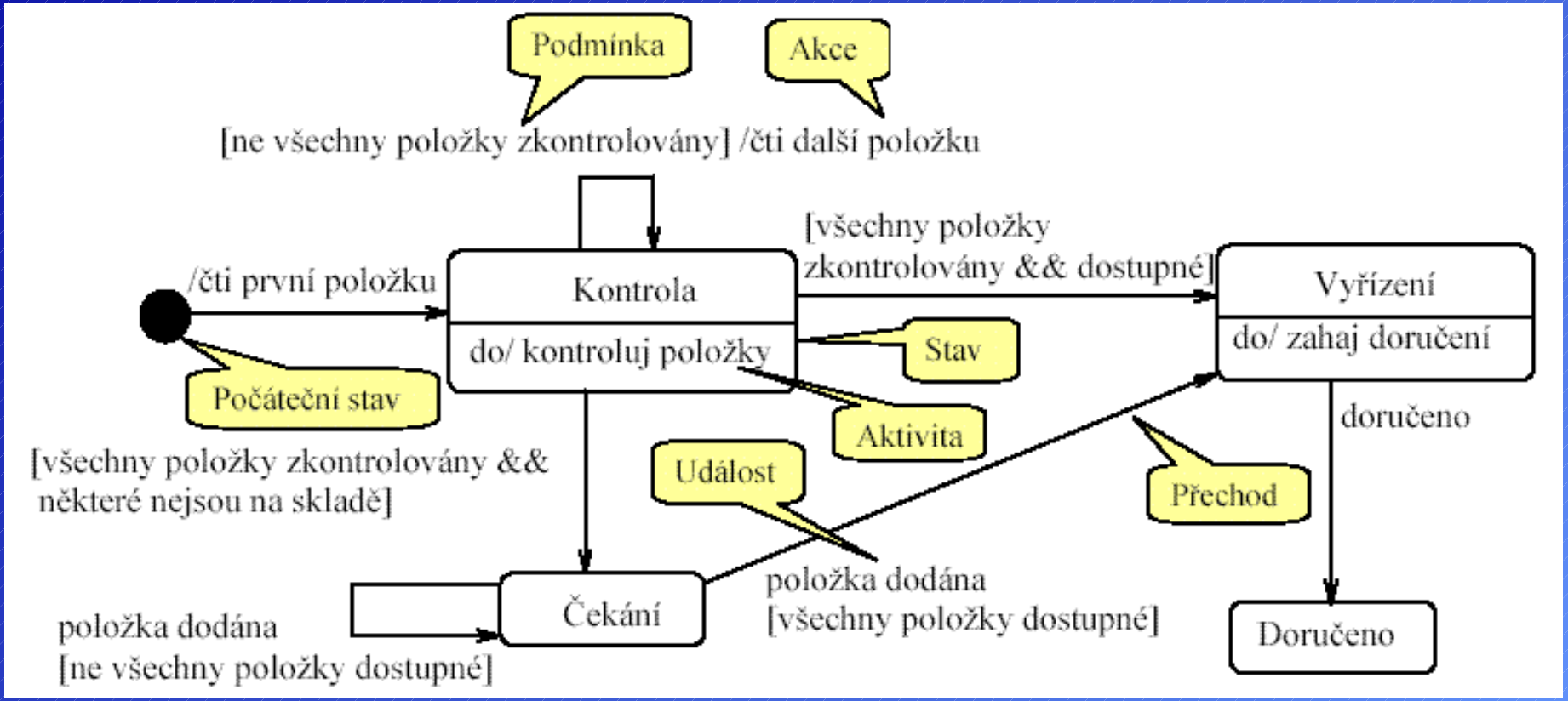
- Tento diagram slouží k zachycení skutečnosti, že objekty procházejí různými stavy. Umožňuje zaznamenat historii objektu, přechody mezi jednotlivými stavy a podmínky těchto přechodů. Podmínky jsou vlastně události, díky nimž přejde objekt z jednoho stavu do druhého. Tento diagram se nepoužívá vždy, ale pouze vyplývá-li to z modelované reality.

Co popisuje?

- Reprezentuje chování objektů, které vykazují nějaké dynamické chování. Ve skutečnosti se používá k modelování chování objektů, ale může sloužit i k modelování chování typu jednání, aktorů, subsystémů, operací nebo metod.

- *Stavový diagram* popisuje *stavy*, v nichž se může objekt nacházet, a jak se stav objektu *změní* jako následek *události*, která na objekt působí. Kreslí se pro jednu třídu a ukazuje chování objektu třídy v průběhu jeho životního cyklu.

Příklad


Podmínka
Akce
[ne všechny položky zkontrolovány] /čti další položku
/čti první položku
Kontrola
do/ kontroluj položky
[všechny položky zkontrolovány && dostupné]
Vyřízení
do/ zahaj doručení
Stav
Počáteční stav
Aktivita
doručeno
[všechny položky zkontrolovány && některé nejsou na skladě]
Událost
Přechod
položka dodána [všechny položky dostupné]
Čekání
položka dodána [ne všechny položky dostupné]
Doručeno

Komponenty diagramu - ***Stav***

- ***Stav -*** je stav v životě objektů nebo interakce, ve které se nachází nějaký stav (např. akce, čekání na událost). Značí se čtvercem se zaoblenými rohy, označeným názvem stavu a pod názvem odděleným čarou lze uvést další vnitřní stavy, kterým může stav nabývat, je-li právě aktivní.

  - příklad: Student: přihlášený, přijatý, zapsaný, student 1.stupně, ...

  - stavu může být přiřazena *aktivita* a *vstupní, výstupní a interní akce* (do/…, entry/…, exit/…, událost/…)

  - počáteční a koncový stav (pseudostavy)

  Aktivita – proces, který trvá „nějakou dobu".
  Akce – proces, který proběhne „rychle" a je nepřerušitelný.

Komponenty diagramu - **Událost**

- ***Událost*** - něco, co se stane v určitém časovém okamžiku, nemá trvání.

  - syntaxe: *jméno_události (seznam parametrů)*

  - příklad: vysokáTeplota (požadovaná teplota)
    - událost po uplynutí času: *after* (např. after (30 minut))
    - událost po splnění podmínky: *when (*např. when (tlak >50) kP))
    - událost ve stavu: *entry*, *exit.*

Komponenty diagramu - *Přechod stavu*

- ***Přechod stavu* -** vztah mezi dvěma stavy, který značí, že objekt v prvním stavu vstoupil do druhého stavu. Změna stavu nastane, jestliže byla splněna požadovaná podmínka. Přechod stavu se značí šipkou z jednoho stavu do druhého a popisem přechodu stavu.

  Přechod mezi jednotlivými stavy bývá vyvolán podmětem z vnějšího okolí, nejčastěji ve formě zprávy zaslané příslušnému objektu nebo jinou externí událostí. Validní stav objektu je zjednodušeně řečeno definován přípustnými hodnotami jeho atributů, přechod mezi stavy je pak vlastně vyjádření změny atributů.

Komponenty diagramu - *Přechod stavu*

Syntaxe ohodnocení přechodu: *Událost [Podmínka]/ Akce*

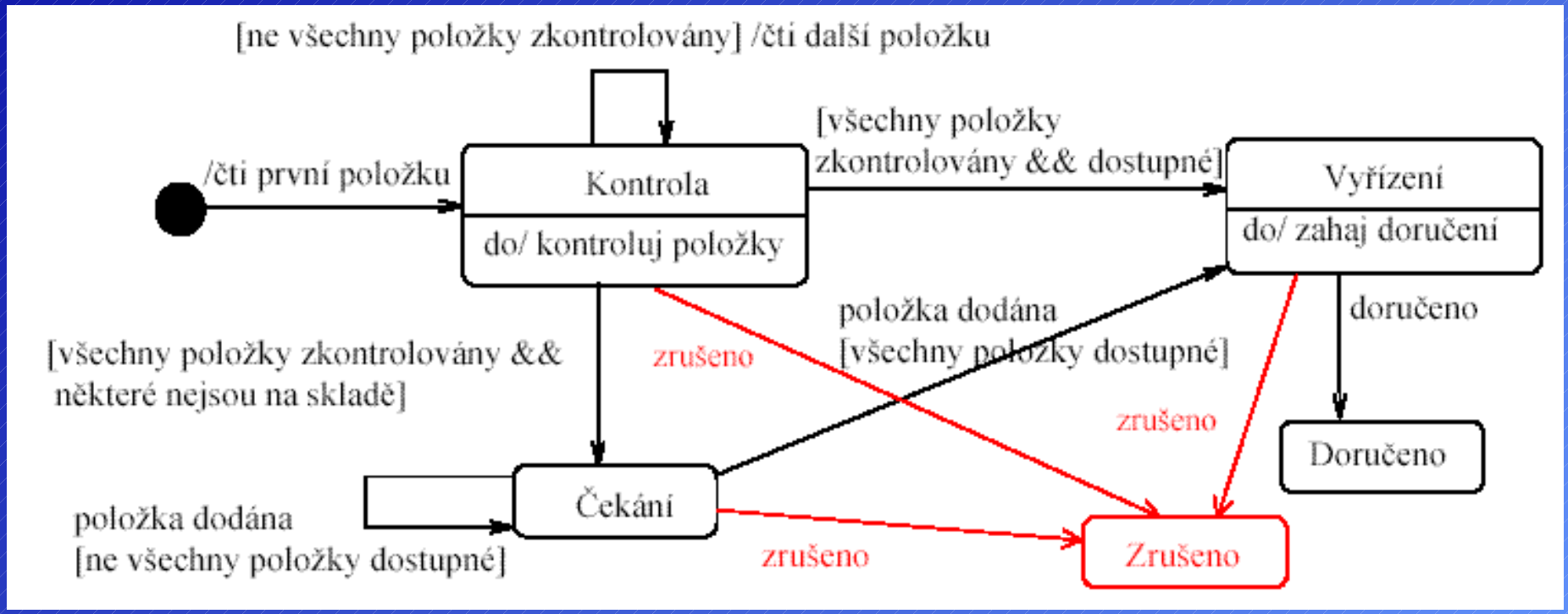
Podmínka (guard) – booleovský výraz platný podmiňující přechod.

- pravidla pro aktivaci přechodu

  - jen jeden přechod ze stavu může být aktivován, není-li událost musí být podmínky výstupních přechodů vzájemně výlučné

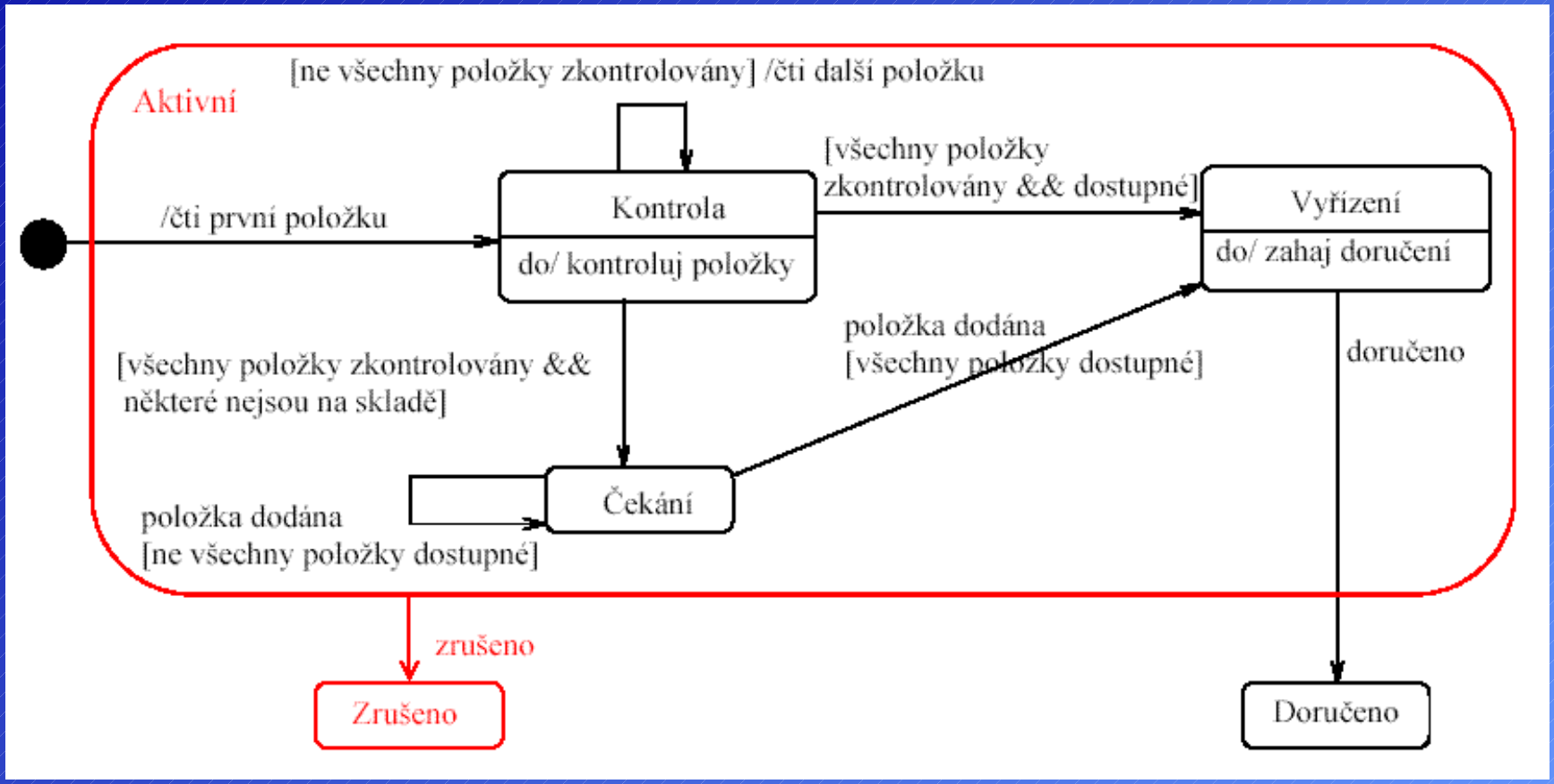
Komponenty diagramu - *Přechod stavu*

- **Příklad: (část 1)**



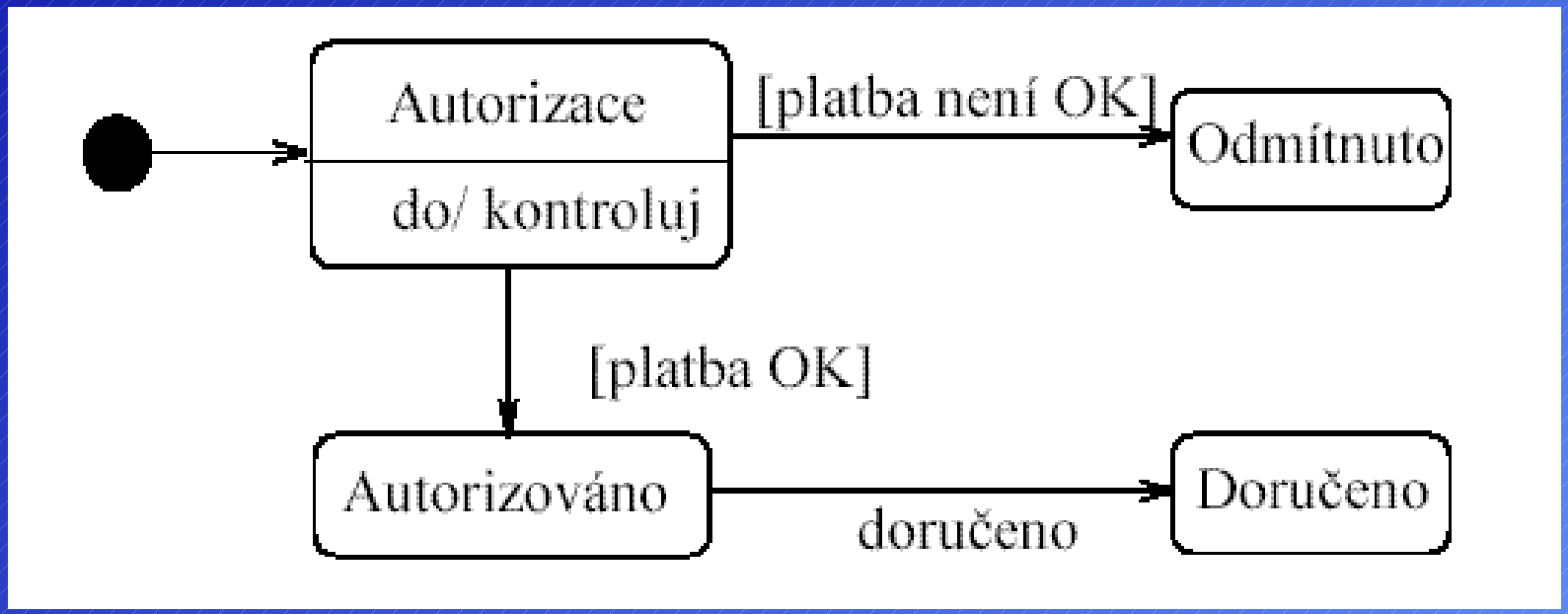
Komponenty diagramu - *Přechod stavu*

- **Příklad: (část 2)**

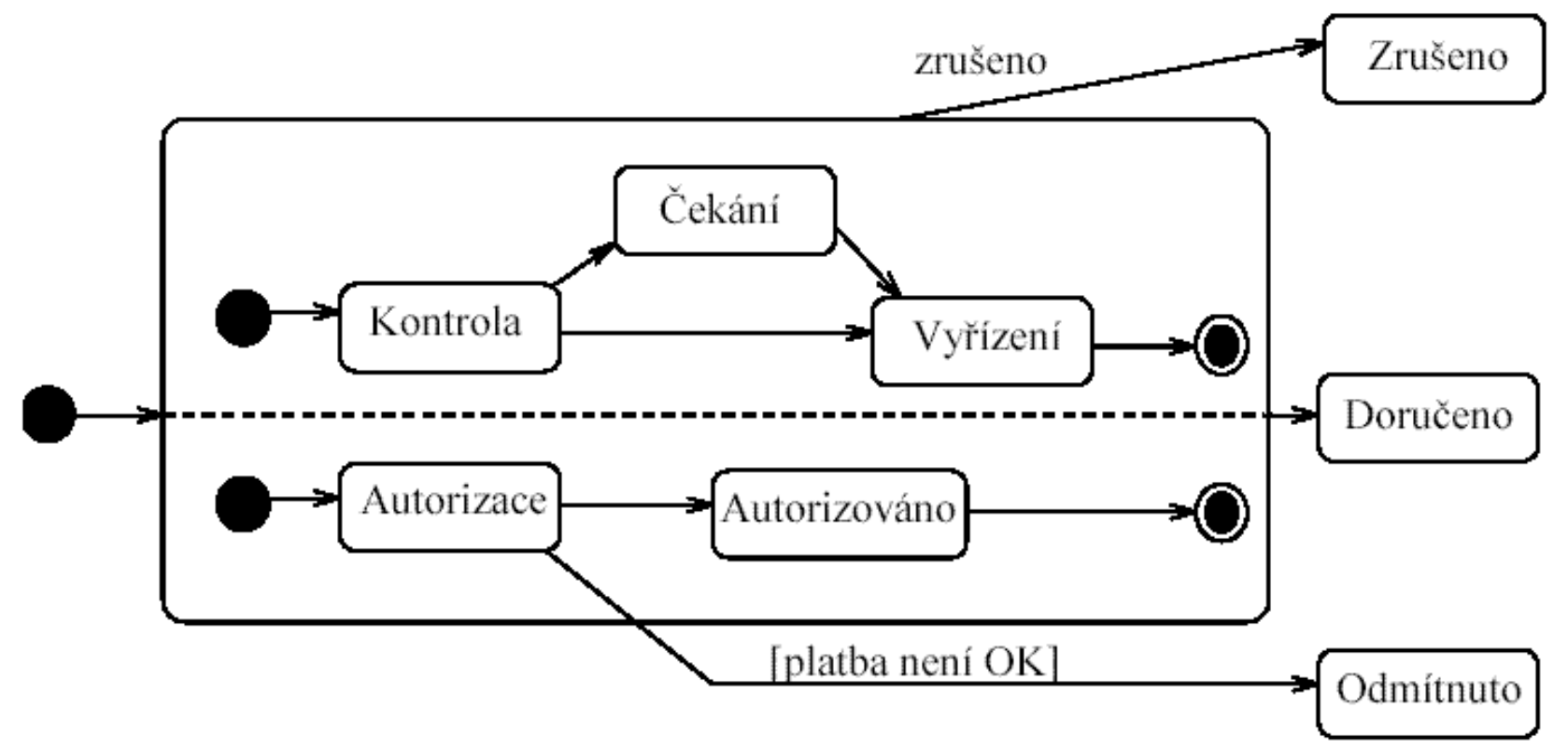


Souběžné stavové diagramy

- složený stav může obsahovat několik souběžných stavových diagramů
- stav je dán kombinací stavů souběžných diagramů



Souběžné stavové diagramy



Konstrukce diagramu

- Pro vytvoření samostatného stavového diagramu začneme zjištěním všech možných stavů, jež může systém dosáhnout. Tyto stavy zakreslíme pomocí uzlů v diagramu. Pak mezi těmito stavy vyhledáme všechna spojení mezi uzly, která reprezentují změny stavů. Nebo druhou možností vytvoření stavového diagramu je začít od počátečního stavu a hledat cesty do dalších stavů (uzlů) až dojdeme ke koncovému(-ým) stavu(ům).

- Na závěr tvorby diagramu stavového stroje je vhodné ověřit, zda je diagram konzistentní:
  - překontrolujeme, jestli jsme nadefinovali všechny stavy, a to nejen za normálních podmínek, ale i při specifických podmínkách. Kontrolu provedeme simulací na diagramu stavového stroje průchodem všemi větvemi a porovnáváme ji s reálnou situací.
  - podíváme se také, zda systém může dosáhnout všech stavů, nechybí-li v diagramu stavového stroje nějaká šipka (změna stavu).
  - a ověříme, zda systém reaguje ve všech stavech na všechny podmínky.

Konstrukce diagramu - *příklad*

Závěr

- Jak je asi každému jasné, životní cyklus objektu nějak začíná a pravidla i nějak končí (v případě, že se nejedná o objekt perzistentní, který je například uchováván v objektové databázi). Ovšem nemusí být již zcela samozřejmé, že tzv. start stav by měl být v diagramu vždy pouze jeden, zatímco stop stavů, tj. stavů ukončujících životní cyklus sledovaného objektu může být více, pochopitelně většinou alespoň jeden. Z implementačního hlediska je start stav ekvivalentní alokaci paměti, resp. zavolání konstruktoru třídy, naopak stop stav je realizován vykonáním destruktoru, v lepším případě prostředky automatické správy paměti.

- Obdobně jako mnohé jiné notace umožňuje UML definovat v těle stavu jednoduchou exekutivu vykonávanou v rámci uvažovaného stavu. V neposlední řadě je možné zachytit důležitá pravidla hlídající konzistenci a validitu celého modelu.