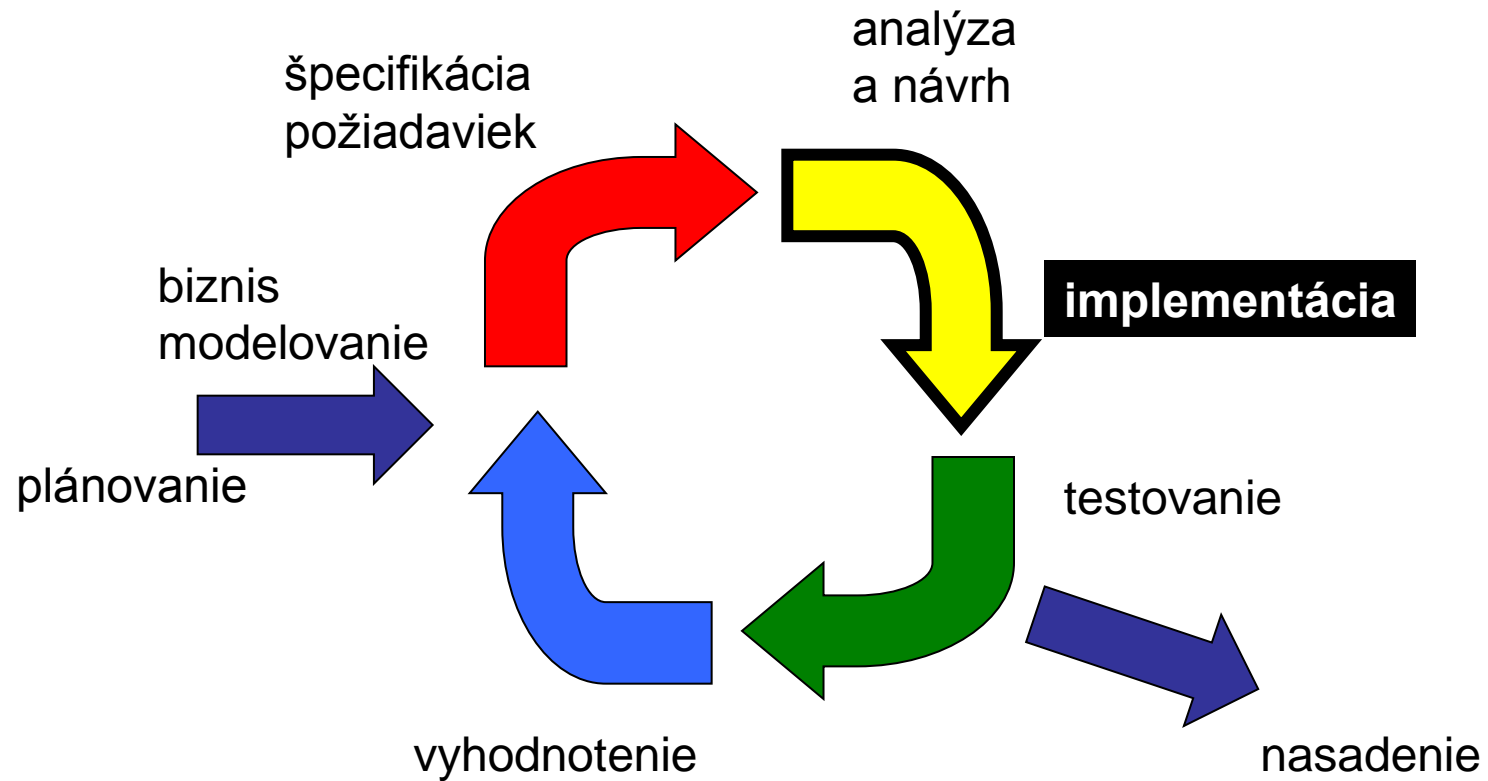


8

Implementácia,
testovanie, nasadenie

RUP – iterácie



Cieľ implementácie

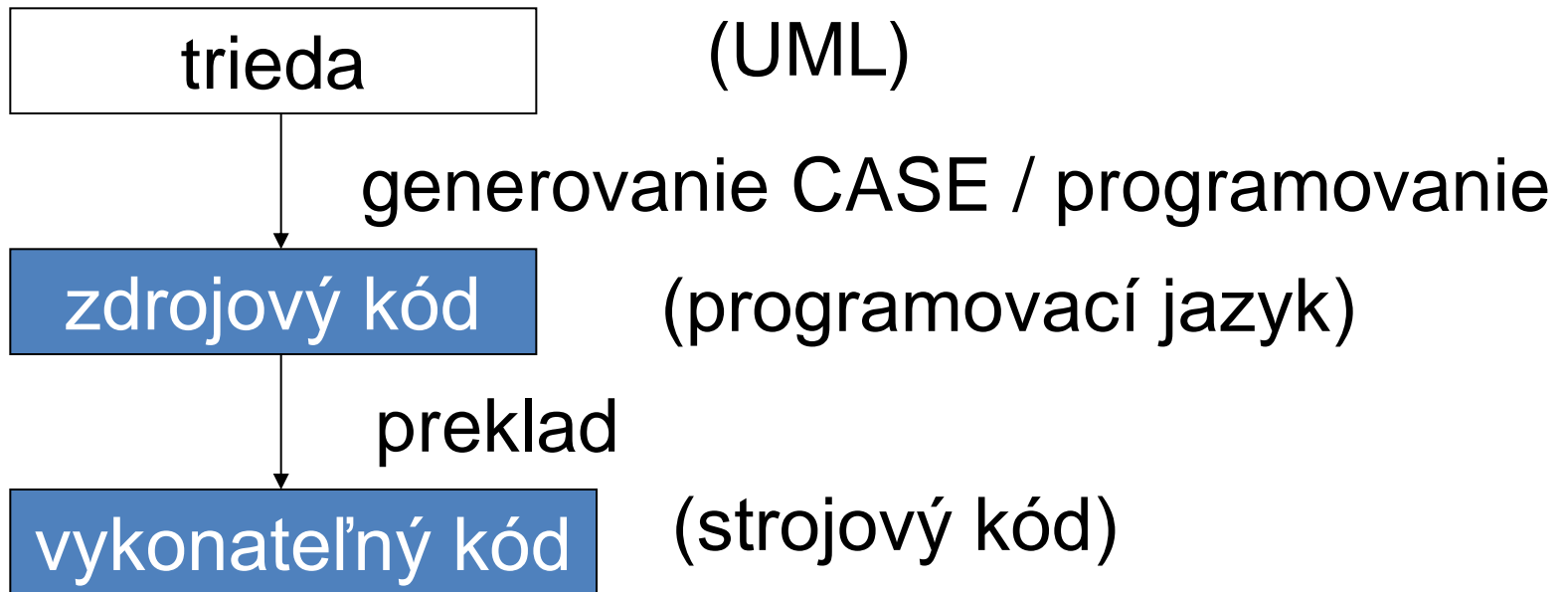
- vytvoriť fungujúci softvér
- model implementácie tvoria softvérové komponenty
 - zdrojový kód prvkov z modelu návrhu
 - vykonateľný (binárny kód) – preklad
 - databázové tabuľky
- konkrétny programovací jazyk
- konkrétny databázový systém

Produkty implementácie

- softvérový komponent – súbor
 - zdrojový,
 - binárny
 - vykonateľný
- UML diagram komponentov
 - štruktúra a závislosti medzi komponentmi
- UML diagram nasadenia
 - spresňuje štruktúru „hardvéru“
 - určuje rozmiestnenie softvérových komponentov

Generovanie/kódovanie - preklad

- softvérový komponent – súbor (zdrojový, binárny kód) – jedna (alebo viac tried)



Komponent UML

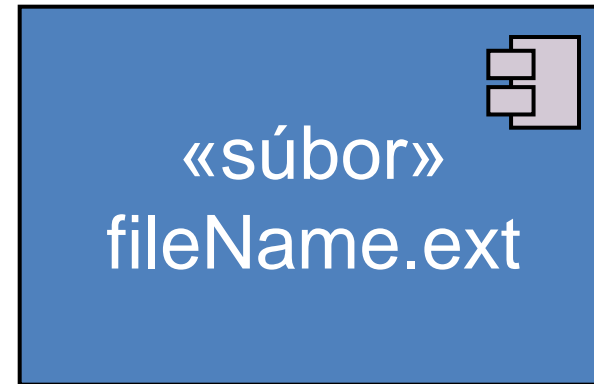


Diagram tried

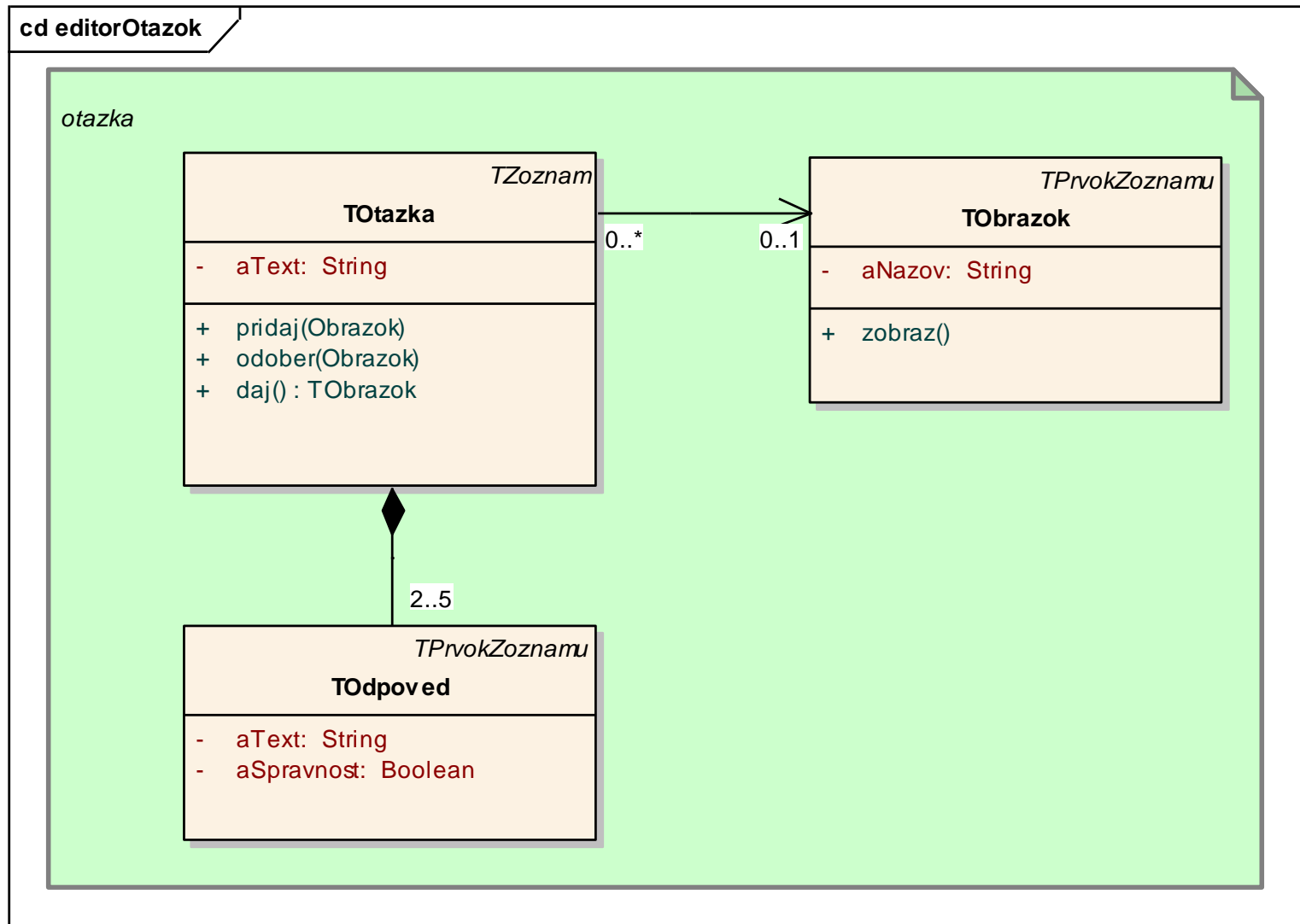
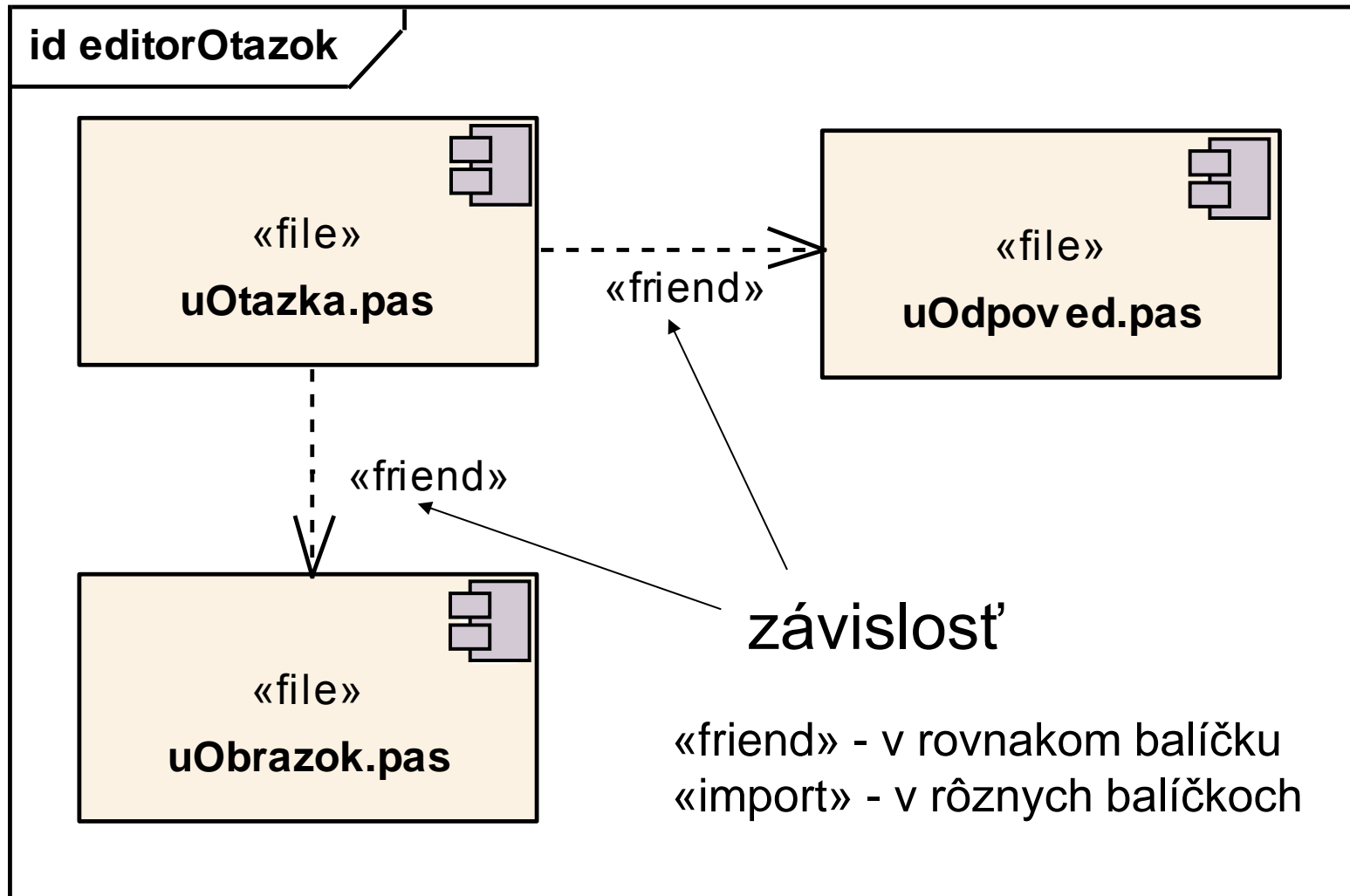


Diagram komponentov



id Component Model

editorOtazok



+ uObrazok.pas



+ uOdpoved.pas



+ uOtazka.pas

CASE: generovanie kódu

<i>UML</i>	<i>Programovací jazyk</i>
Trieda	Class
Rozhranie, rola	Interface
Operácia	Metóda
Atribút	Atribút
Asociácia	Atribút
Závislosť	Lokálna premenná, parameter
Prípad použitia	Postupnosť správ
Balíček	Adresár

Generovanie triedy - Deklarácia

```
TOtazka = class (TZoznam)
```

```
  private
```

```
    atribút aText: String;
```

```
  public
```

atribúty

```
    kompozícia TOdpoved: TOdpoved;
```

```
    asociácia aTObrazok: TObrazok;
```

```
    procedure pridaj(paObrazok: TObrazok);
```

```
    operácia procedure odober(paObrazok: TObrazok);
```

```
    function daj: TObrazok;
```

metódy

```
    constructor Create; overload;
```

```
    destructor Destroy; override;
```

```
end;
```

Generovanie triedy - Implementácia

```
destructor TOtazka.Destroy;  
begin  
  inherited Destroy;  
end;
```

doprogramovať ručne!

```
function TOtazka.daj: TObrazok;  
begin  
  
end;
```

stavový diagram
diagram aktivít
sekvenčný diagram

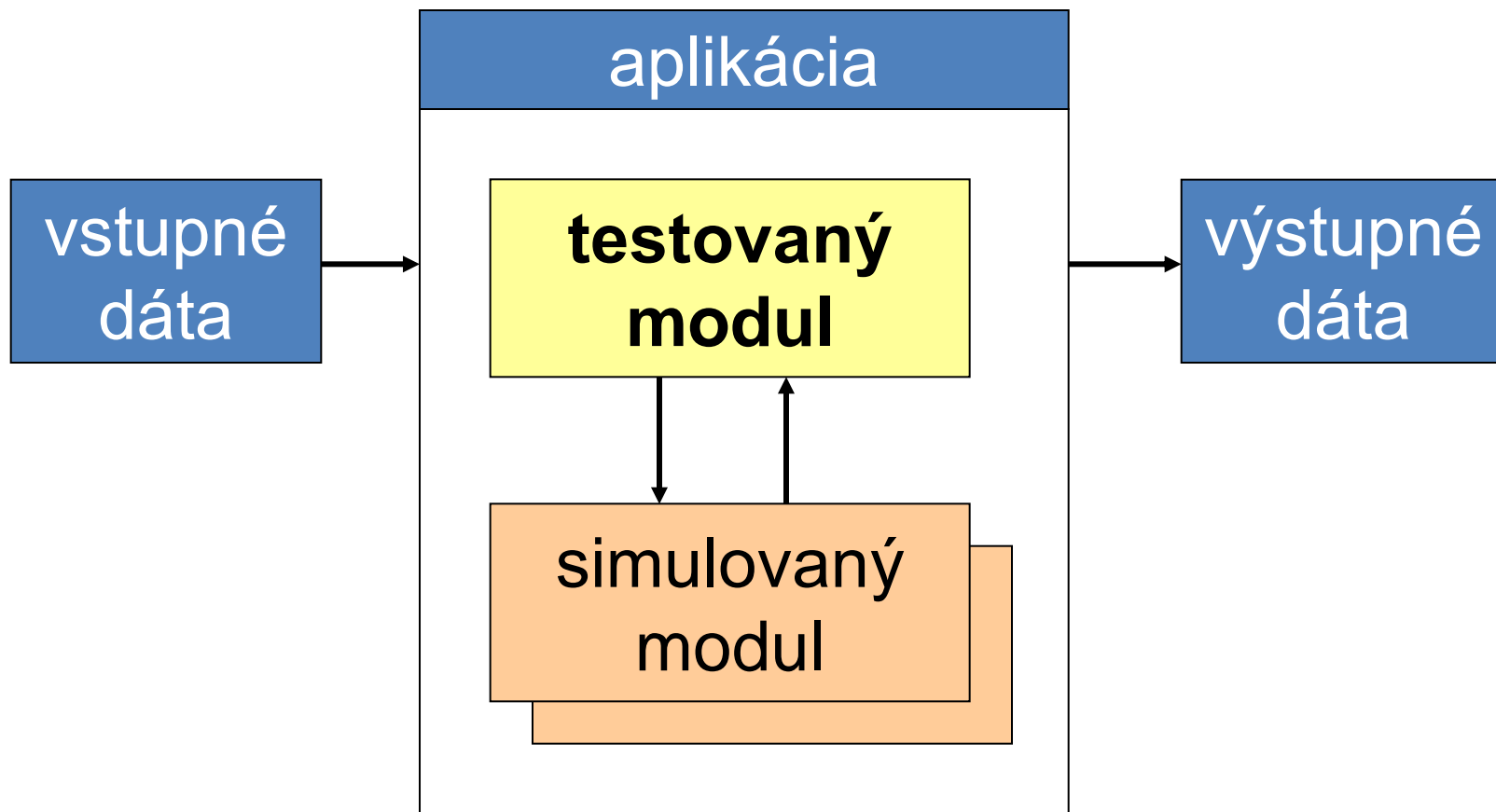
delphi - ukázať

Stratégia implementácie

postup, ako sa realizujú jednotlivé softvérové súčiastky

- zhora nadol
- zdola nahor
 - vplyv stratégie návrhu
 - vplyv na stratégiu testovania
- štýl programovania
- princípy objektového a štruktúrovaného programovania

Implementácia zhora nadol



Implementácia zhora nadol

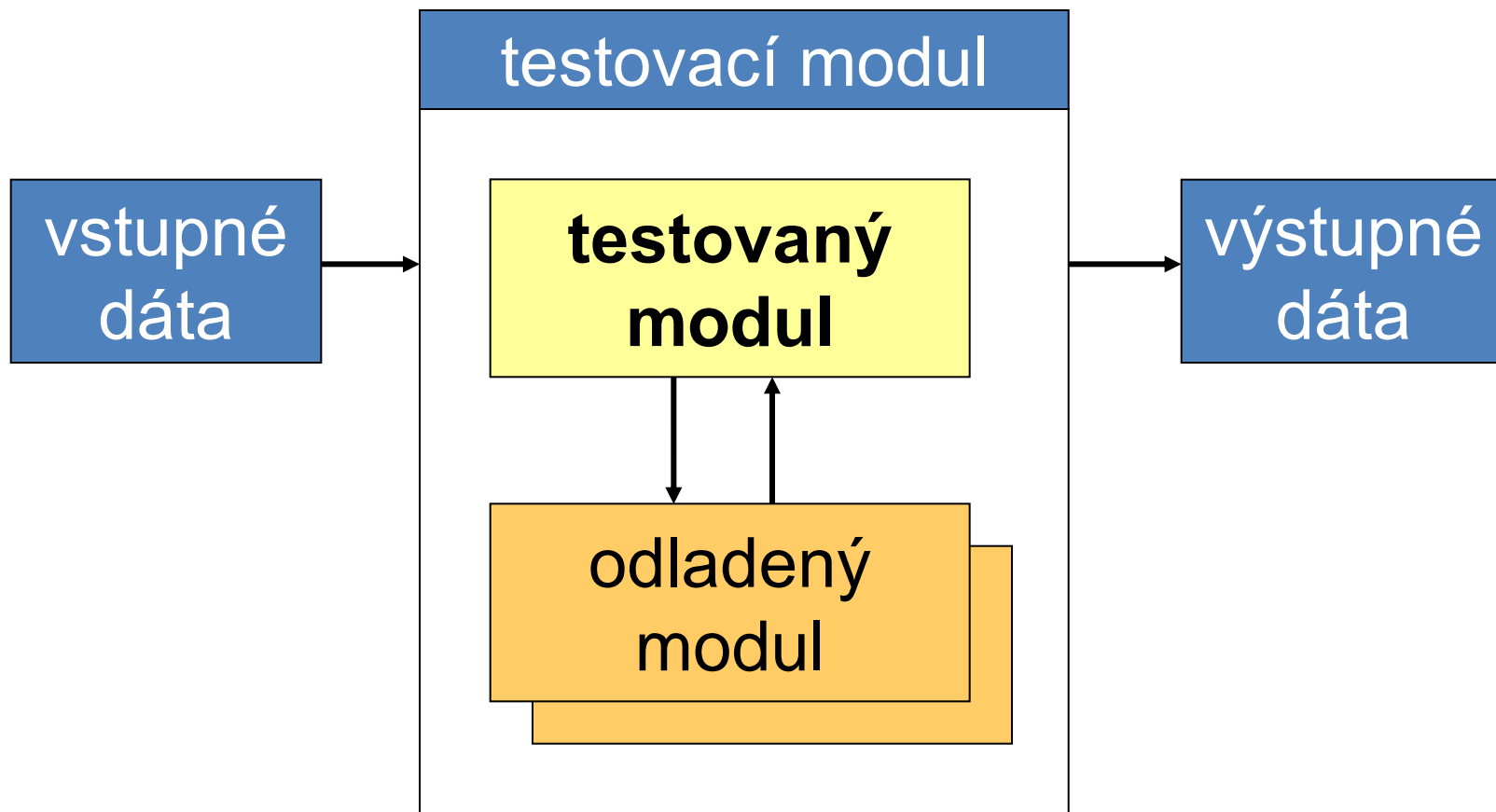
výhody

- skorá prezentácia systému
- skorá identifikácia závažných chýb
- viacnásobné overenie logiky systému

nevýhody

- testovanie modulov je jednoduchšie
- neexistujúce moduly sa simulujú

Implementácia zdola nahor



Implementácia zdola nahor

výhody

- samostatné odladenie modulov

nevýhody

- logika systému sa testuje až nakoniec
- systém sa prezentuje až po dokončení
- tvorba špeciálnych testovacích modulov

Štýl programovania

- mená samo dokumentujúce
- mená konzistentné – rovnaký spôsob tvorby v celom systéme
- ukrývanie informácií
- vnáranie – pozor na priveľkú hĺbku
- vedľajšie efekty minimalizovať
- pretty programming – odsadenie, jeden prikaz – jeden riadok...

Verziónovanie

- Revision control, Source control, Version control
- Software Configuration Managment



Verziónovanie je správa zmien dokumentov, počítačových programov, veľkých webových aplikácií a iných kolekcií informácií.

- Zmena identifikovaná prostredníctvom – čísla revízie
- Nevyhnutné pri viac užívateľských vývojárskych projektoch

Verziónovanie - zmysel

- Pravdepodobne už aj používate
- Nevyhnutné pre veľké, rýchlo sa meniace projekty s mnohými autormi:
 - Zálohovanie a obnova
 - Synchronizácia
 - Krátke UNDO
 - Dlhé UNDO
 - História zmien
 - História používateľa
 - Pieskovisko
 - Vetvenie a spájanie

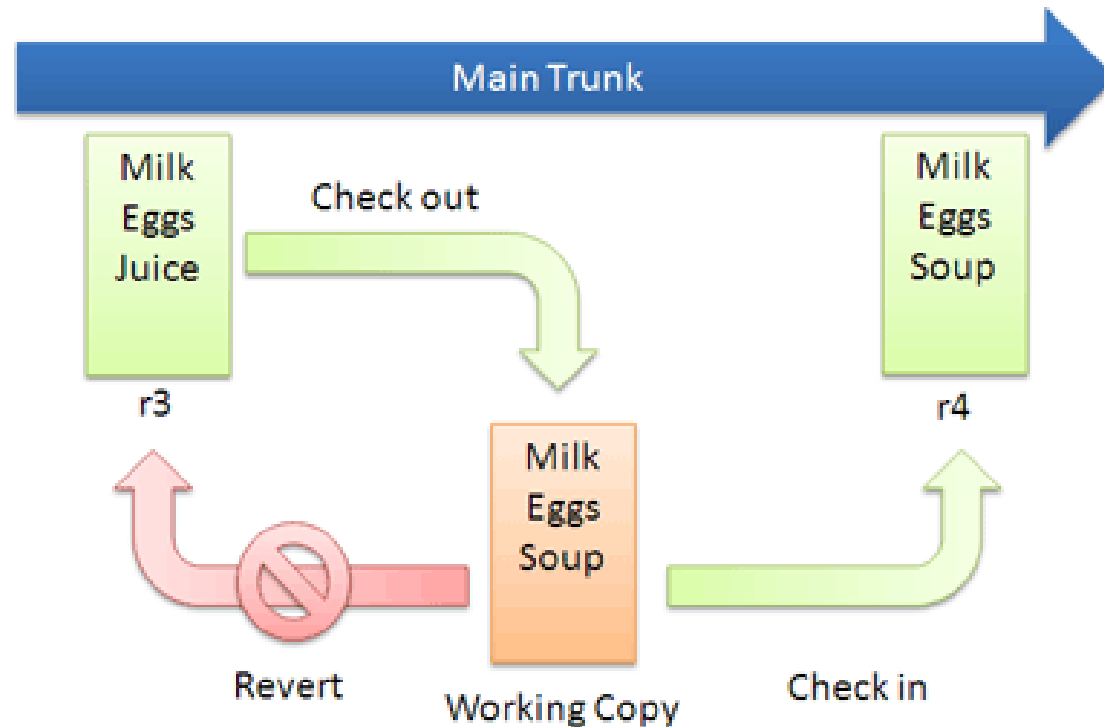
- Repozitár – databáza zmien
- Server - obsahuje repozitár
- Working Set/Working Copy
- Trunk/Main
- Head

Basic Checkins



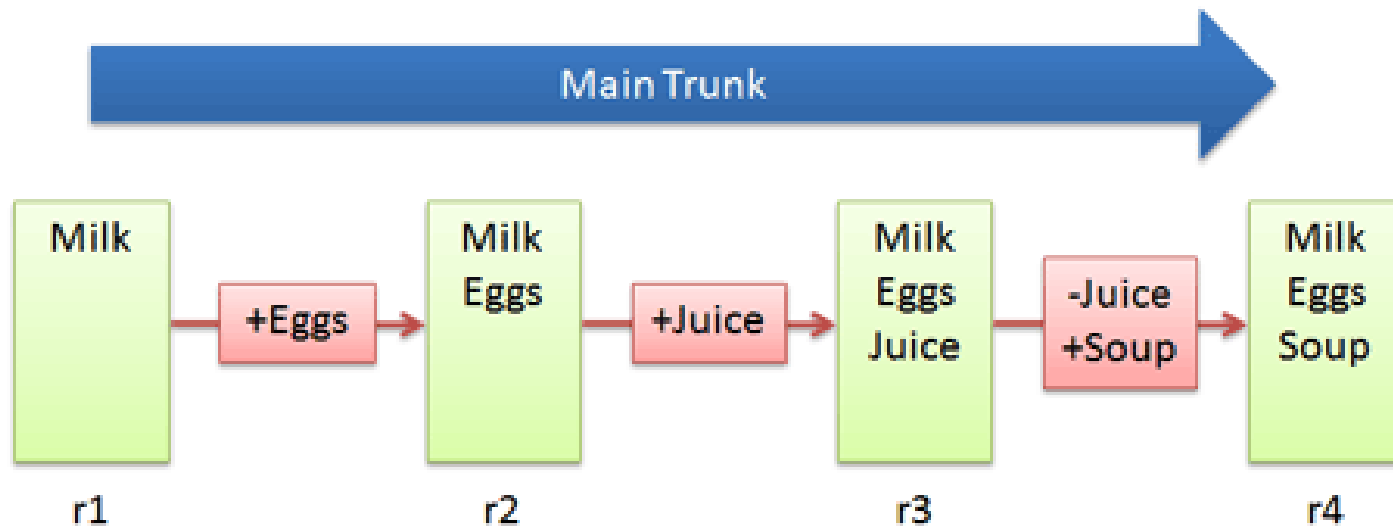
- Checkout a edit
- Revert
- Update

Checkout and Edit

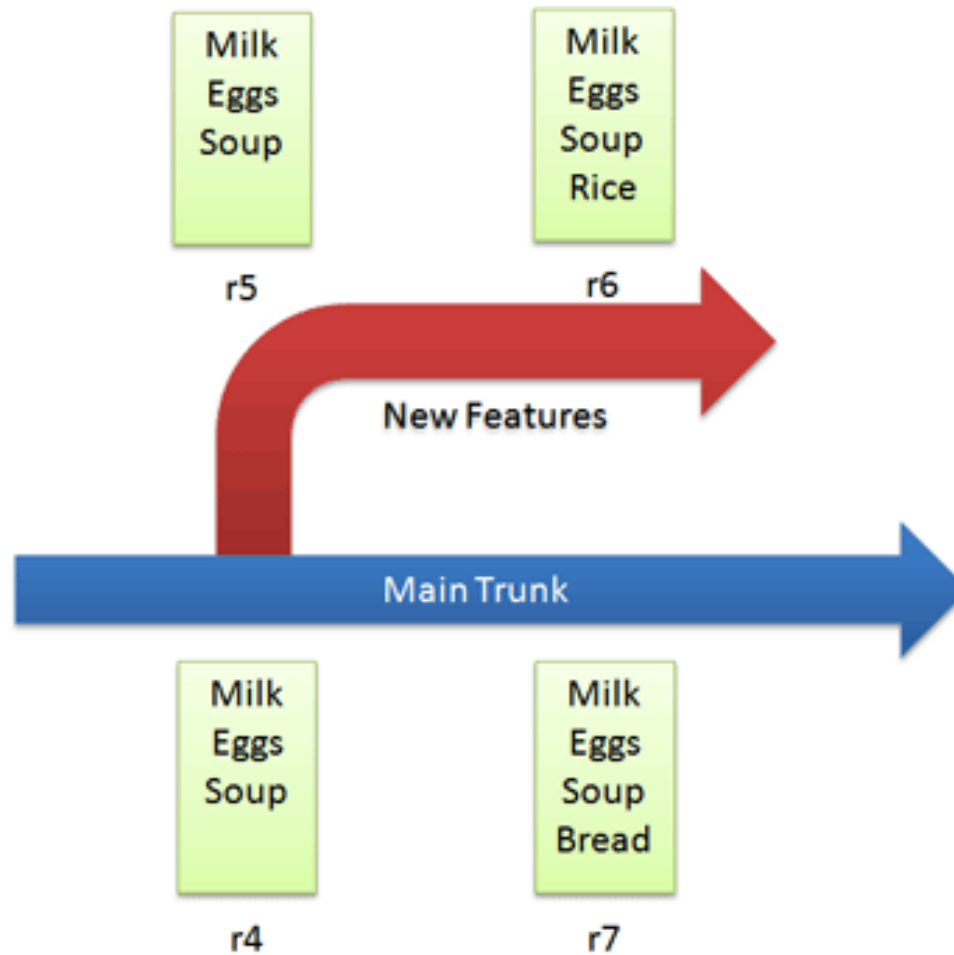


- Diffs
- Aký je rozdiel medzi r1 a r4?

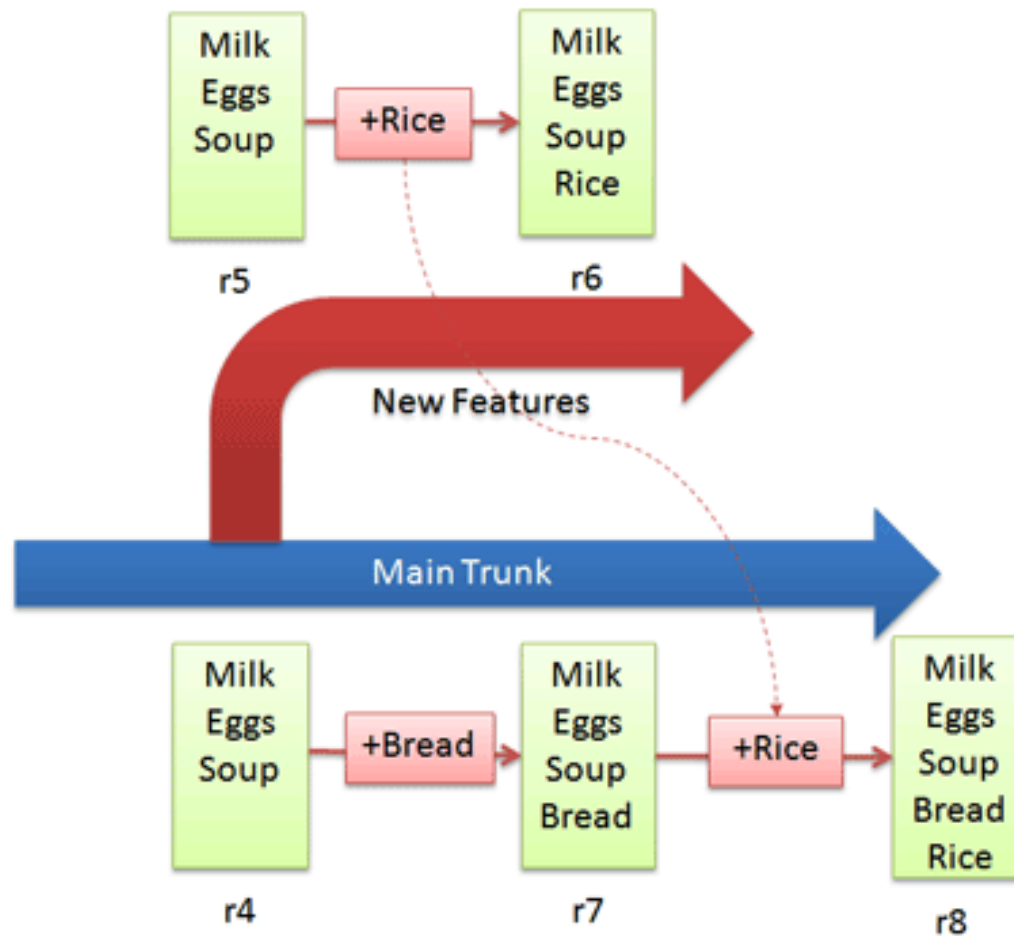
Basic Diffs



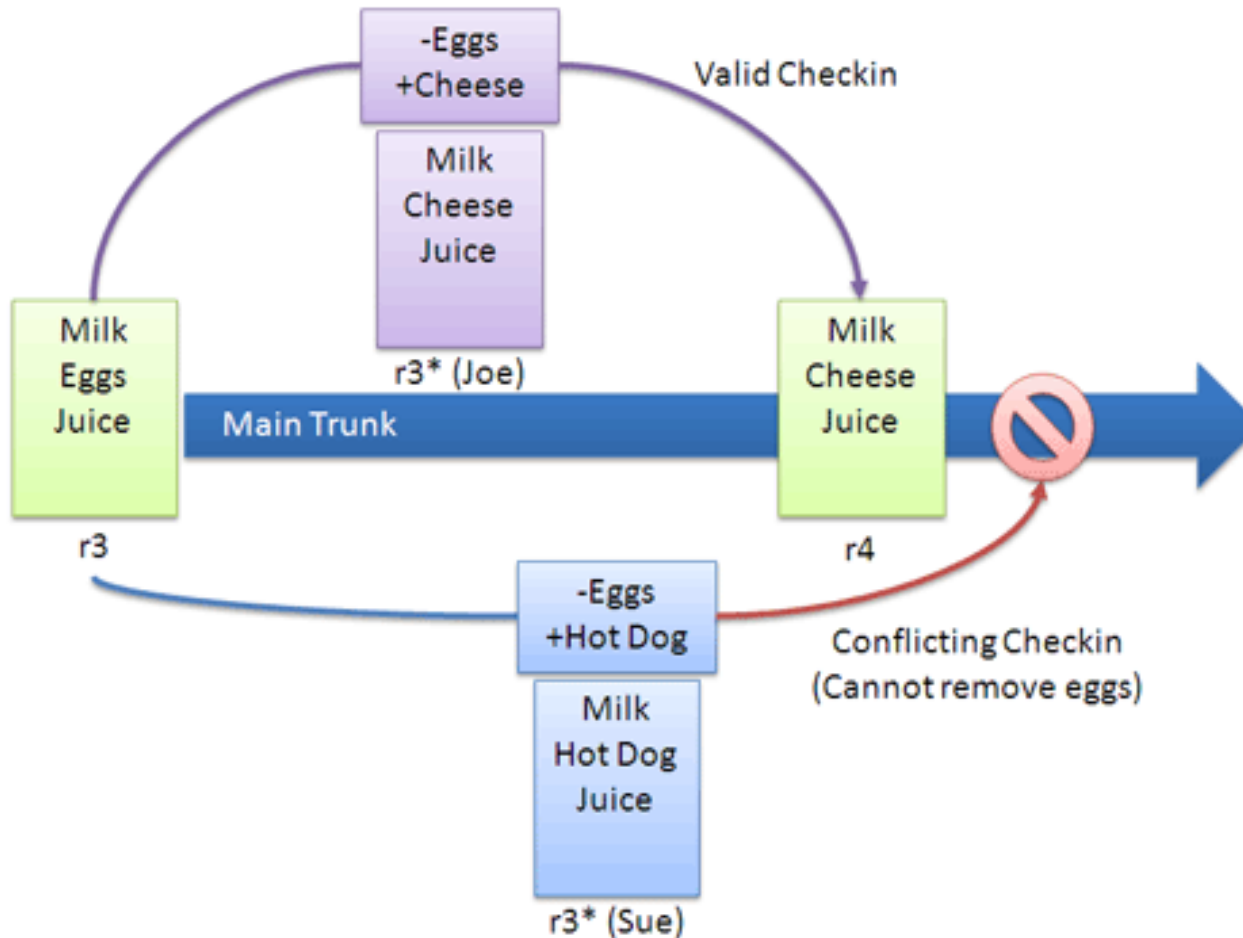
Branching



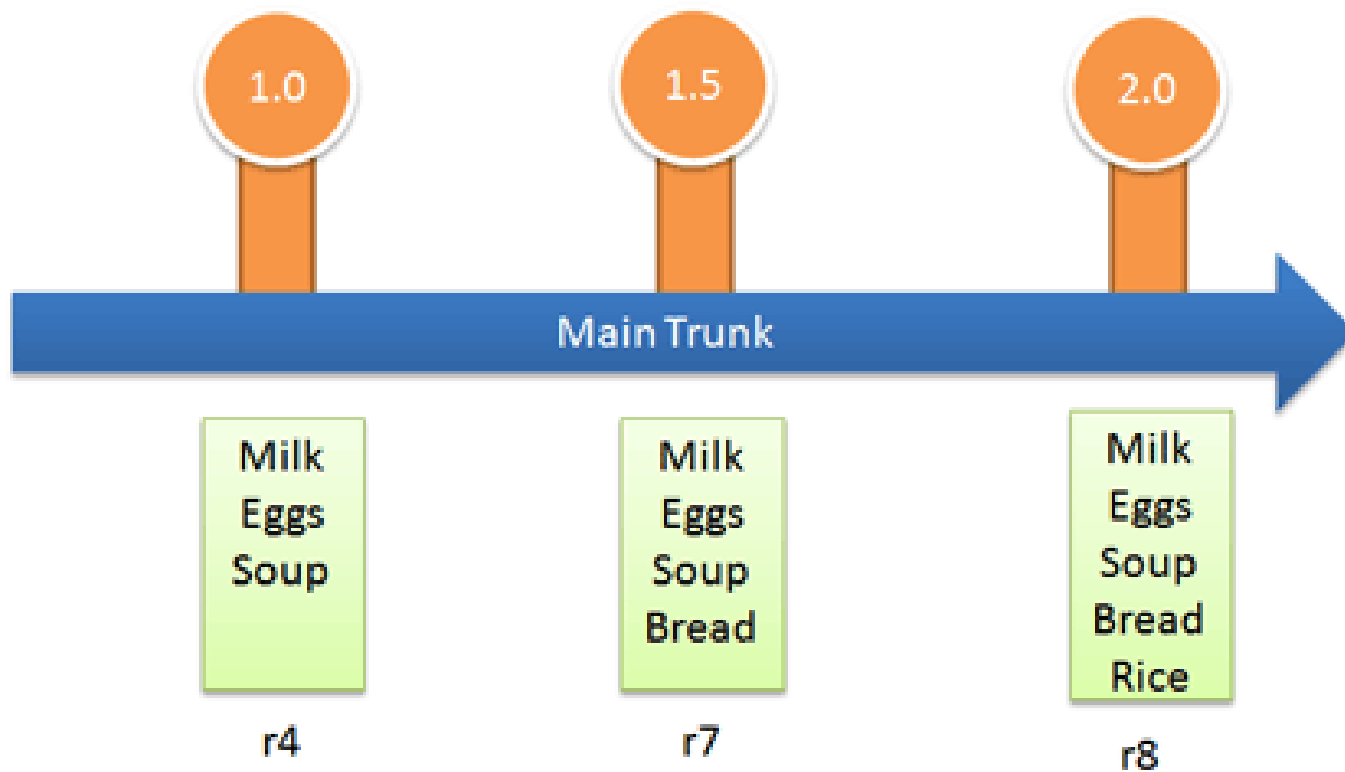
Merging



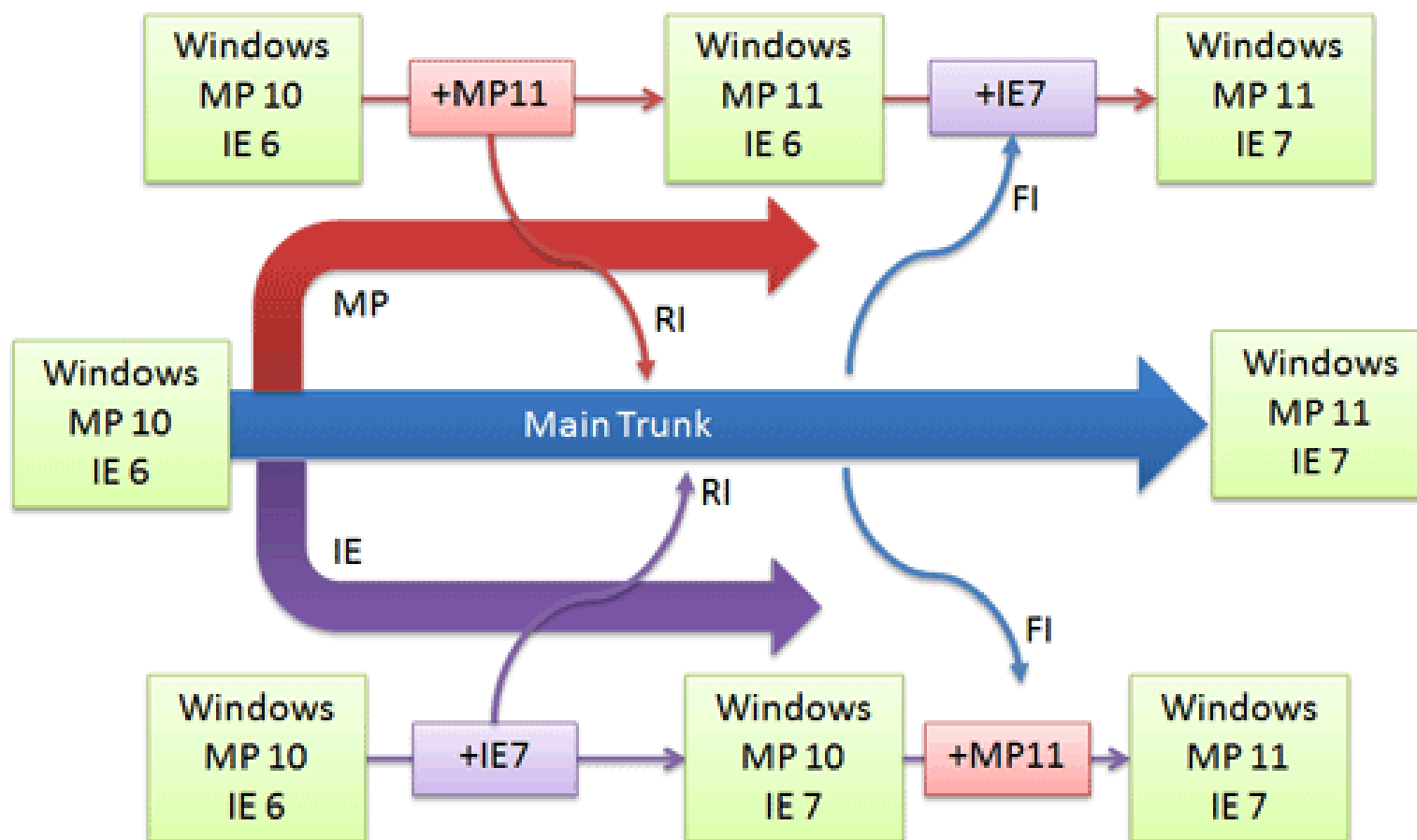
Conflicts



Tagging



Managing Windows



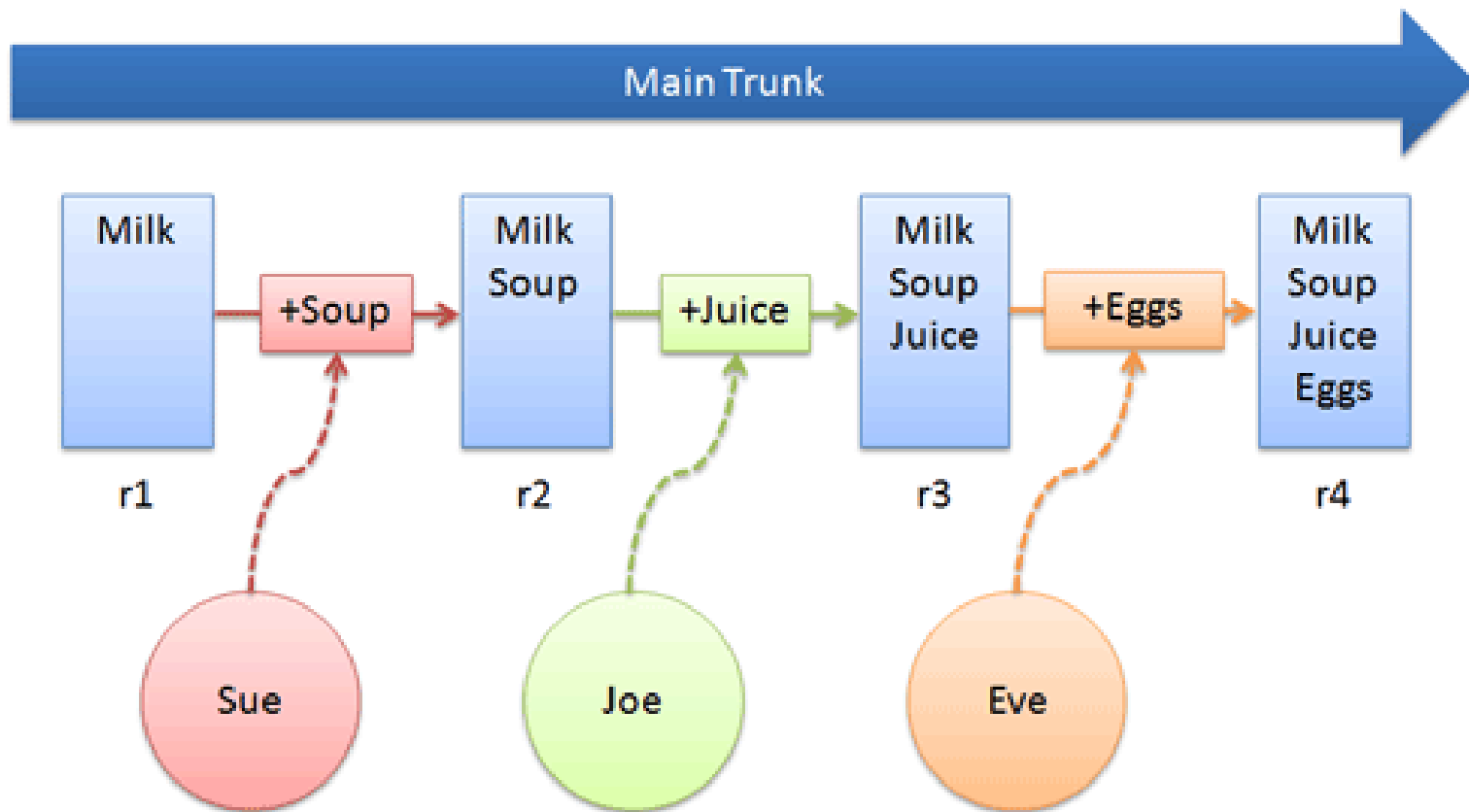
Distribučovaná správa verzií

- Distribučovaná
 - jednoduchšie zdieľanie zmien
- Centralizovaná
 - Jednoduché na backup, undo a synchronizáciu
 - Horšie je to so spájaním a vetvením



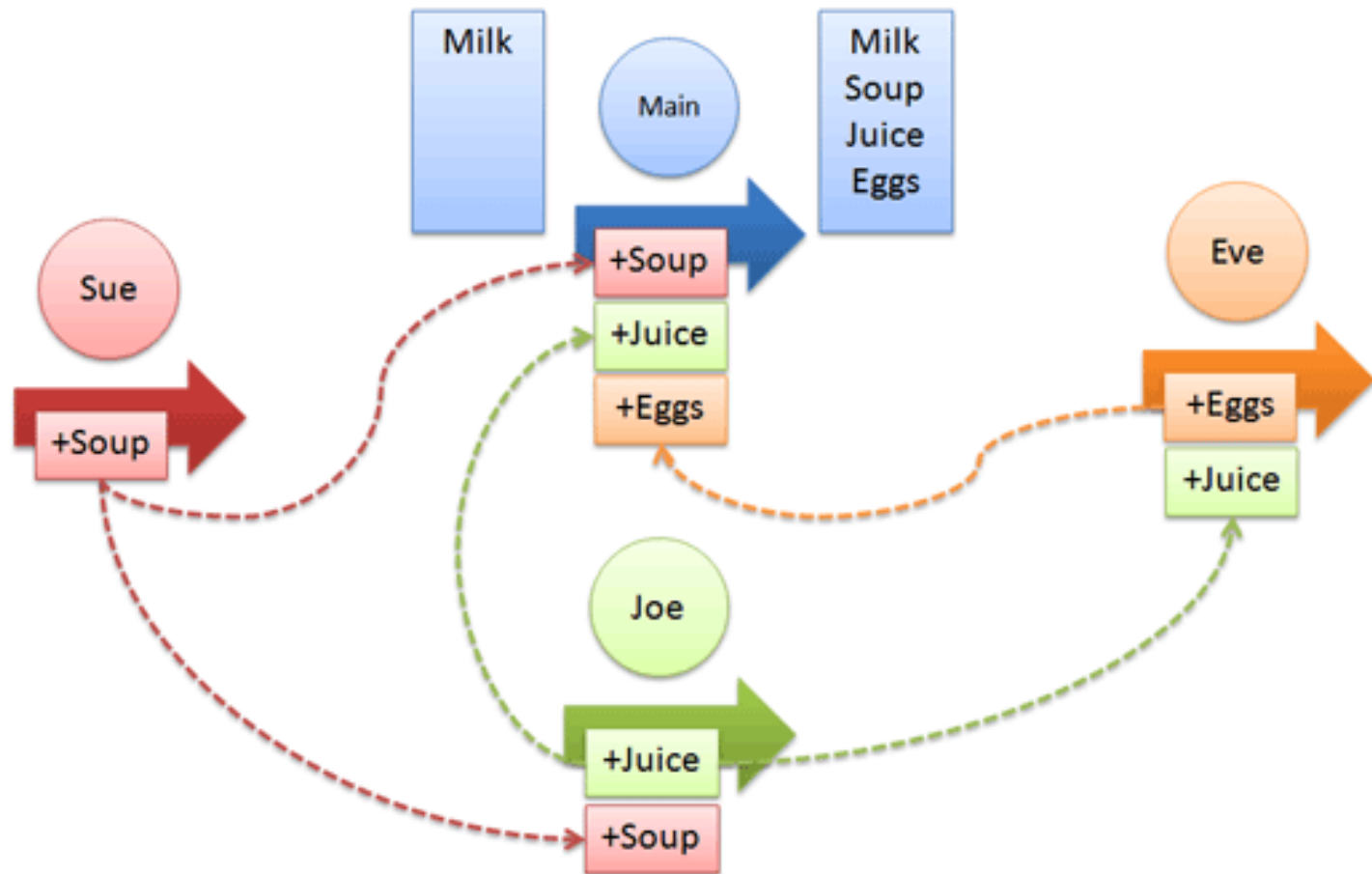
Centralizovaný VCS

Centralized VCS



Distribučovaný VCS

Distributed VCS



- Push – pošli zmeny do iného repozitára
- Pull – zober si zmeny z iného repozitára

Výhody

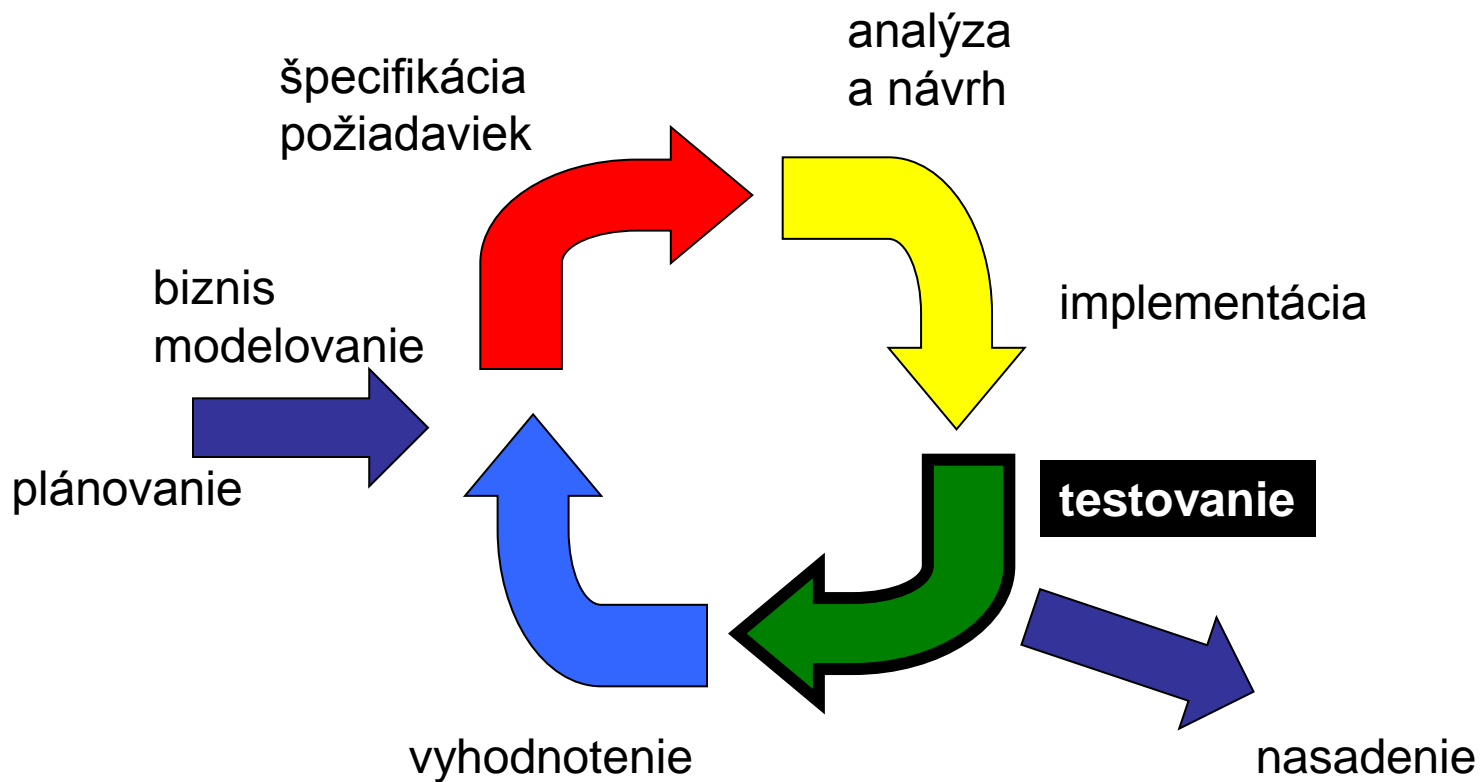
- Každý ma svoje pieskovisko
- Funguje aj offline
- Je rýchly
- Dobre spravuje zmeny
- Vetvenie a spájanie je ľahké
- Menej správy

Nevýhody

- Stále potrebujete zálohy
- Neexistuje niečo ako „posledná verzia“
- Nie sú tu skutočné čísla revízií

- Používajte veržionovanie
- Krok za krokom
- Nevzdávajte to
 - Podstatný je koncept

RUP – iterácie



UML nemá žiadny diagram

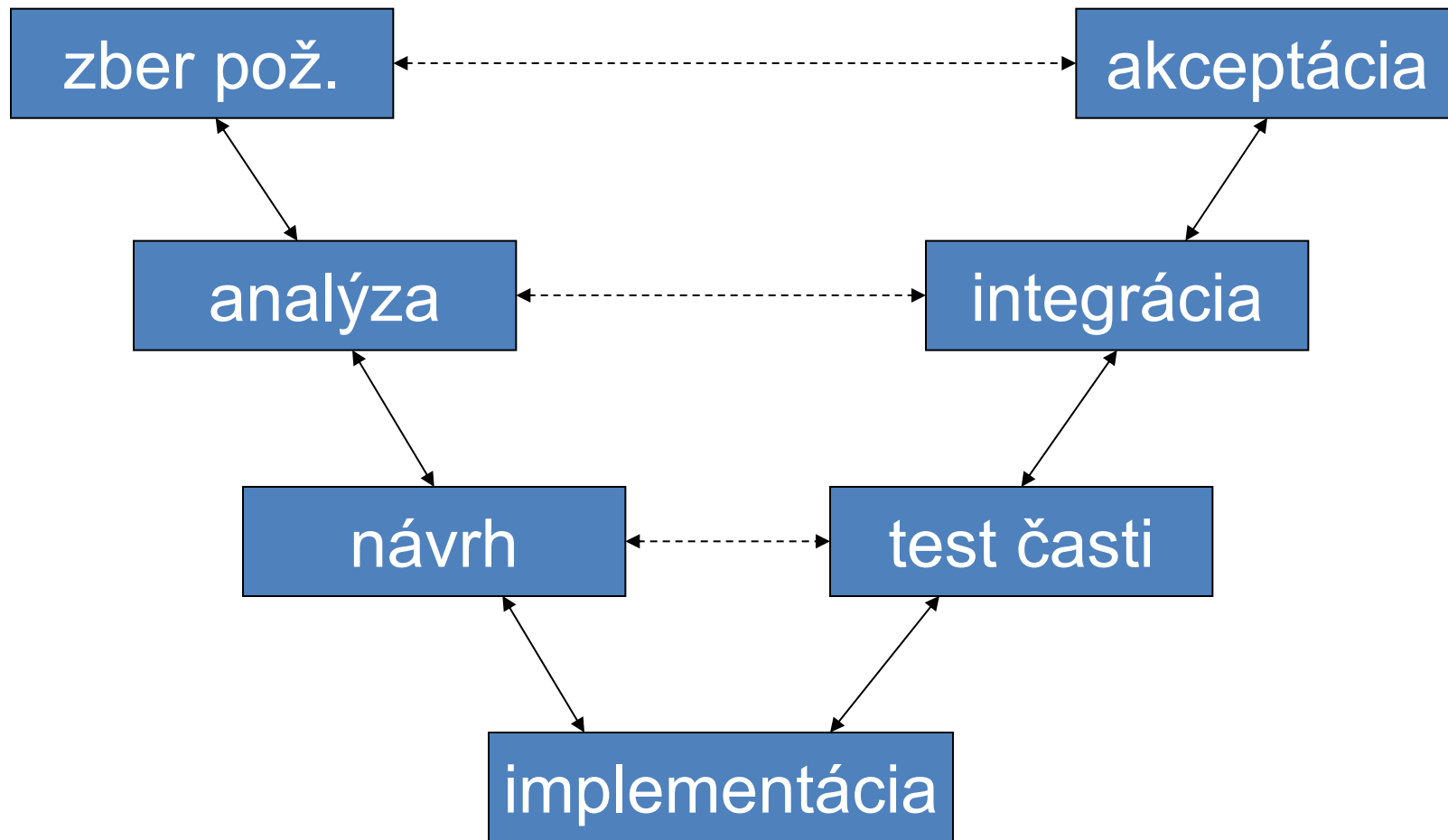
Model tvoria:

- testovacie úlohy – čo testovať, aké vstupy, výstupy
- testovacie procedúry – ako vykonať úlohu
- testovacie komponenty – automatizácia procedúry

pre každú iteráciu

- integračné testy – v priebehu iterácie
 - každý vytvorený produkt
- systémové testy – koniec iterácie
 - spustiteľná verzia aplikácie

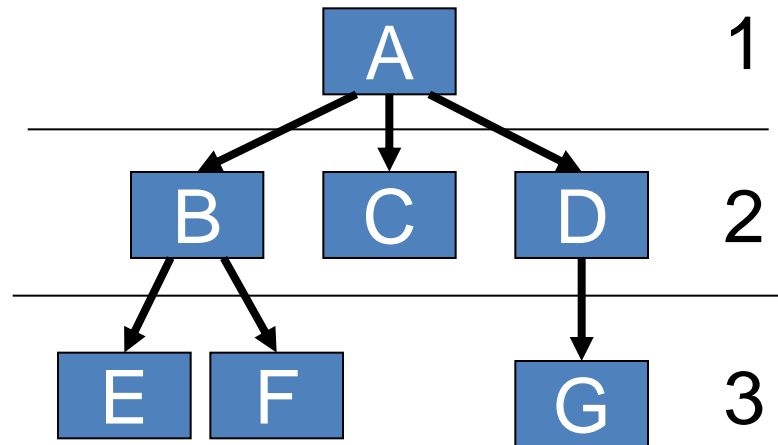
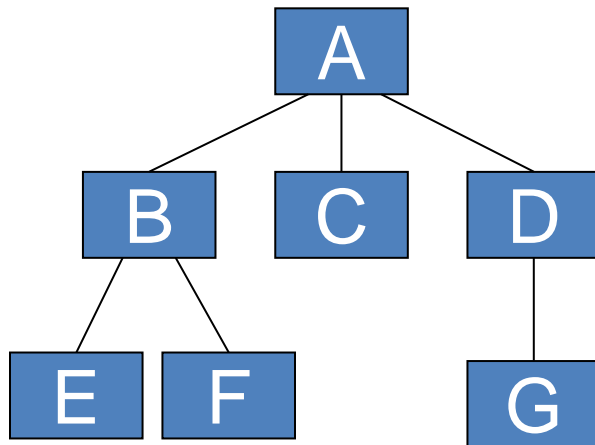
Testovanie



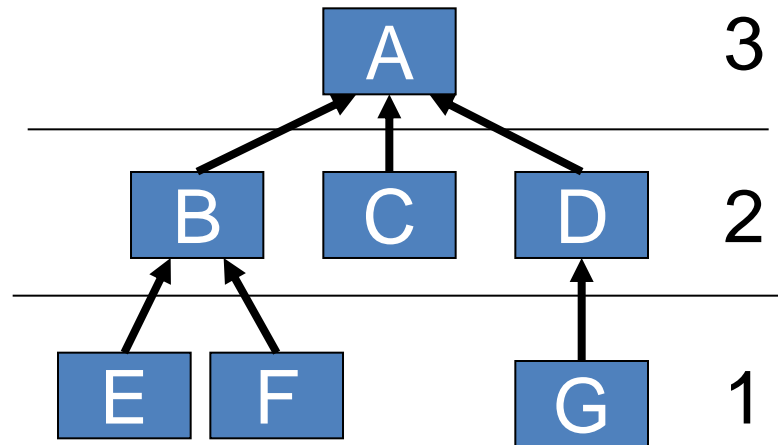
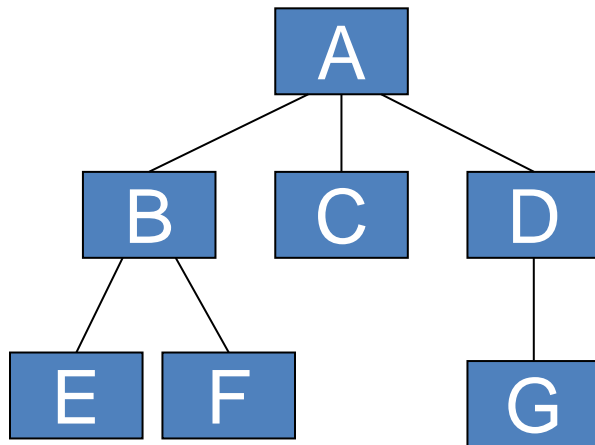
Stratégie testovania

- testovanie zhora nadol
- testovanie zdola nahor
- jednofázové testovanie
- sendvičové testovanie
- testovanie porovnávaním

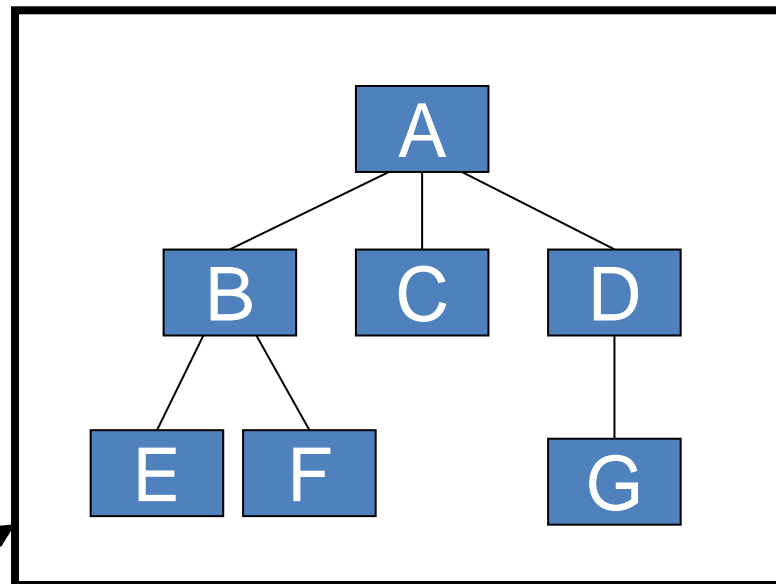
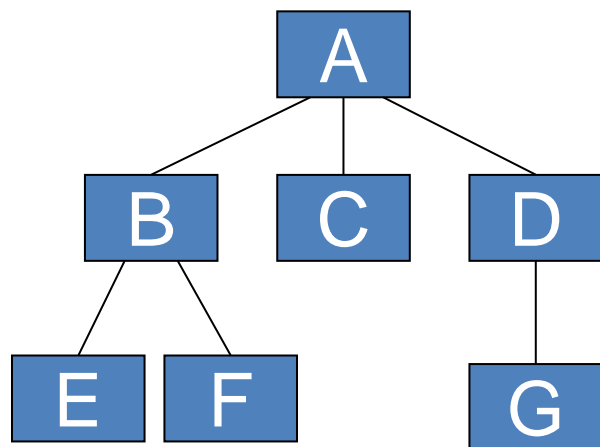
Testovanie zhora nadol



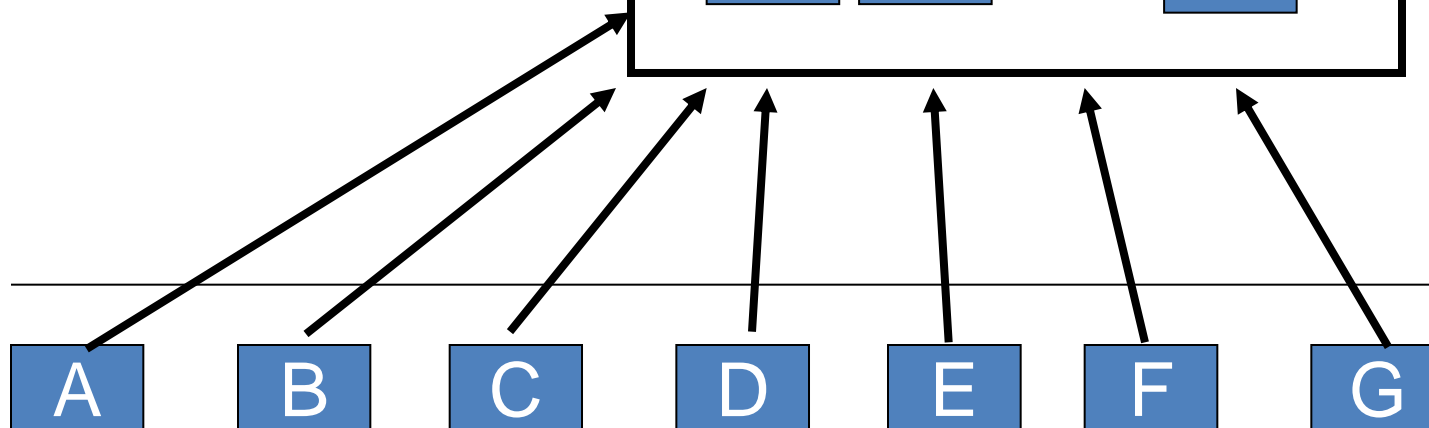
Testovanie zdola nahor



Testovanie jednofázové

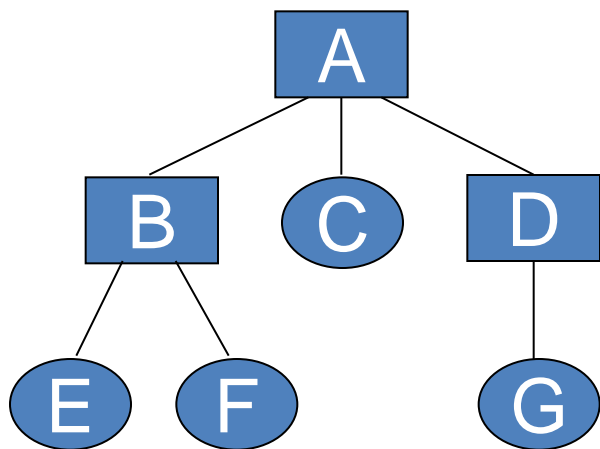


2



1

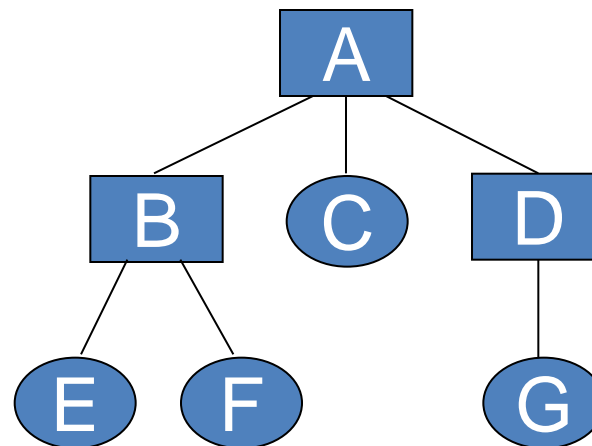
Testovanie sendvičové



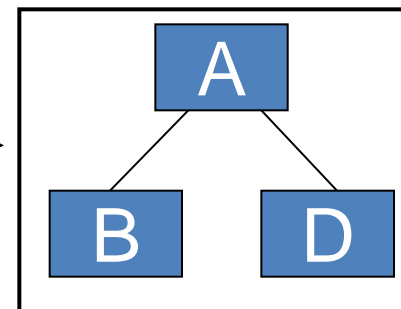
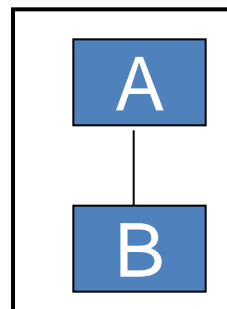
logický modul



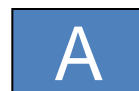
funkčný modul



3

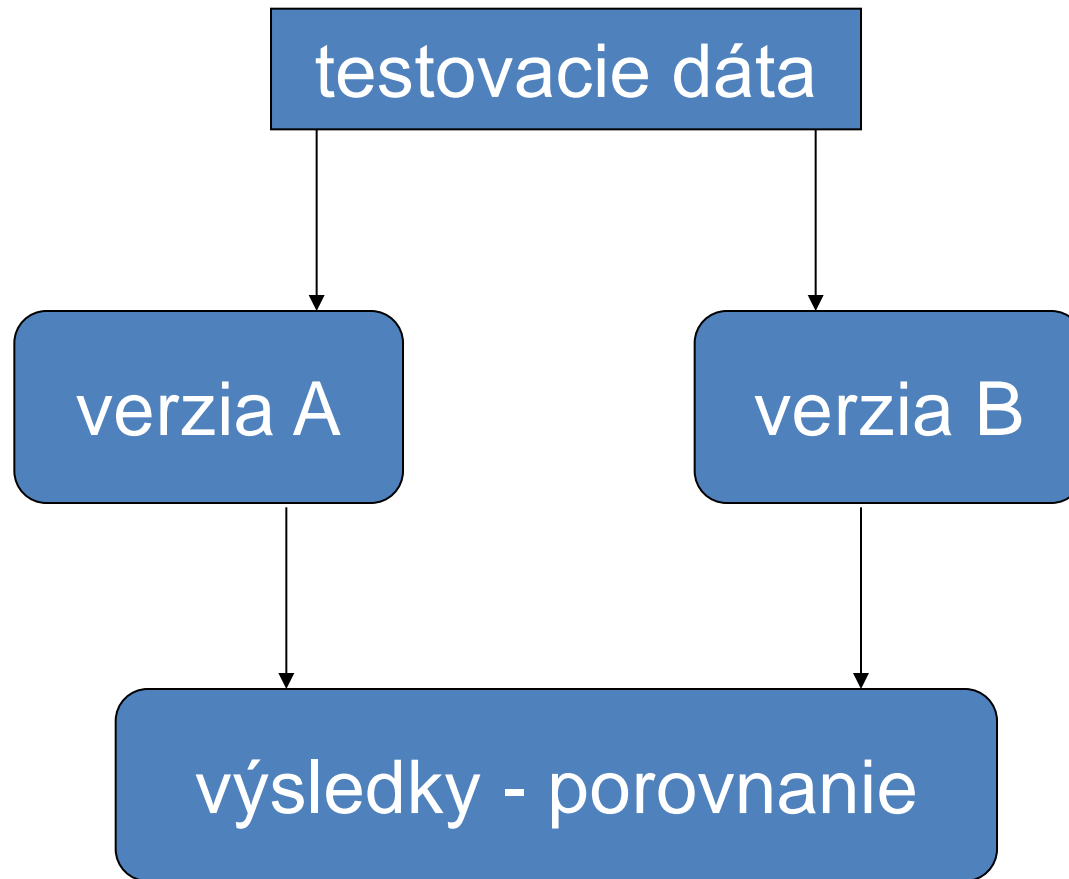


2

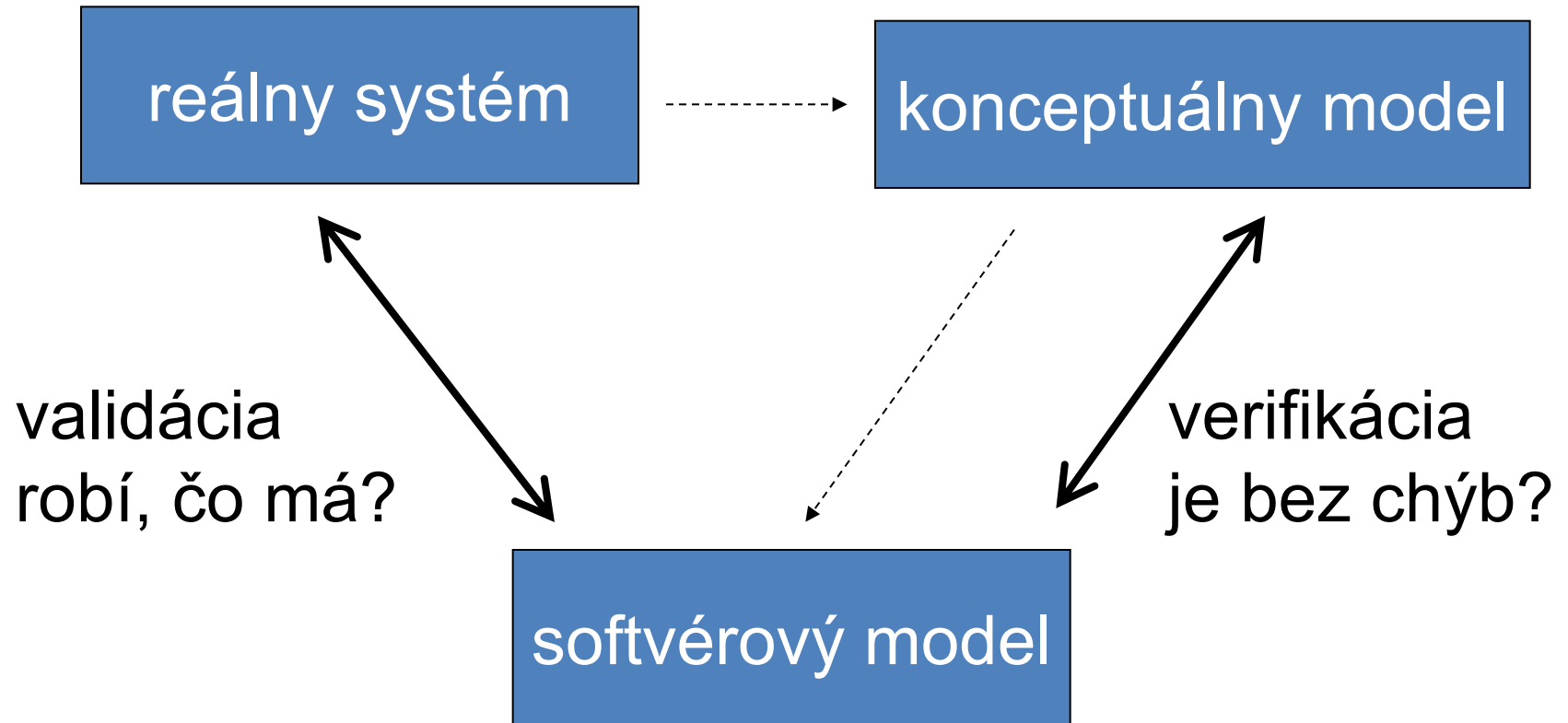


1

Testovanie porovnávacie



Verifikácia - Validácia



užívateľ testuje, či systém spĺňa zadanie

- správnosť – plní požiadavky
- robustnosť – odolný voči chybám
- výkonnosť – pamäťová a časová náročnosť
- dokumentácia – úplnosť a zrozumiteľnosť

Testovanie alfa, beta

produkt na všeobecný predaj – nie je na objednávku

alfa testovanie

- vývojárska firma
- užívateľ sledovaný vývojármi
- známe prostredie nasadenia

beta testovanie

- užívatelia na svojich počítačoch
- rôznorodé prostredie – neznáme pre vývojárov
- správa od užívateľa > modifikácia > servis pack

RUP – iterácie

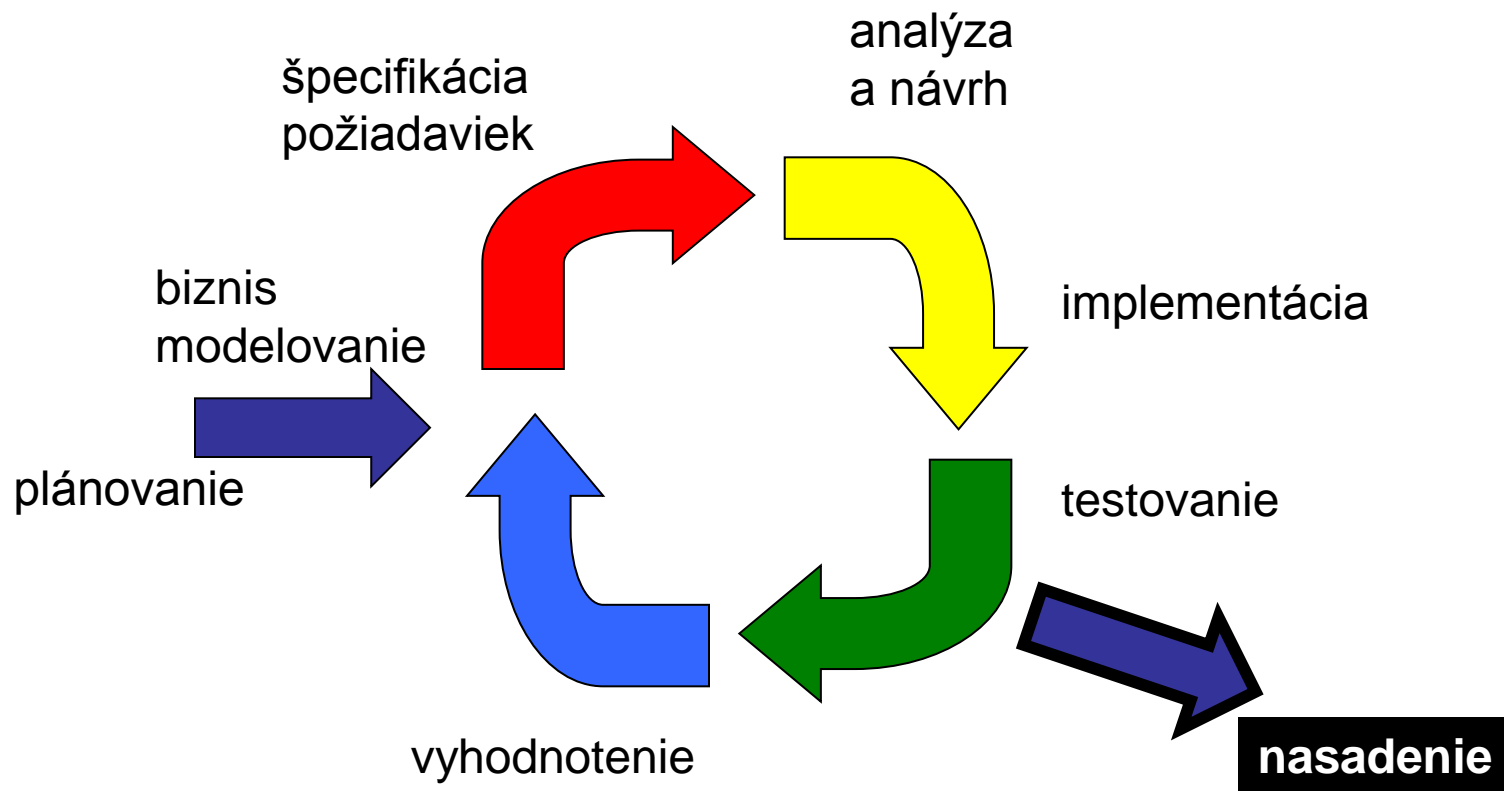
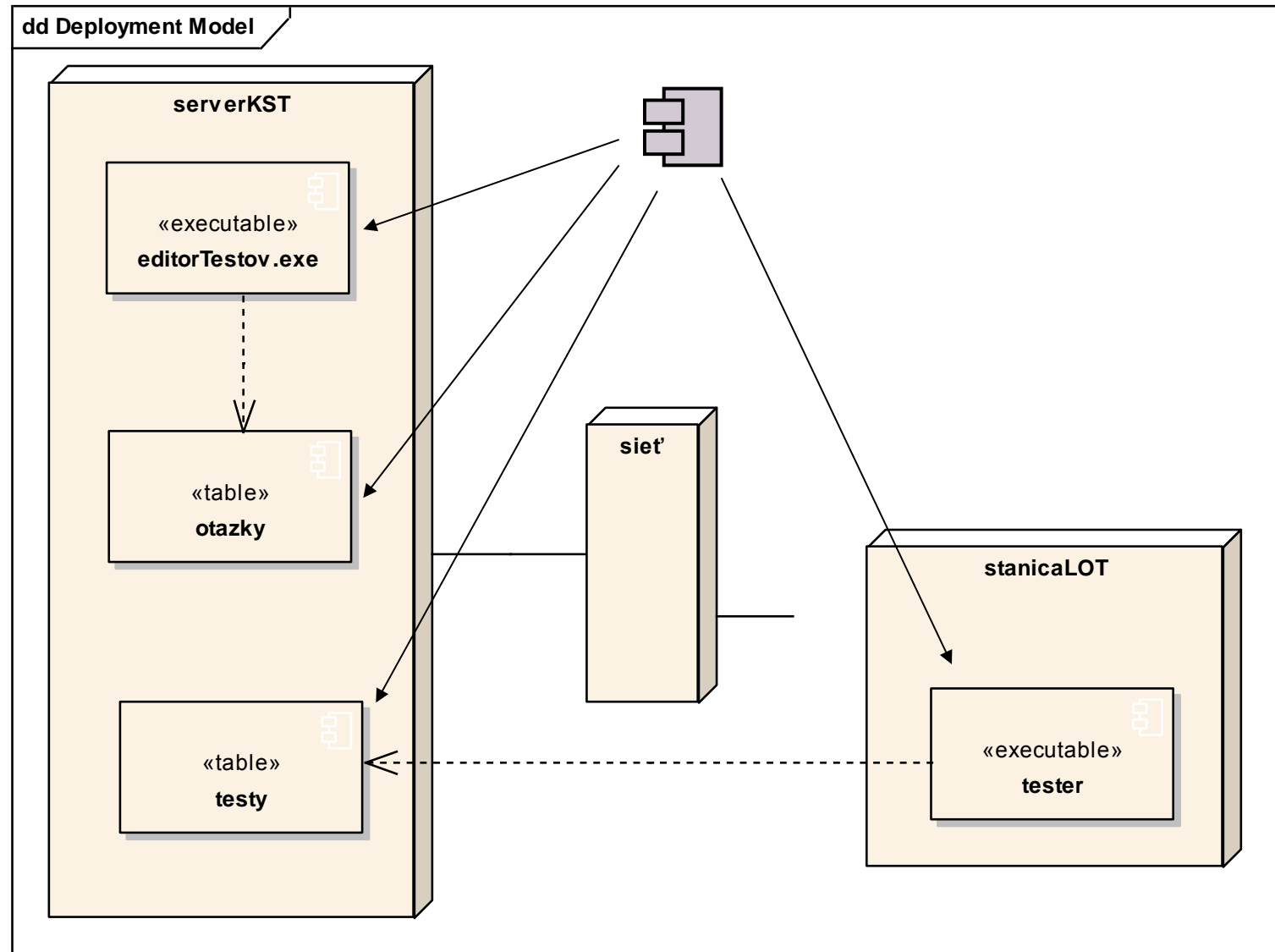


Diagram nasadenia



- inštalácia systému (u zákazníka)
- testovanie po inštalácii (beta)
- plán zálohovania a obnovy
- uvedenie do rutínnej prevádzky

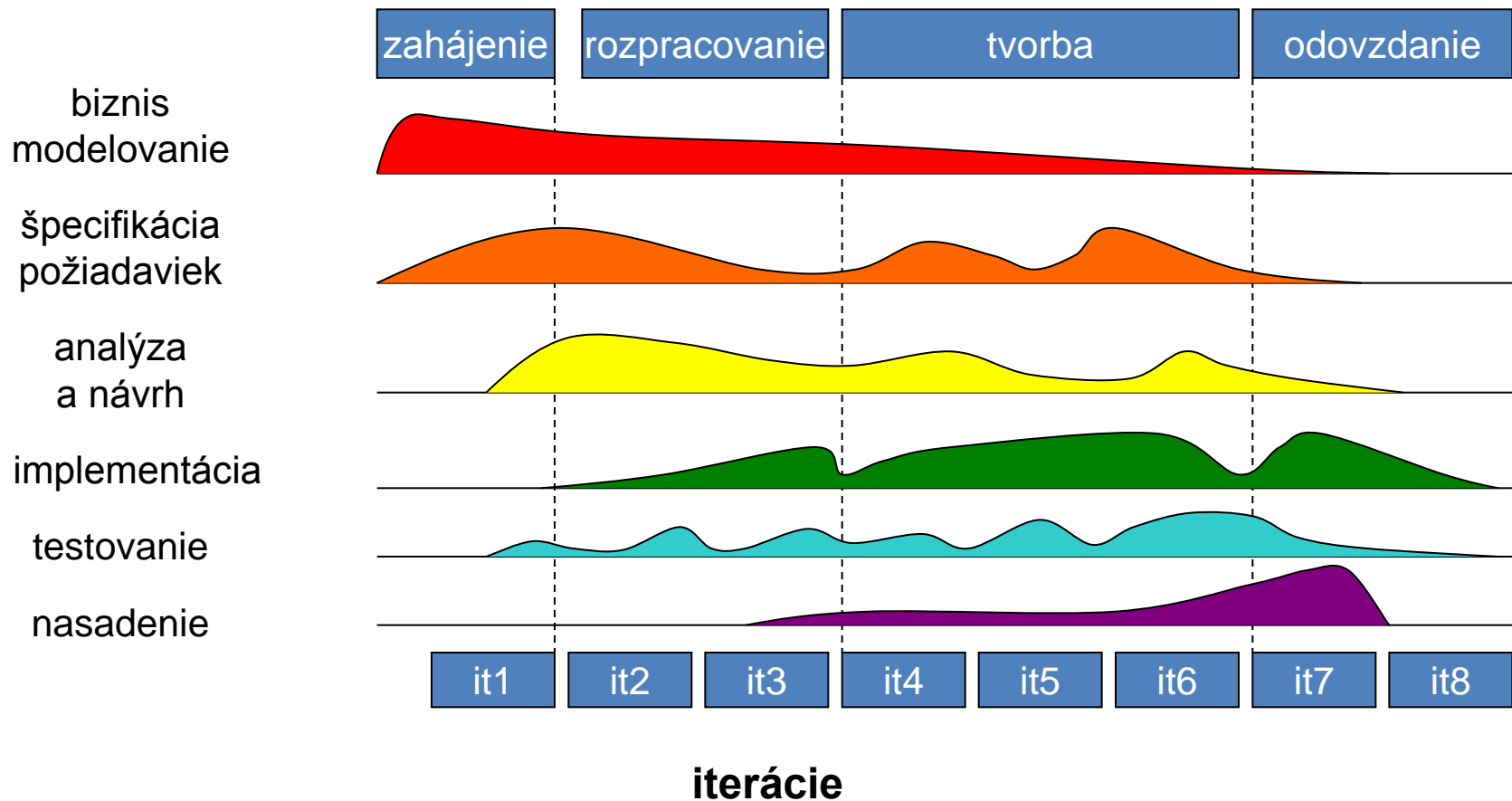
- diagramy UML
- zdrojové texty programov
 - komentáre – nová a aktuálna informácia
 - identifikátory – seba dokumentujúce
- užívateľská príručka

CASE – generovanie dokumentácie

RUP – schéma (obsah x čas)

tok činností

fázy



Ďakujem za pozornosť

Vaše otázky...

