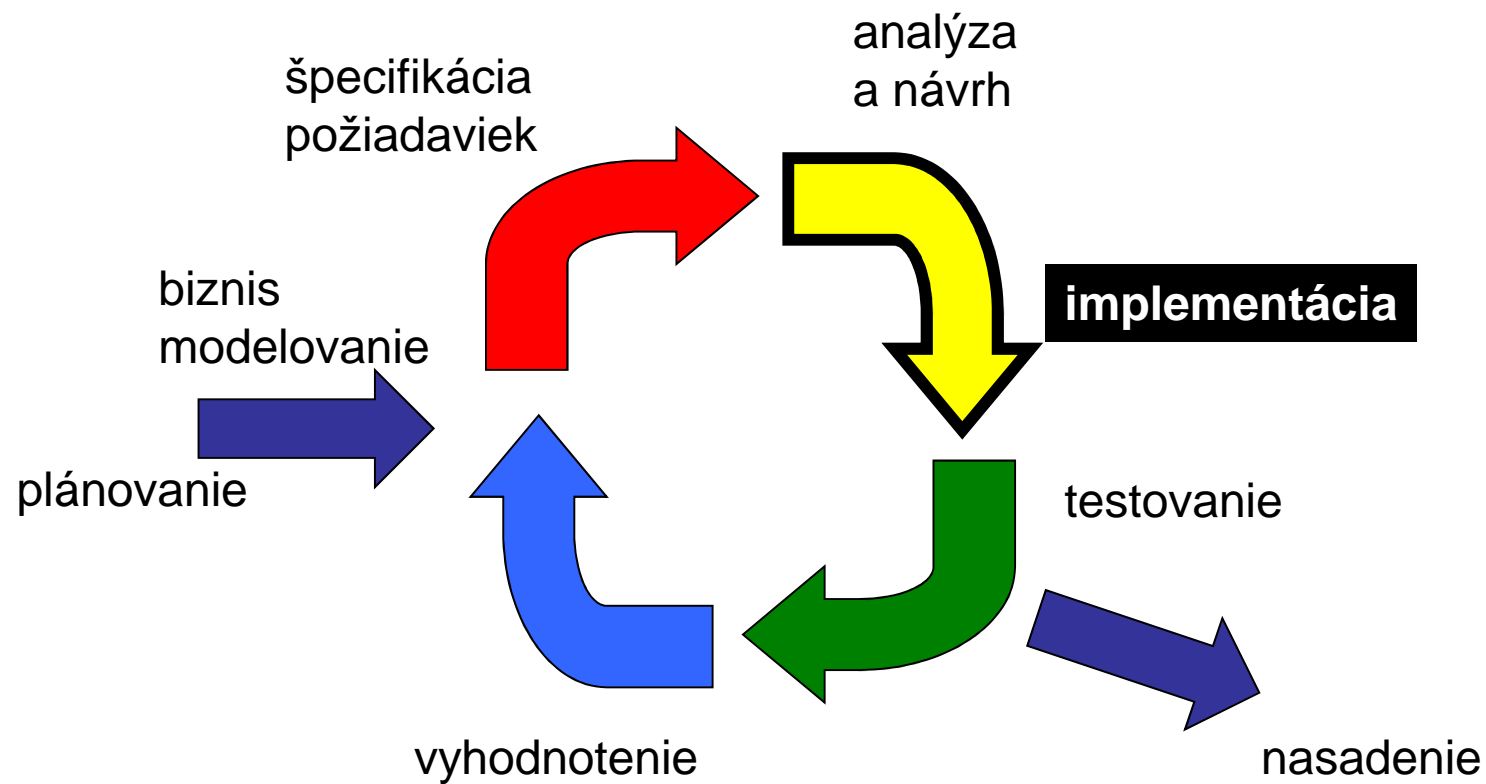


10

Implementácia,
testovanie, nasadenie

RUP – iterácie



Cieľ implementácie

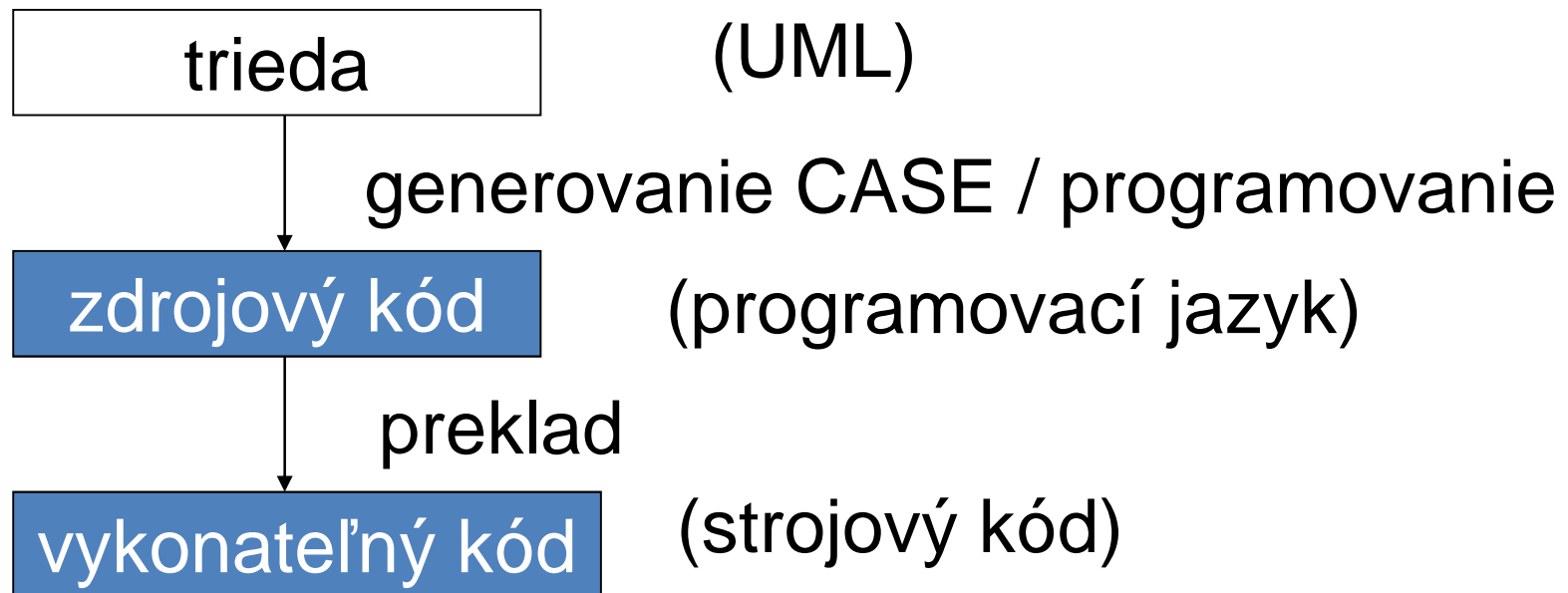
- vytvoriť fungujúci softvér
- model implementácie tvoria softvérové komponenty
 - zdrojový kód prvkov z modelu návrhu
 - vykonateľný (binárny kód) – preklad
 - databázové tabuľky
- konkrétny programovací jazyk
- konkrétny databázový systém

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

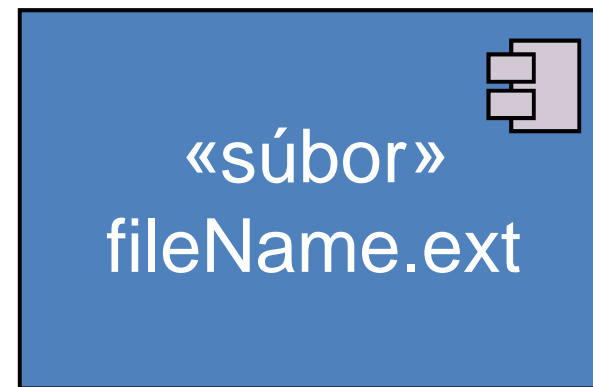


Diagram tried

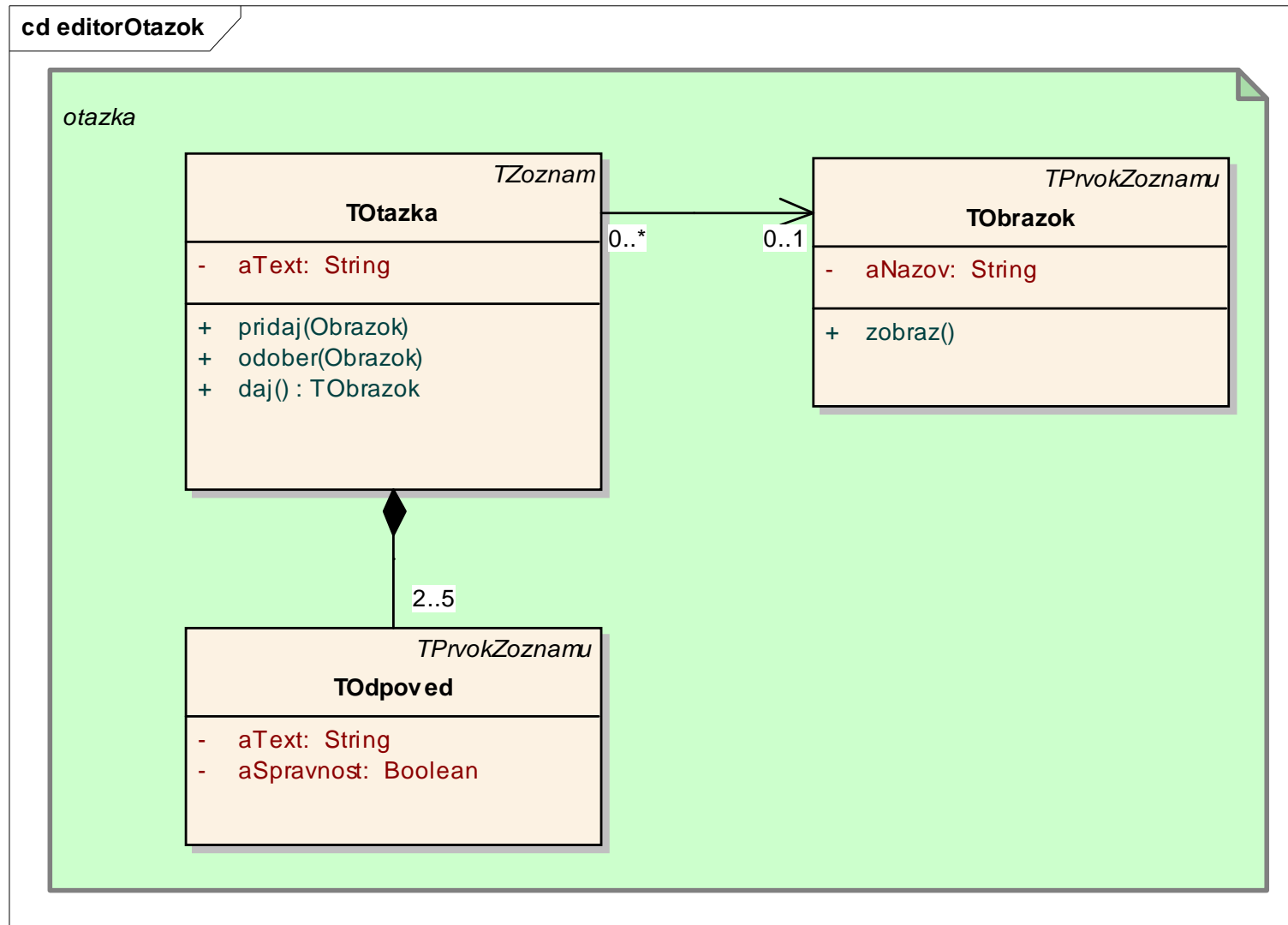
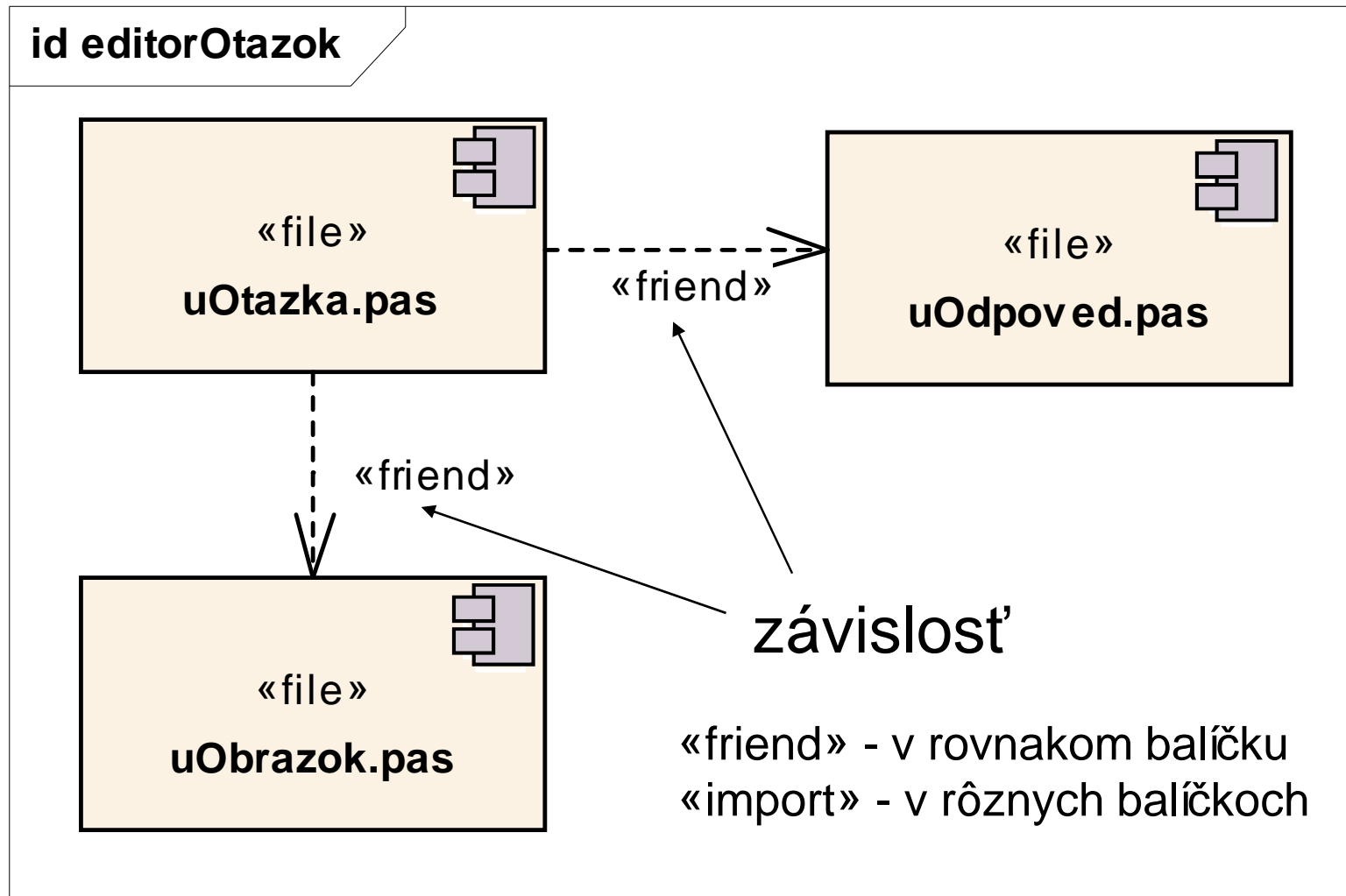


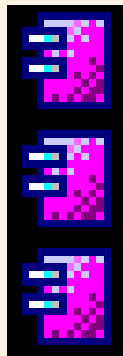
Diagram komponentov



Balíček komponentov

id Component Model

editorOtazok

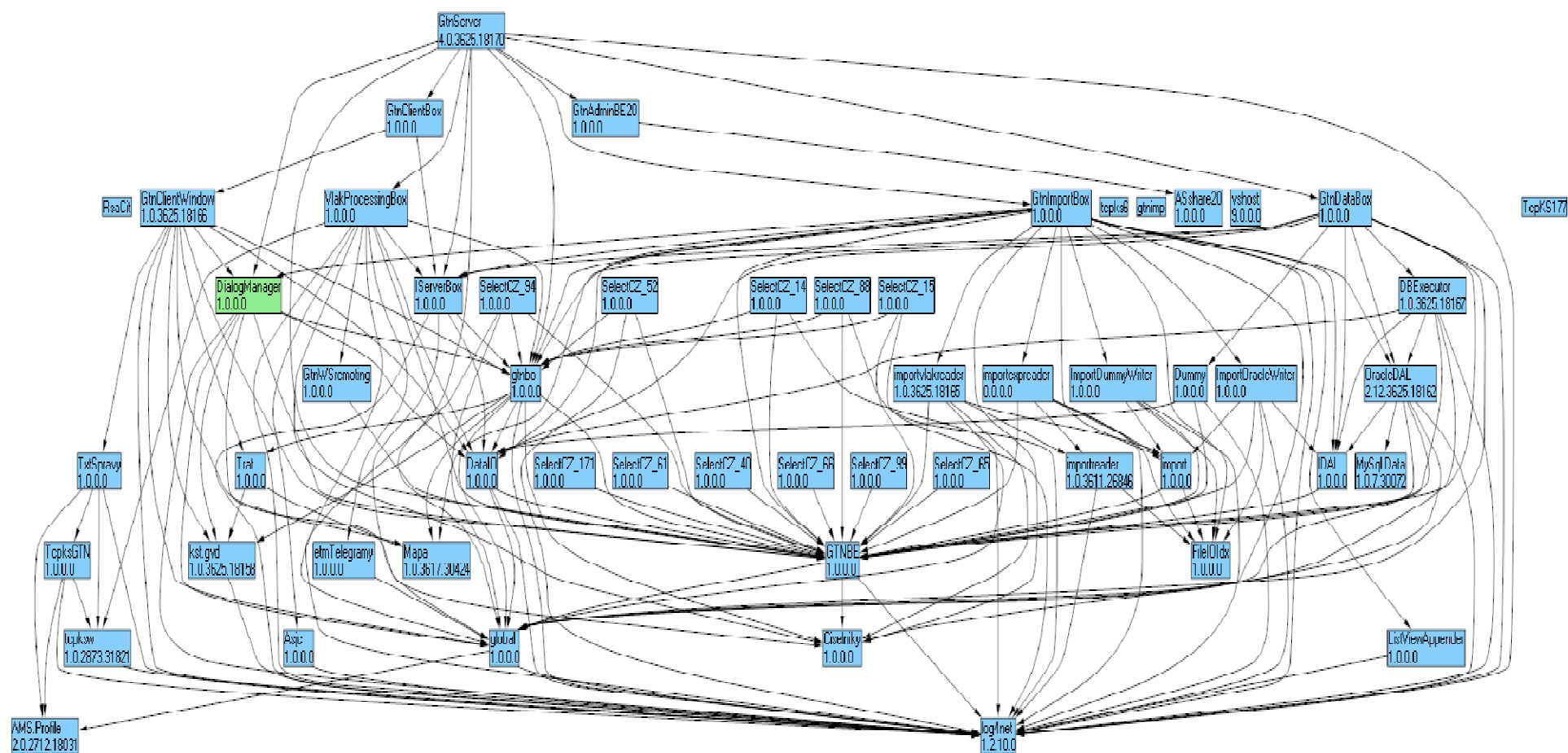


+ uObrazok.pas

+ uOdpoved.pas

+ uOtazka.pas

Reálny projekt



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;
```

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

doprogramovať ručne!



stavový diagram
diagram aktivít
sekvenčný diagram

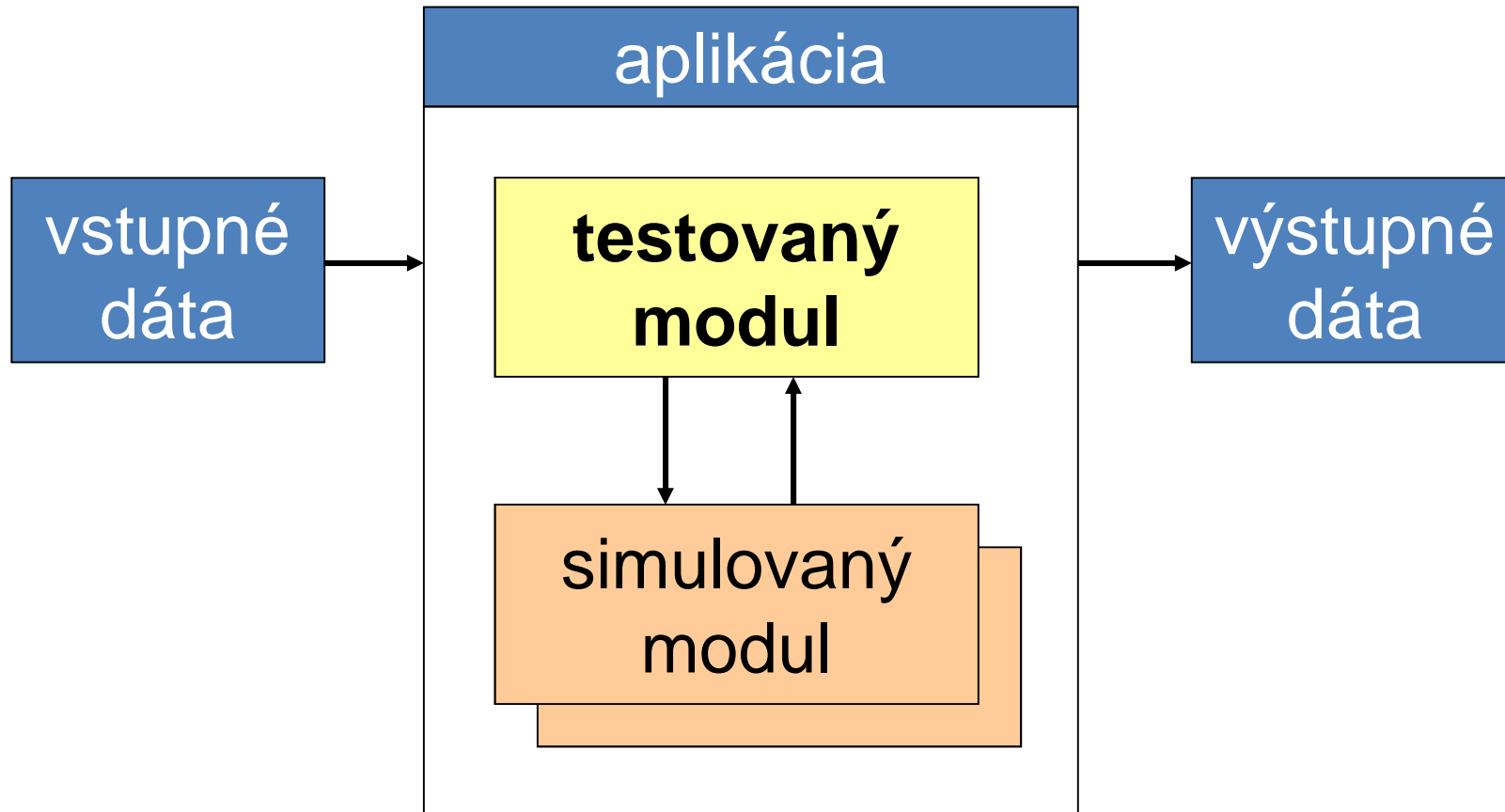
delphi - ukázať

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

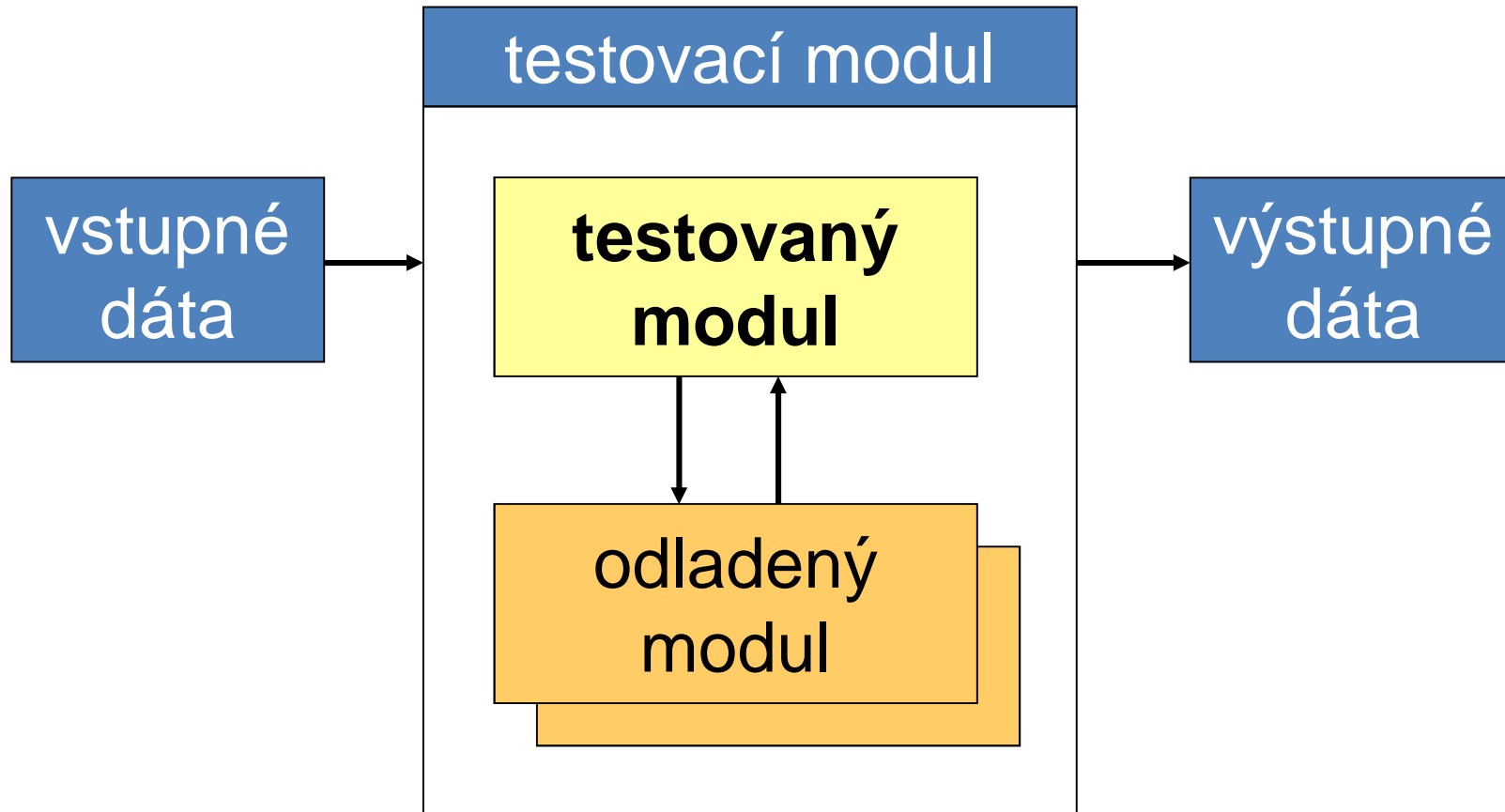
výhody

- 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

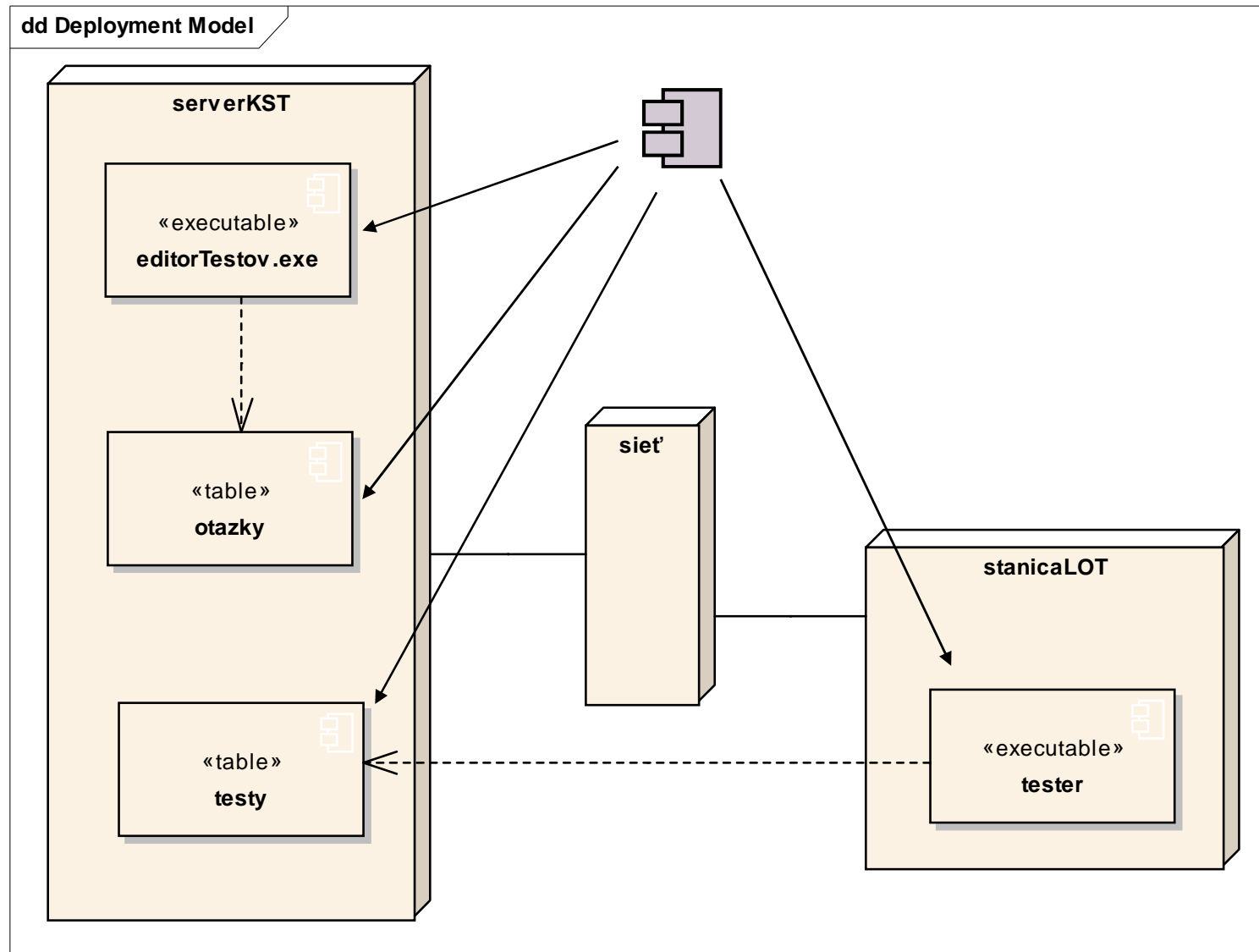
Štruktúrované programovanie

- riadiace konštrukcie: postupnosť, vetvenie, cyklus
- modul má jeden vstup a jeden výstup
- neštruktúrované konštrukcie (go to, break, continue) len ak máme výnimočne dobrý dôvod

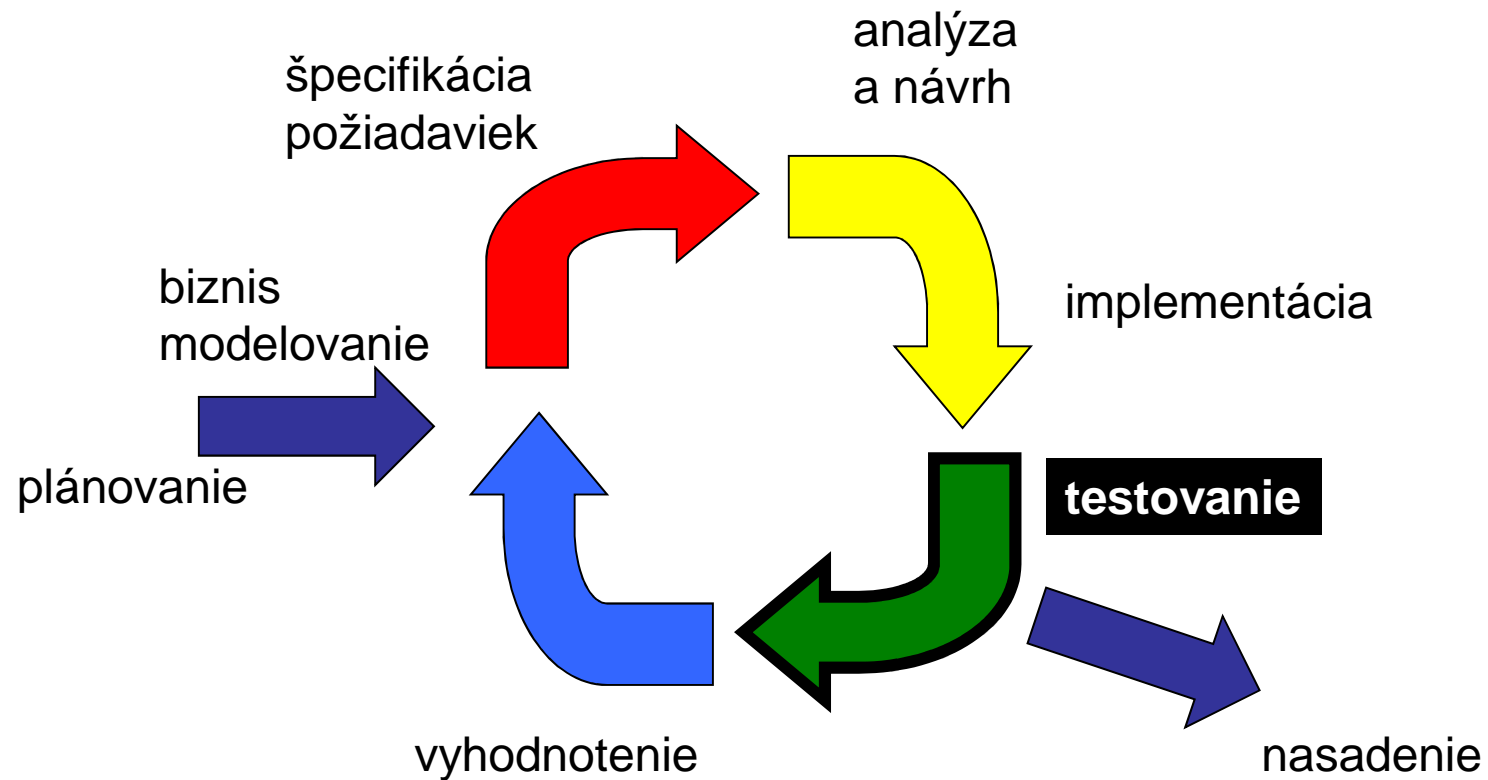
Štýl programovania

- 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 príkaz – jeden riadok...

Diagram nasadenia



RUP – iterácie



Model testovania

UML nemá žiadny diagram

Model tvoria:

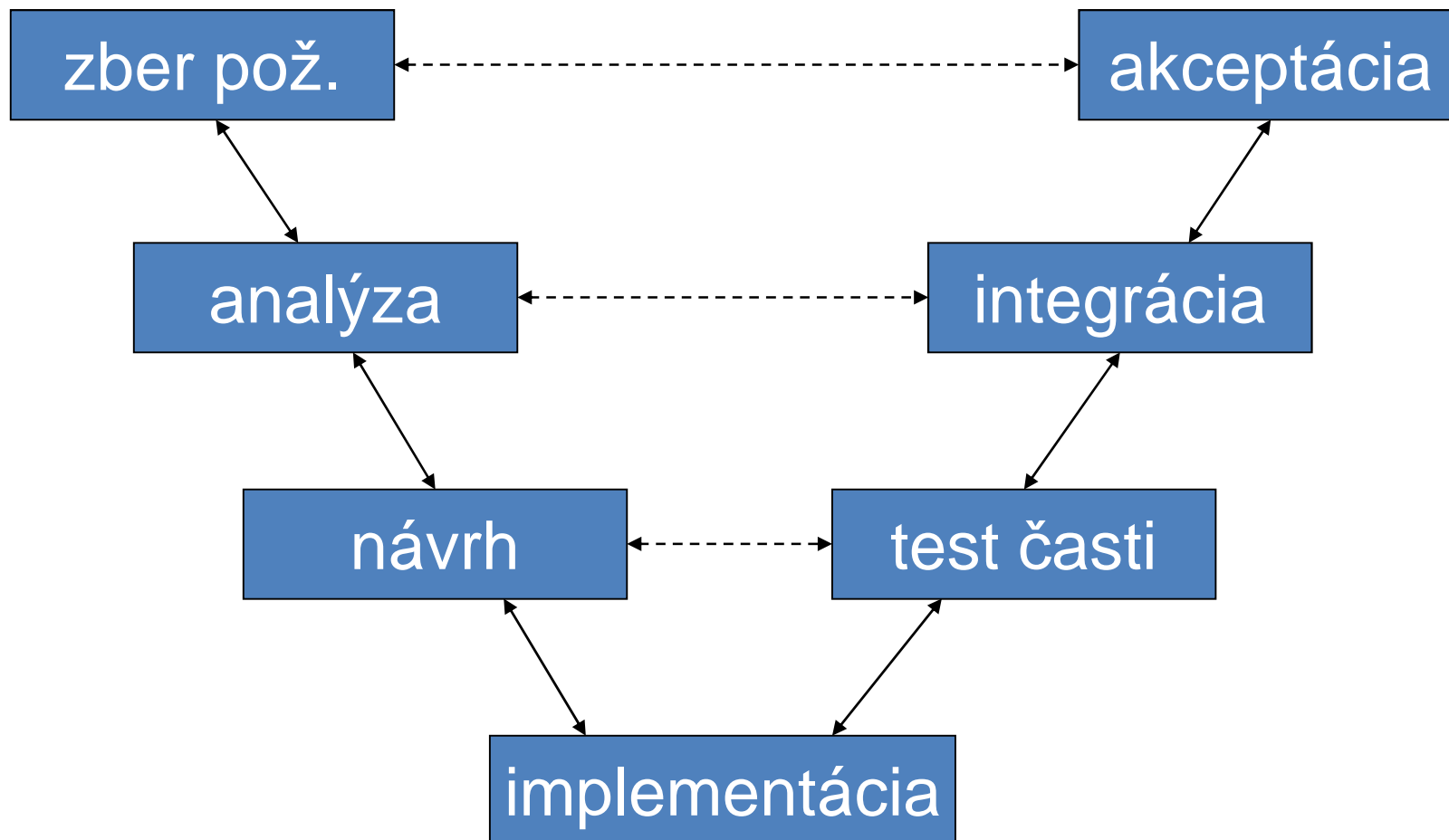
- testovacie úlohy – čo testovať, aké vstupy, výstupy
- testovacie procedúry – ako vykonať úlohu
- testovacie komponenty – automatizácia procedúry

Testovanie

pre každú iteráciu

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

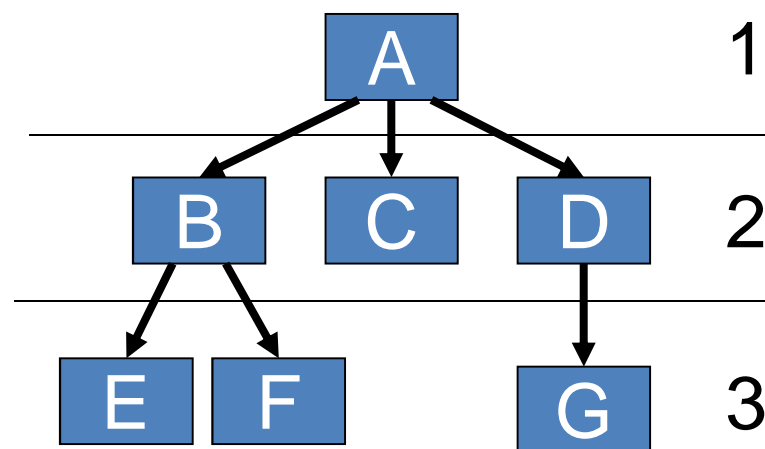
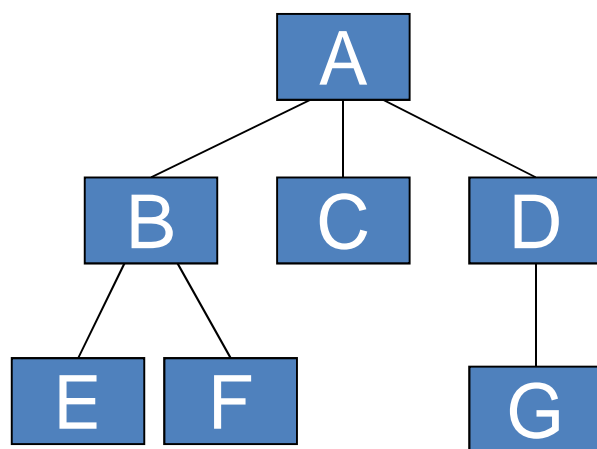
Testovanie



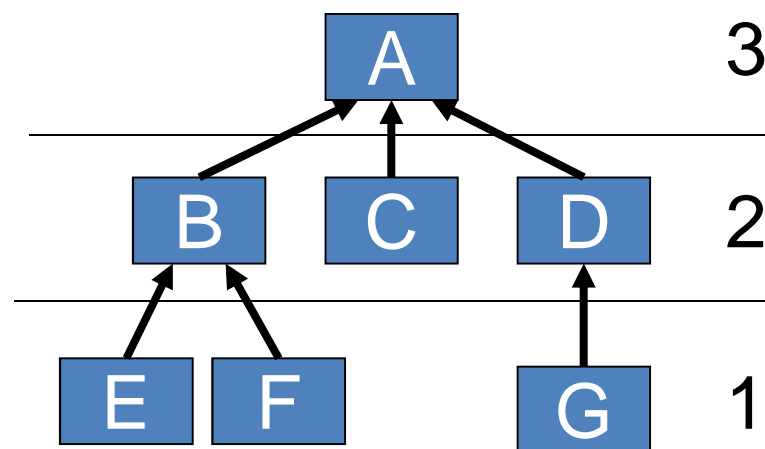
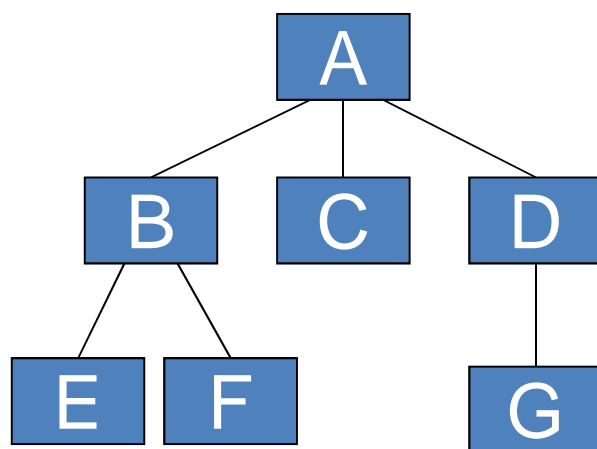
Stratégie testovania

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

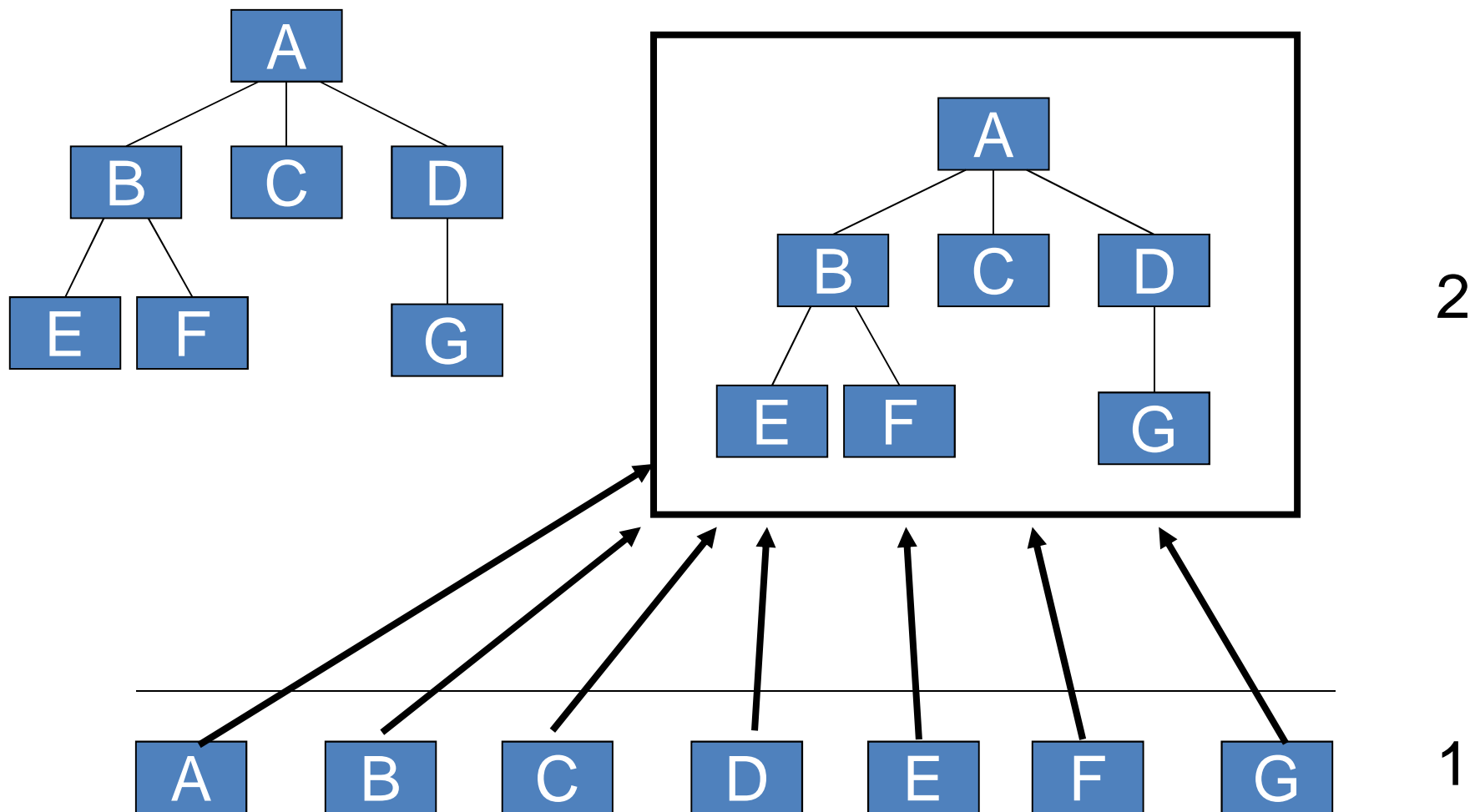
Testovanie zhora nadol



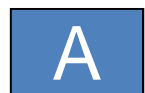
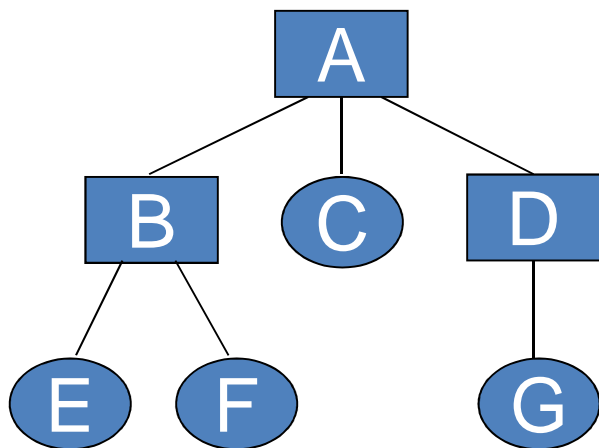
Testovanie zdola nahor



Testovanie jednofázové



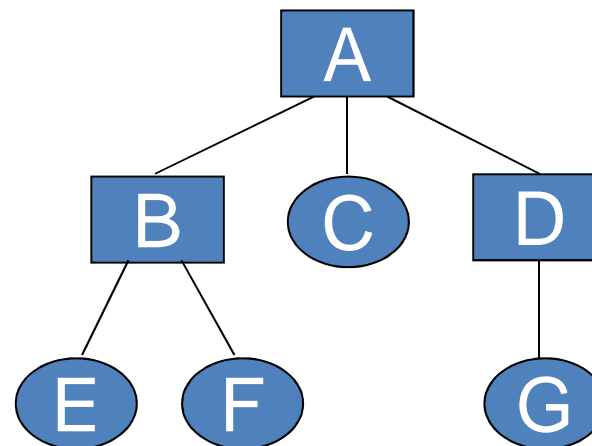
Testovanie sendvičové



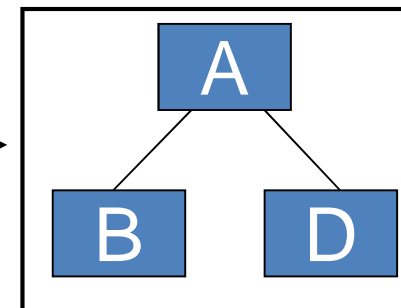
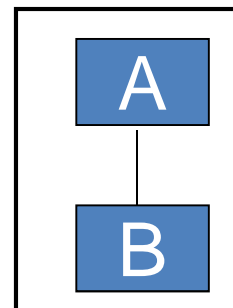
logický modul



funkčný modul



3

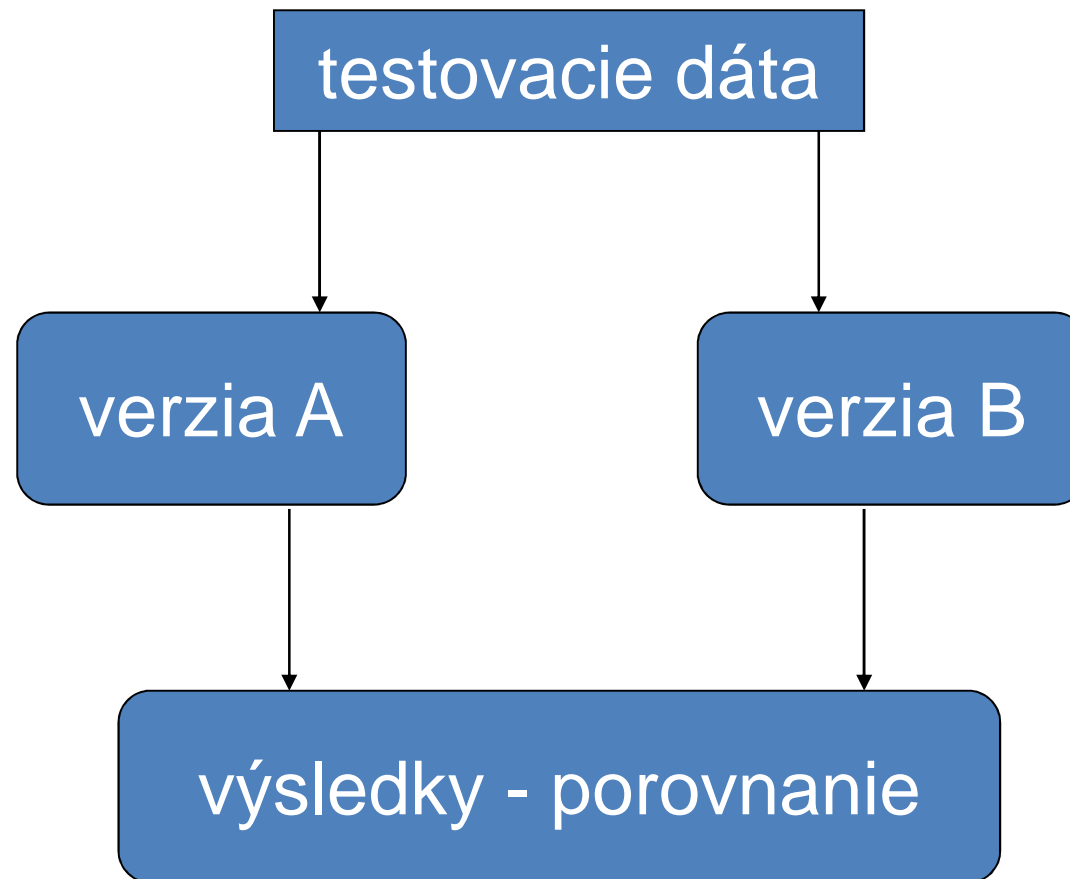


2

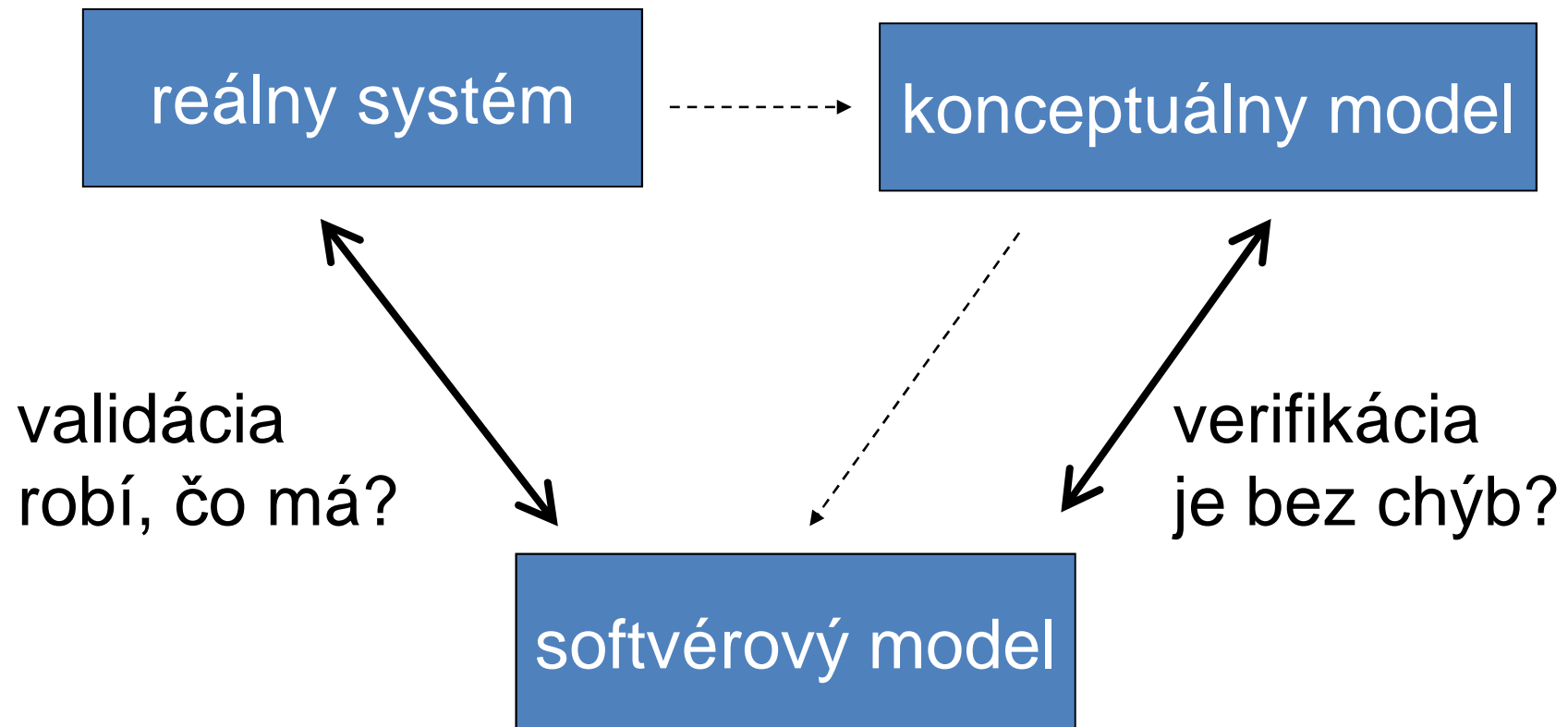


1

Testovanie porovnávacie



Verifikácia - Validácia



Akceptácia

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

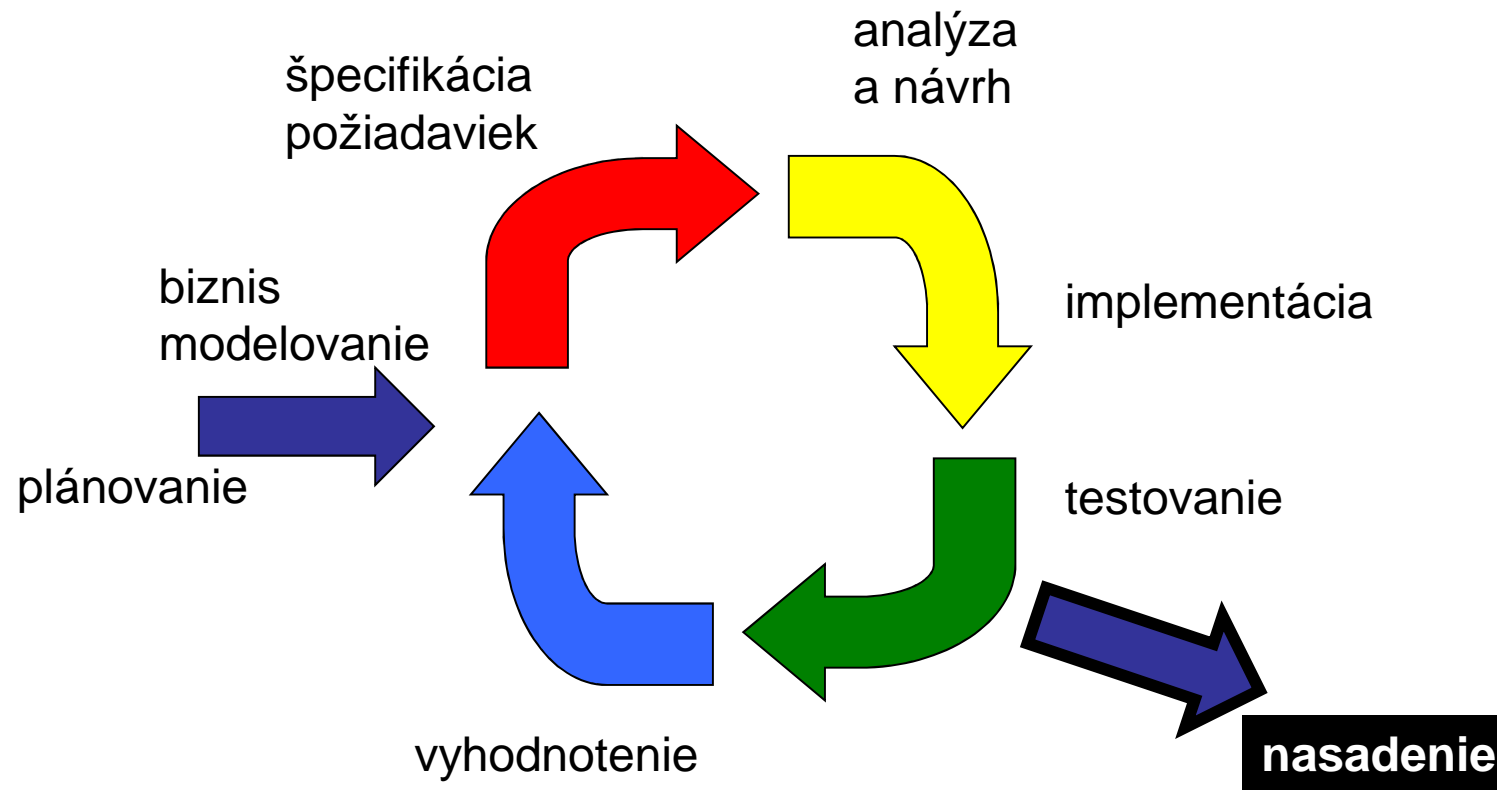
alfa testovanie

- 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



Nasadenie

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

Dokumentácia

- 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

biznis
modelovanie

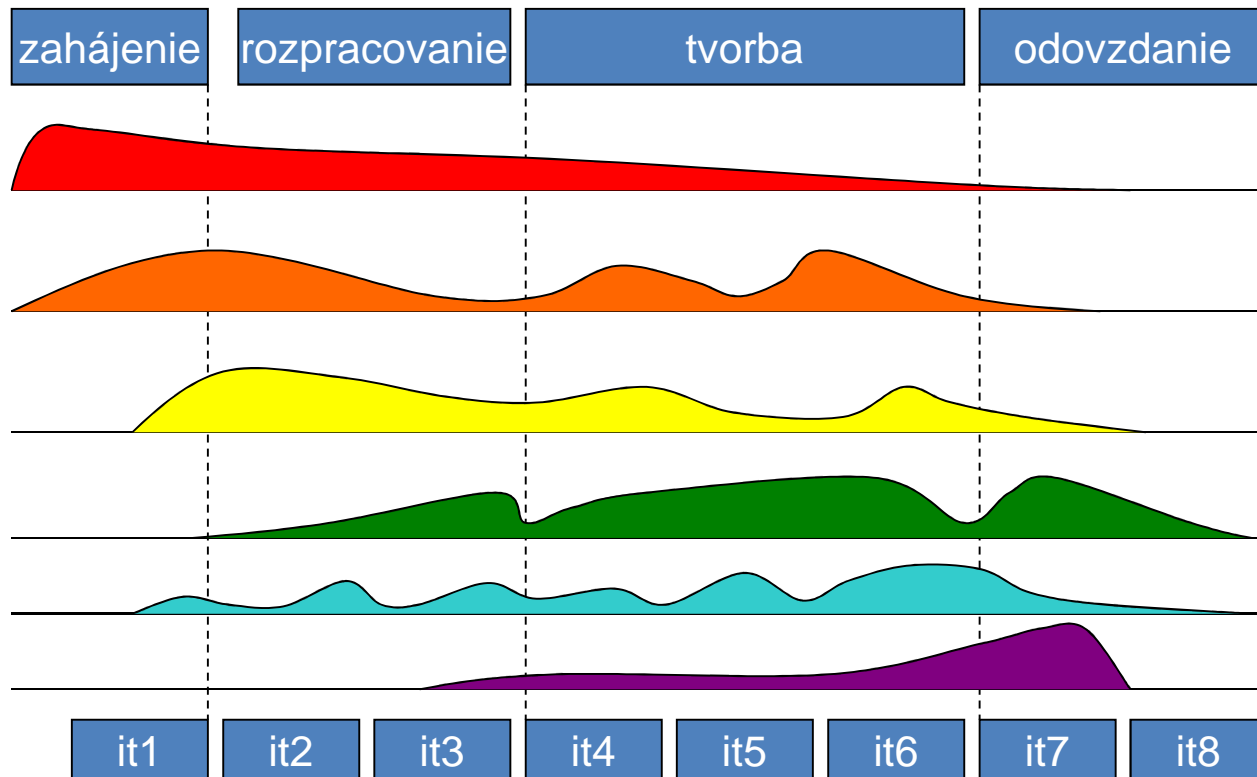
špecifikácia
požiadaviek

analýza
a návrh

implementácia

testovanie

nasadenie



iterácie

Ďakujem za pozornosť