Softvérové inžinierstvo

1

Úvod do softvérového inžinierstva

(5BS03)

Ľubomír Sadloň

Marek Tavač

Ján Ružbarský

KST Katedra softvérových technológií

Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Softvérové inžinierstvo (SI)?

- Inžinierstvo – praktická aplikácia teórie, metód a nástrojov pri návrhu strojov, mostov a pod.

Softvérové inžinierstvo

  - ***disciplína, ktorá sa zaoberá tvorbou rozsiahlych softvérových systémov***
  - aplikácia inžinierskych metód na softvér, zaoberá sa všetkými aspektmi tvorby softvéru
  - aplikácia systematického, disciplinovaného, merateľného prístupu na vývoj a údržbu softvéru.

Rozdiel medzi SI a informatikou

- Informatika sa zoberá algoritmami, spôsobom práce počítačov a softvérových systémov (exaktný popis)
- Softvérové inžinierstvo rieši praktické problémy tvorby softvéru (je nutné používať ad hoc metódy)

Ciele

- Inžiniersky prístup k tvorbe systémov
- Ukázať rôzne aspekty tvorby systémov
- UML
- Práca v tíme
- Objektový prístup
- Enterprise Architect

- Nie fyzické programovanie

Manažment


Projekt mana-žment
Pohľad z druhej strany
Ciele
Analýza systémov
Tvorba dokumen-tácie

Informatika

Aplikačný blok podpory nasadzovania a údržby databázových aplikácií

Cieľ: Analýza, návrh a implementácia aplikačného bloku na podporu nasadzovania a údržby databázových aplikácií.

Obsah: Analýza procesov nasadzovania aplikácií z pohľadu zabezpečenia konzistencie databáz s vývojovým cyklom aplikácií.

Na základe analýzy návrh univerzálneho aplikačného bloku za účelom automatizácie týchto procesov spĺňajúceho nasledujúce požiadavky:

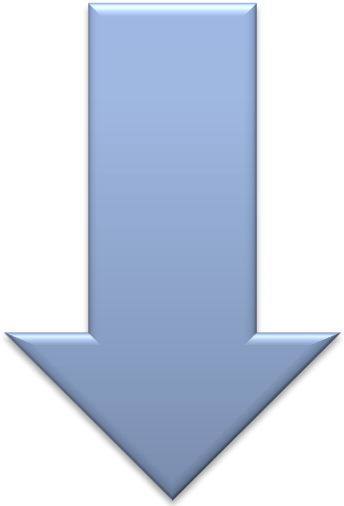
- Identifikácia a riešenie nekonzistencií
- Podpora rôznych systémov verzionovania
- Podpora integrácie používateľského rozhrania
- Nezávislosť na type relačnej databázy
- Univerzálnosť, rozširovateľnosť, konfigurovateľnosť

Funkčnosť a použiteľnosť navrhnutého riešenia preukázať ukážkovou aplikáciou.

Obsah

- Úvod do SI
- RUP procesy
- Biznis modelovanie
- Zber požiadaviek
- Agilné metodiky
- Analýza
- Testovanie
- Návrh
- Implementácia
- Nasadzovanie

Manažment



Informatika

Základné pojmy

- **Softvér**
  - *programy + dokumentácia + konfiguračné dáta*
- **Softvérový systém**
  - *pozostáva z niekoľkých programov, konfiguračných súborov, systémovej dokumentácie (štruktúra systému) a užívateľskej dokumentácie (použitie systému)*
- **Softvérový produkt**
  - *softvér, ktorý sa dá predať zákazníkovi*

História

- Prvé počítače programovali jednotlivci alebo malé tímy: jazyky Fortran alebo assembler
- Počítače III. generácie: možnosti nových aplikácií, ktoré boli oveľa rozsiahlejšie než predchádzajúce systémy
  - Dôležité systémy roky meškali, predraženie projektov → softvérová kríza
  - Spôsoby vývoja malých SW projektov sa nedali použiť pre vývoj veľkých systémov
  - 1968 – konferencia NATO o softvérovej kríze – vznikol termín softvérové inžinierstvo

História - prístupy

- Založené na vstupoch a výstupoch
- Založené na životnom cykle projektu

Štruktúrovaná analýza a návrh

  - ***Metodológia založená na dekompozícii procesov a diagramov toku dát***

- Dátovo orientované

Objektovo orientovaná analýza a návrh

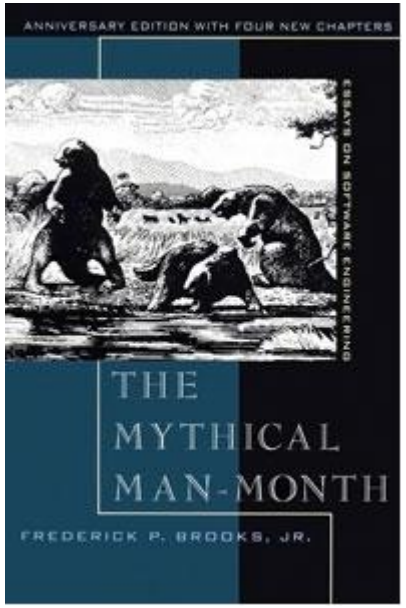
  - ***Analýza a návrh založená na notácii objektov, ktoré zachytávajú dáta a procesy v jednom***

Štruktúrovaná vs objektová

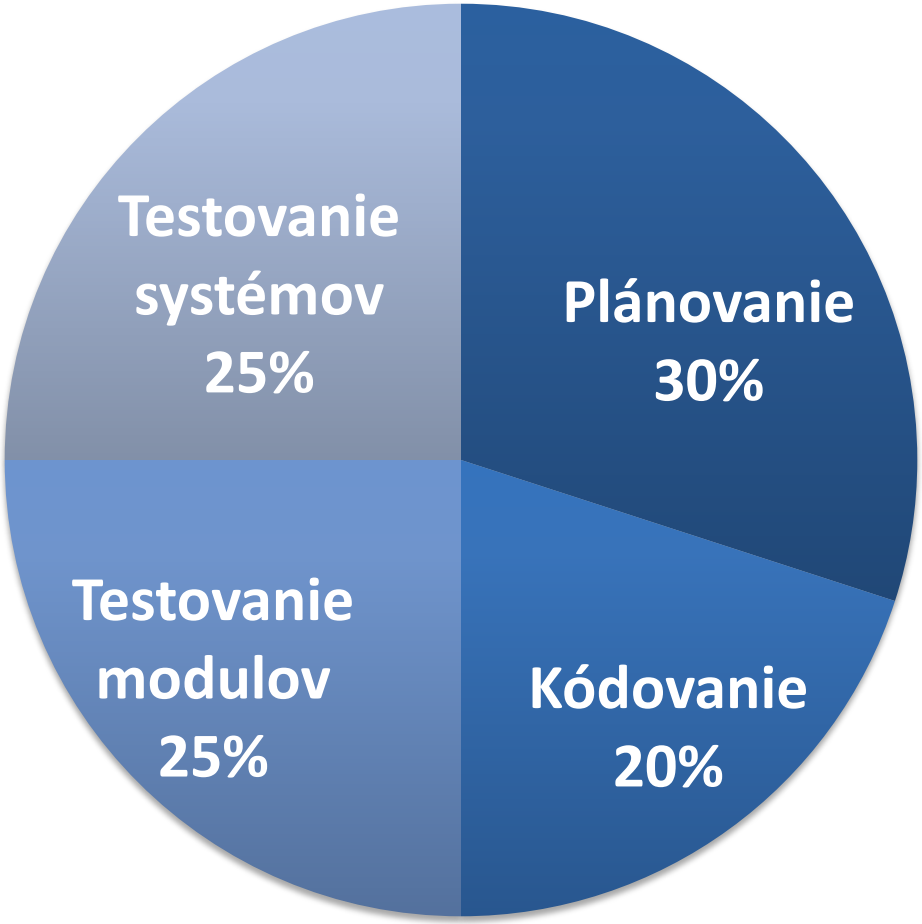
Charakteristika	Štrukturálna analýza a návrh	Objektovo orientovaná analýza a návrh
Metodológia	Životný cyklus	Iteračná / Inkrementálna
Dôraz	Procesy	Objekty
Riziko	Vysoké	Nízke
Znovu použiteľnosť	Nízka	Vysoká
Vyzretosť	Rozvinutá a rozsiahla	Vyvíjajúca sa
Vhodná na	Dobre definované projekty so stálymi užívateľskými požiadavkami	Rizikové rozsiahle projekty s meniacimi sa užívateľskými požiadavkami

Brooks: „The Mythical Man Month“

- Prečo je ťažké vytvárať veľké softvérové systémy
- Programátor je schopný napísať iba 1000 riadkov odladeného kódu za rok
- Veľké projekty sú iné ako malé - skúsenosti z malých projektov sa nedajú použiť na veľké.
- Kódovanie je tá najľahšia časť, ťažšie je:
  - Rozdeliť projekt do modulov
  - Zaistiť komunikáciu medzi modulmi
- Ak 1 programátor píše malý program, predstavuje to najjednoduchšiu časť a jeho efektivita je vyššia (20 riadkov denne)



Rozloženie práce pri projekte



Mythical Man Month ?

- Vyjadrenie náročnosti vývoja SW – počet ľudí x čas
- Čas a počet ľudí nie sú zameniteľné
  - Ak projekt trvá 15 ľuďom 2 roky, nie je možné aby 360 ľuďom (15 x 24) trval iba mesiac, resp. 60 ľuďom 6 mesiacov.
- Dôvod:
  - Práca nie je paralelizovateľná
  - Aby sme využili veľký počet programátorov, musíme rozdeliť projekt na veľa malých častí
  - Ladenie a testovanie systému sa dá ťažko paralelizovať
- Brooksov zákon:
  - **Ak pridáme ľudí k omeškanému projektu, projekt ešte viac omeška.**

Inžiniersky prístup – stavby

- Najskôr podrobný plán – model
- Normy a technologické postupy
- Výroba produktu podľa plánu
- Špecializácia – architekt a staviteľ
- Architekt vie urobiť plány
- Staviteľ vie plány čítať a stavať podľa nich

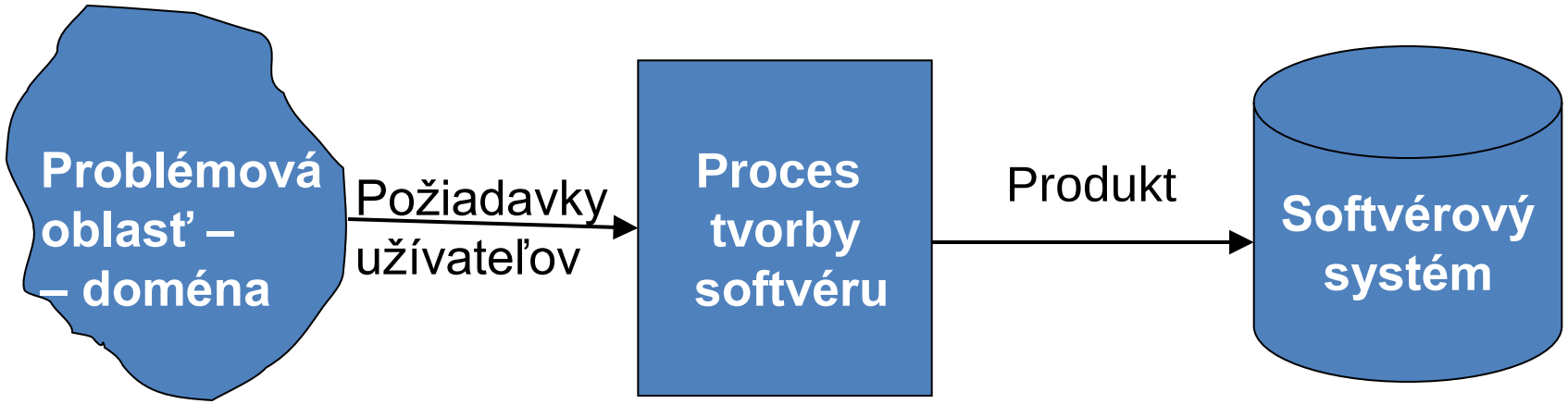
Tvorba softvéru - problémy

- Tvorba bez plánu, resp. iba hrubý náčrt
- Normy a technologické postupy neexistujú ako uzákonený štandard
- Pôvodné ciele sa menia v priebehu tvorby softvéru
- Kumulácia profesií
- Spoločný jazyk sa hľadá – nádejný – UML

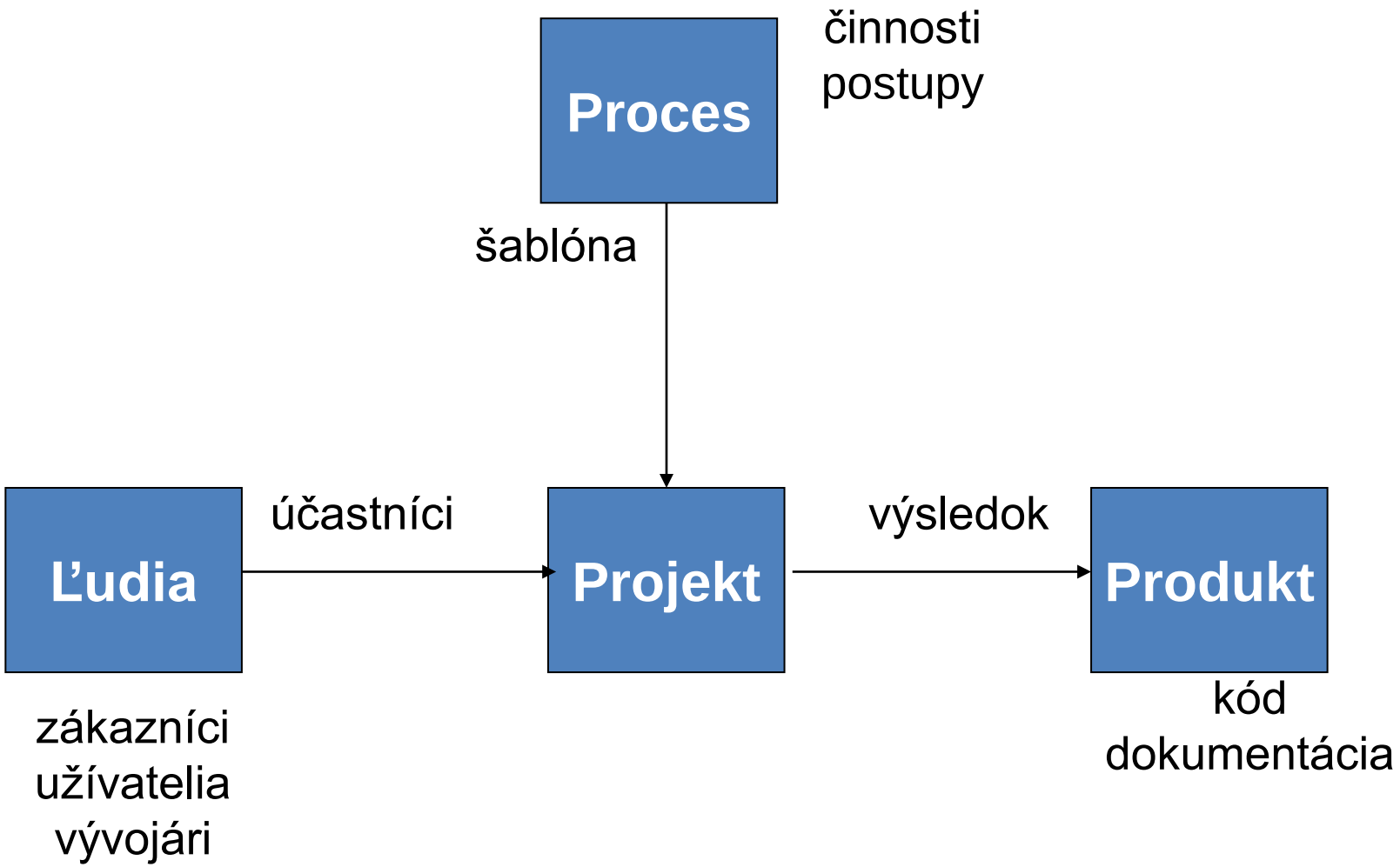
Skúsenosti

- Pre projekt je dôležité mať skúsených vývojárov
- Brooks – väčšina chýb nie je v kóde ale v samotnom návrhu
- Minimálna skúsenosť tímu je nebezpečná – „efekt druhého systému“
  - Prvý produkt týmu – minimálny (dôležité že pracuje, členovia týmu sú spokojní)
  - Druhý produkt týmu – implementované to čo sa pri prvom produkte vynechalo
  - Výsledok – druhý systém je veľký a nevýkonný
  - Tretí produkt je už v poriadku

Tvorba softvérových systémov



Základné prvky vývoja softvéru



Softvérový proces

Činnosti

- ***Procesy, ktoré napomáhajú zaistiť, že výsledný produkt je správny, kompletný a zrozumiteľný***
- konkrétny návod na vykonanie činnosti

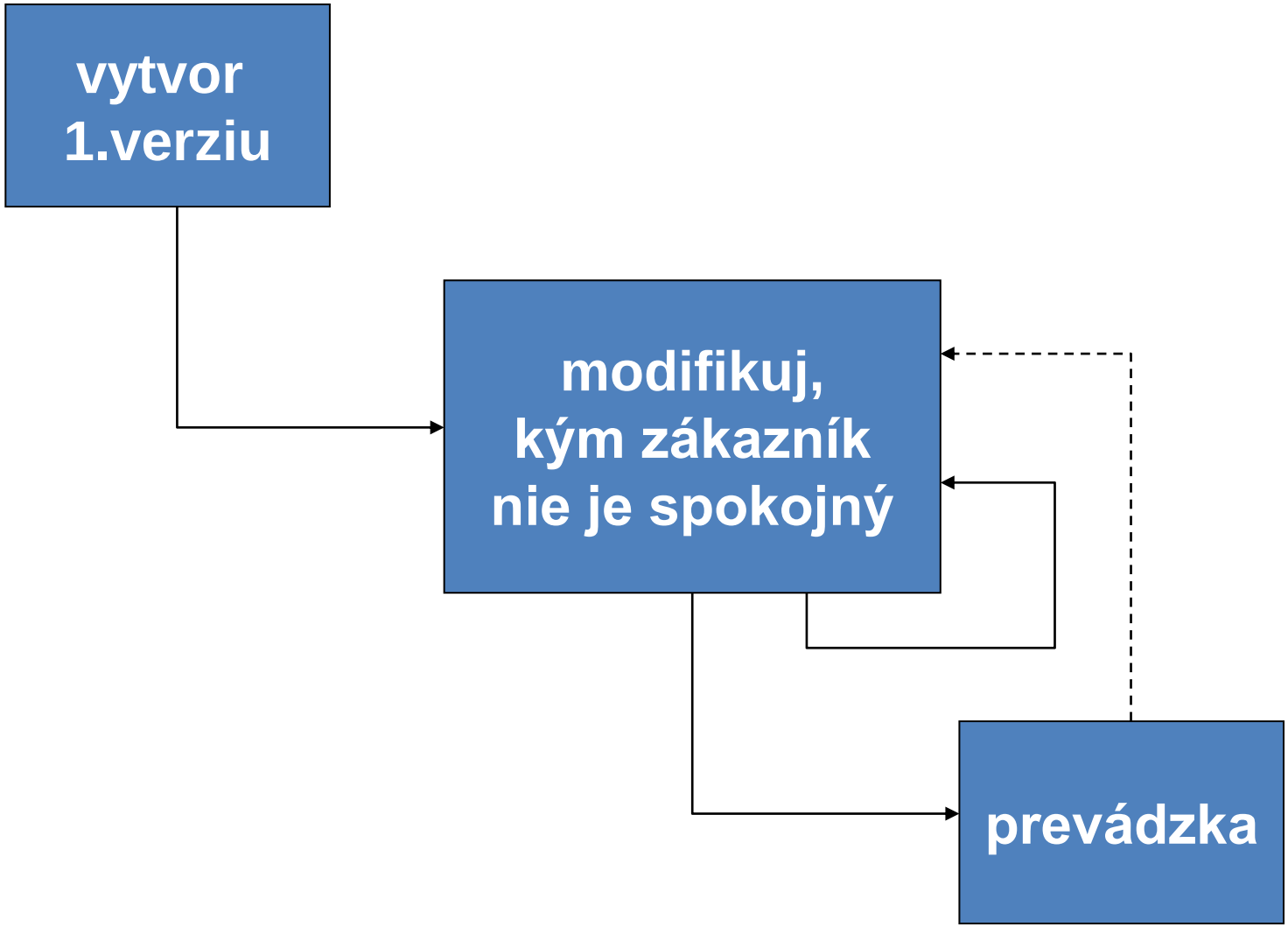
Metodika (pracovný postup)

- ***Sekvencia činností, ktoré napomáhajú pri vývoji finálneho produktu***
- výsledky – produkty činnosti
- riadenie kvality

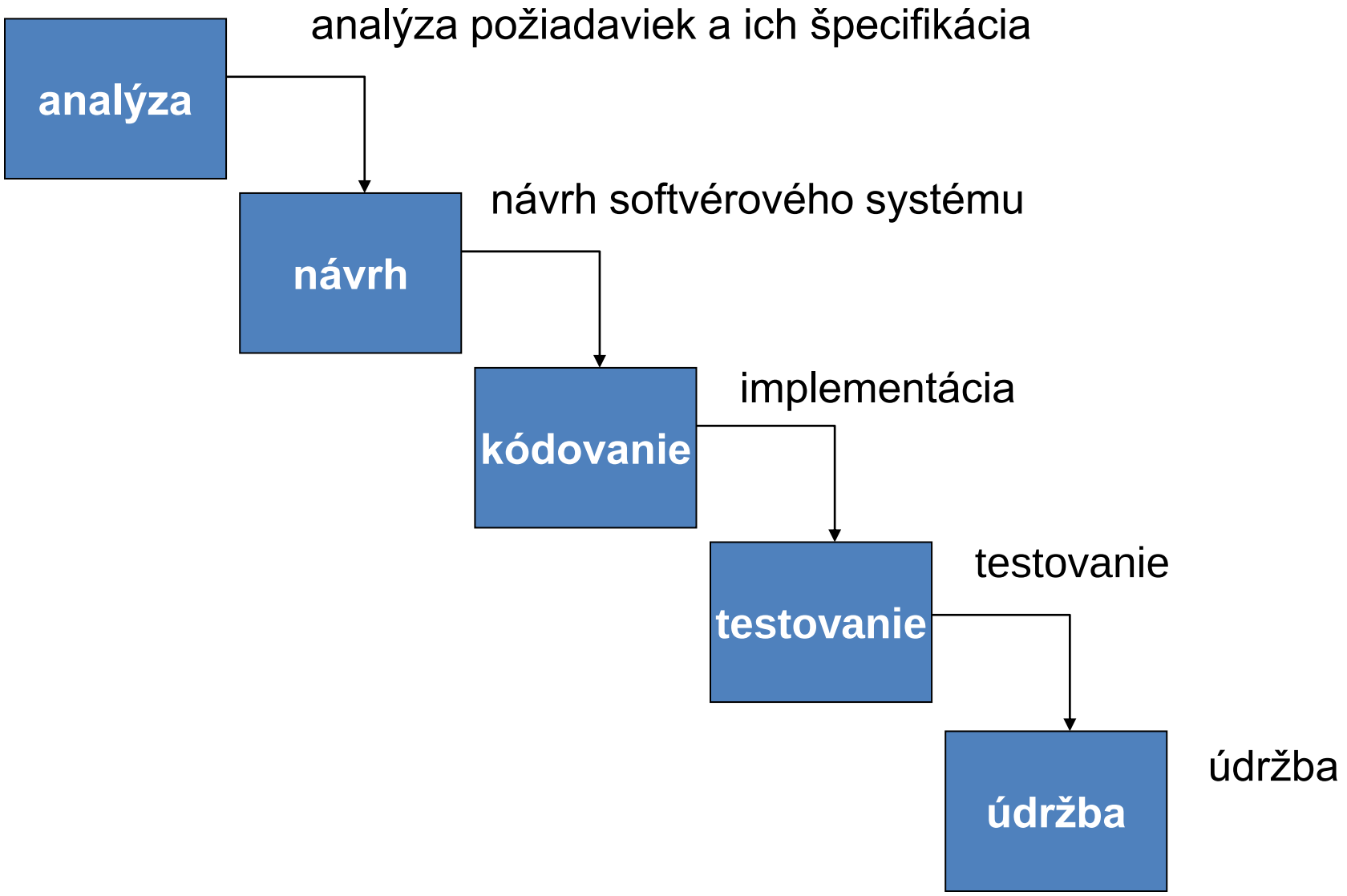
- Nástroje (podpora)

➢ *Životný cyklus projektu*

Životný cyklus – vytvor a oprav



Životný cyklus – vodopádový



Analýza

- Analýza domény, získavanie (zber) a definícia požiadaviek
  - Zoznámenie sa so širším kontextom systému
  - Konzultácia s užívateľmi – zistenie cieľov a služieb systému
  - Ciele a požiadavky sú definované v dokumente špecifikácie požiadaviek

Návrh

- Návrh systému a návrh softvéru
- Rozdelenie požiadaviek na HW a SW, definícia architektúry systému
- Identifikácia a popis základných abstrakcií a ich vzťahov (UML)

Kódovanie a testovanie

- Dizajn je realizovaný ako množina modulov, tried, programov
- Overenie špecifikácií modulov
- Integrácia a testovanie systémov
  - Jednotlivé moduly sú zostavené do výsledného systému
  - Úplný systém je otestovaný na zhodu so špecifikáciou
  - Po otestovaní – odovzdaný zákazníkovi

Údržba

- Najdlhšia fáza životného cyklu – praktické používanie
- Oprava chýb programu a dizajnu + rozširovanie systému

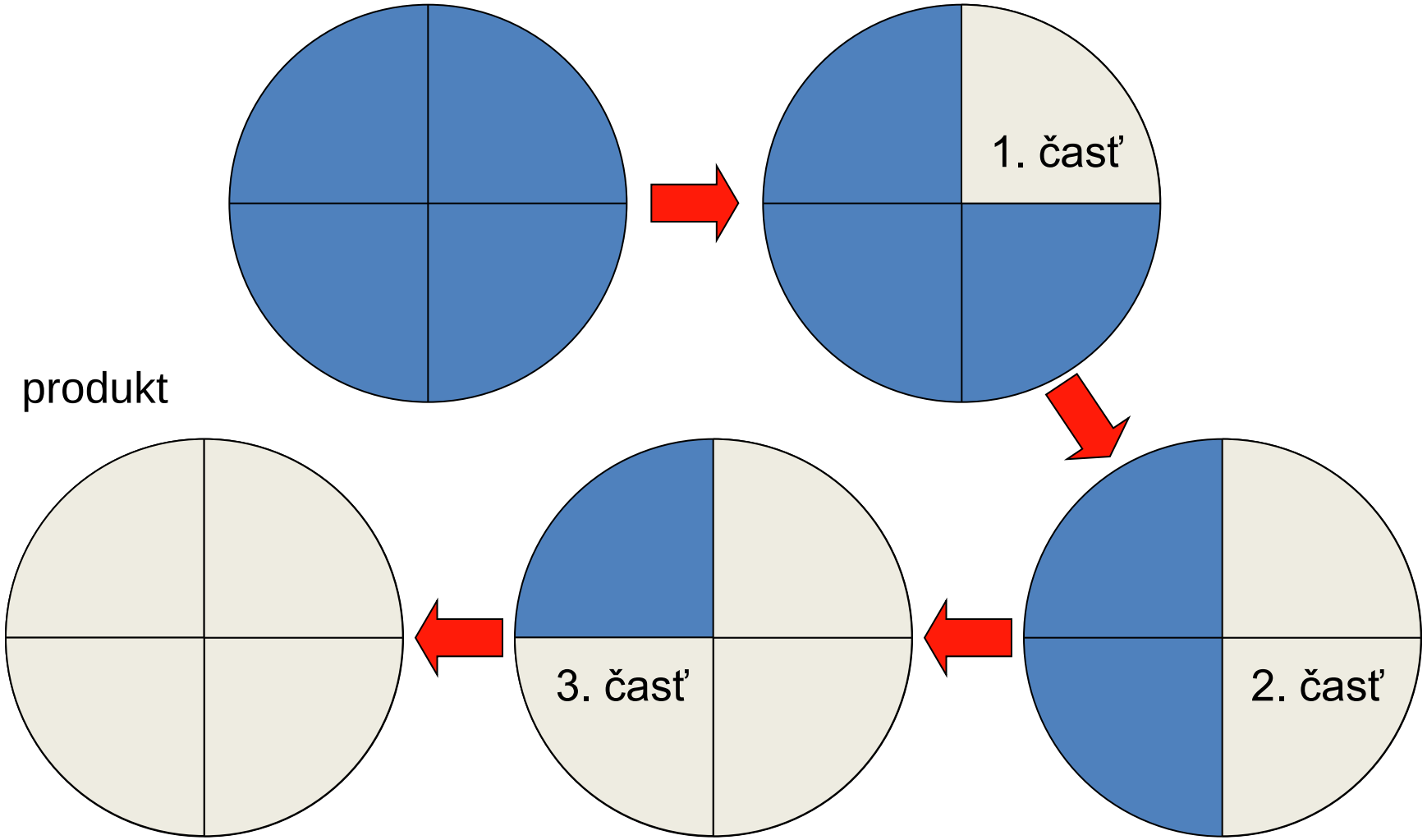
Životný cyklus – vodopádový

- Klady
  - zavádza systém do vývoja softvéru
  - proces sa dá plánovať a kontrolovať
  - vynucuje si zavedenie pravidiel
  - vynucuje si dodržanie termínov
- Nedostatky
  - fázy sa neprekrývajú – sekvenčný postup
  - od zadanie – po hotový produkt – dlhý čas
  - výsledok závisí od analýzy – vhodný ak je problém známy
  - kontrola kvality produktu až po dokončení

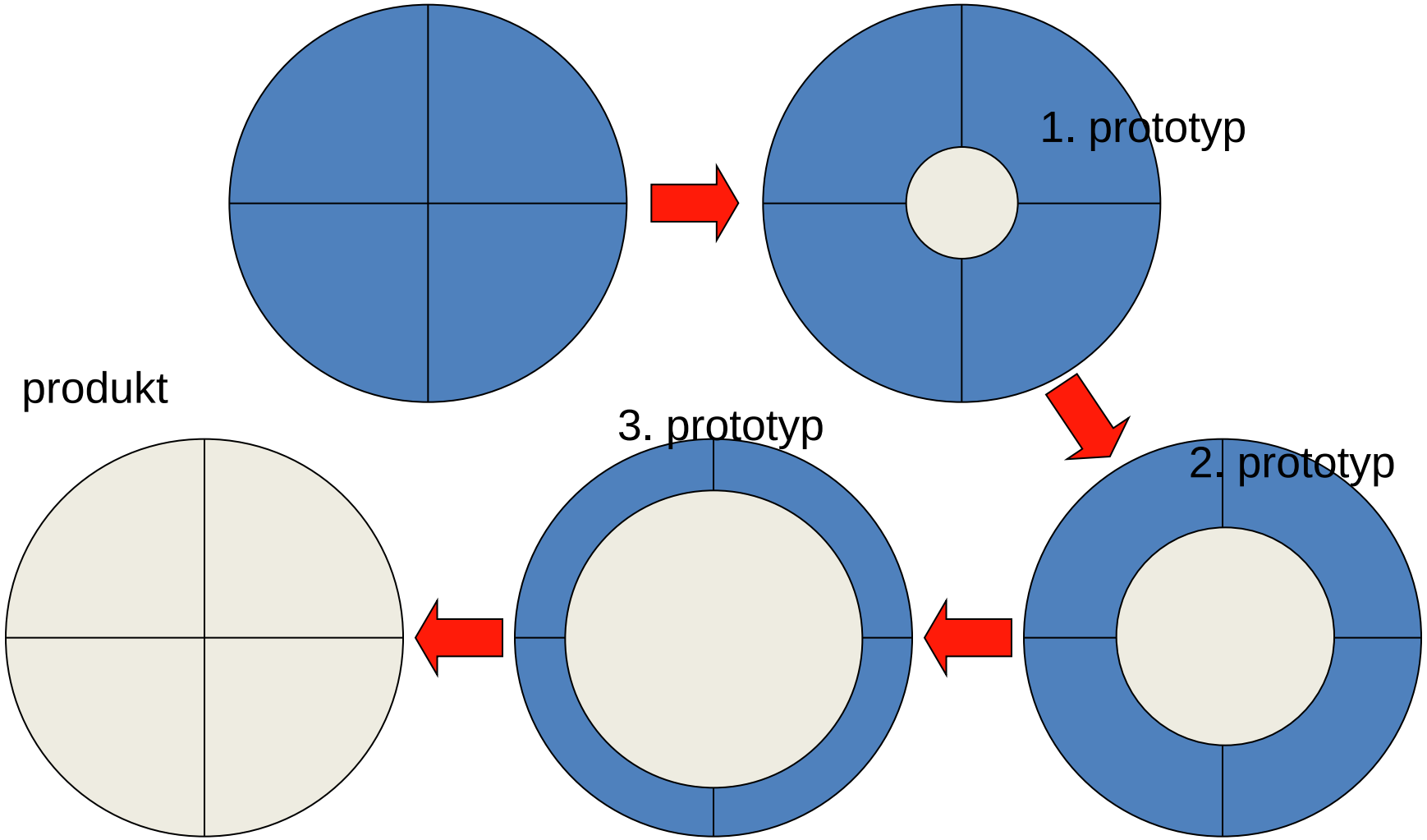
Životný cyklus – vodopádový

- Modifikácie
  - **inkrementálny** model
    - postupné dopĺňanie úplných častí
    - systém menších vodopádov
  - **špirálový** model
    - vytváranie prototypov celého produktu
    - paralelný vývoj všetkých častí

Inkrementálny model



špirálový model



RUP – Rational Unified Process

- výsledok vývoja veľkých softvérových firiem na čele s firmou Rational - IBM
- princípy – najlepšie praktiky tvorby SW
  - iterácie
  - správa požiadaviek
  - komponentová architektúra
  - vizuálne modely
  - overovanie kvality
  - riadenie zmien

RUP – princípy

- **iterácie**
  - chyby sa odhalia v začiatočných návrhoch
  - postupná tvorba systému po častiach tak, aby každá iterácia končila spustiteľným kódom
- **správa požiadaviek**
  - získavanie a dokumentovanie požiadaviek zadávateľa
  - požiadavka – vlastnosť produktu

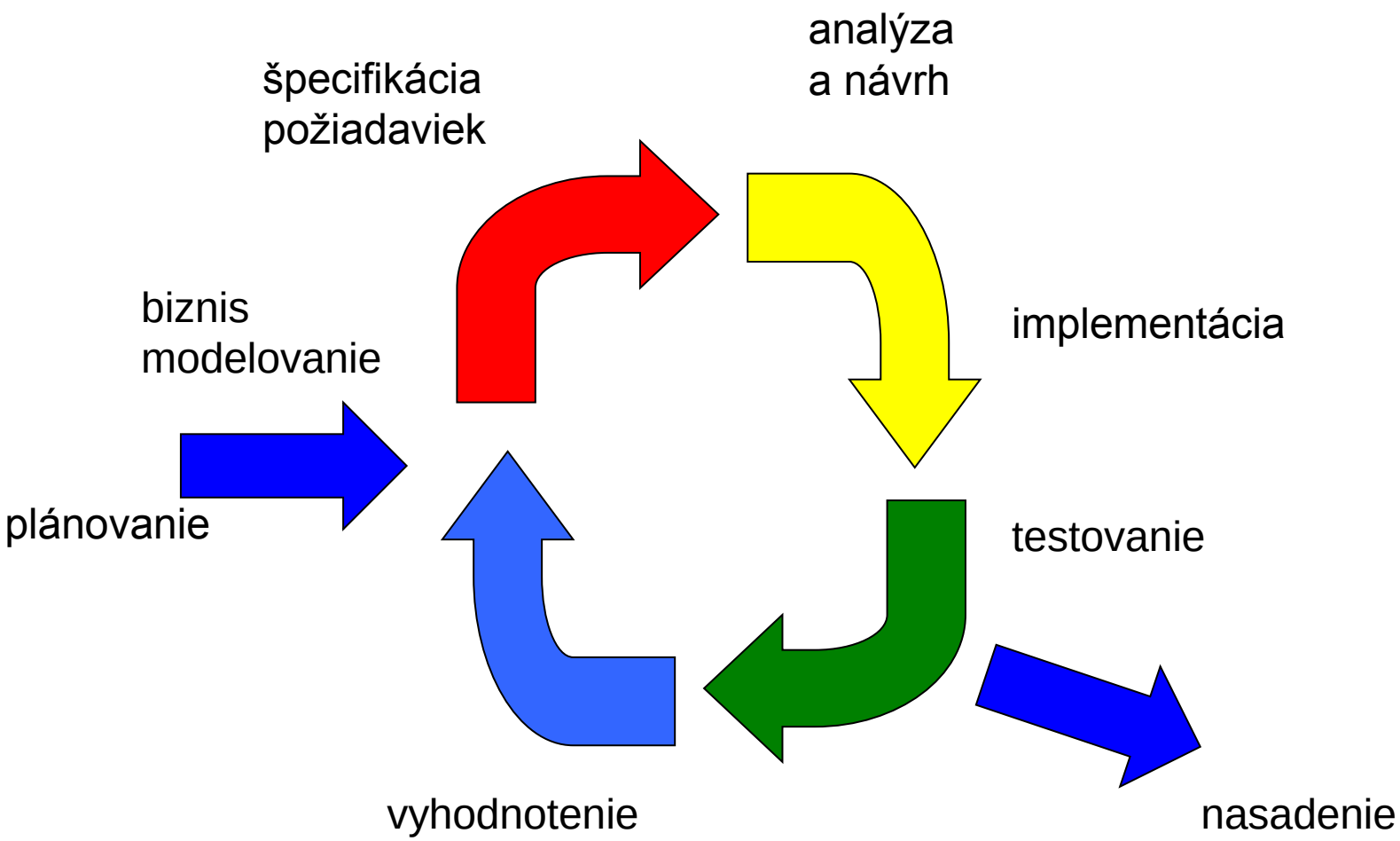
RUP – princípy

- **komponentová architektúra**
  - rozdeľuj a panuj
  - štruktúra prvkov systému
  - využívanie existujúcich a tvorba nových komponentov
- **vizuálne modely**
  - vytváranie a aktualizácia modelov v grafickej podobe – UML
  - vo všetkých fázach tvorby systému
  - zrozumiteľnosť
  - komunikácia: užívateľ - vývojár, vývojár - vývojár

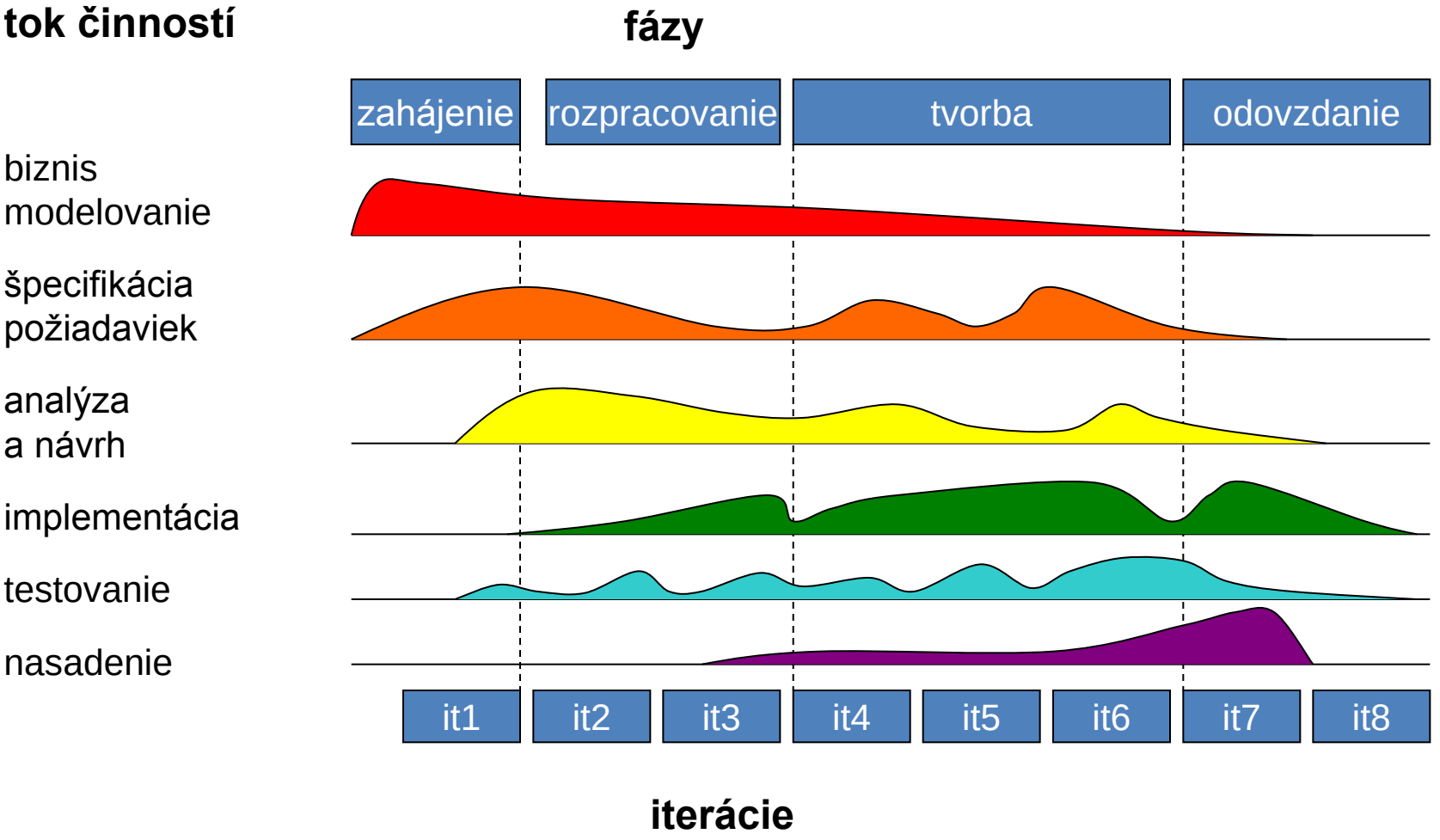
RUP – princípy

- **overovanie kvality**
  - sledovanie kvality pomocou stanovených kritérií počas celého procesu vývoja
- **riadenie zmien**
  - objektívne zmeny v priebehu riešenia – napr. zákony

RUP – iterácia



RUP – schéma (obsah x čas)



RUP – fázy = 1 cyklus (verzia)

- **Zahájenie**

  definícia vízie, určenie rozsahu systému
- **Rozpracovanie**

  návrh architektúry systému
- **Tvorba**

  produkcia, beta verzia systému
- **Odovzdanie**

  produkcia, beta verzia systému, tvorba dokumentácie

Záver

- systémový prístup – softvérový proces
- životný cyklus projektu
- RUP

Ďakujem za pozornosť.