Programmatūras attīstības tehnoloģijas

Dr.sc.ing., asoc. prof. Oksana Ņikiforova
DITF LDI
Lietišķo datorzinātņu katedra
Rīga - LV1048, Meža 1/3, 510.kab., tel.67 08 95 98
oksana.nikiforova@rtu.lv

Programmatūras process

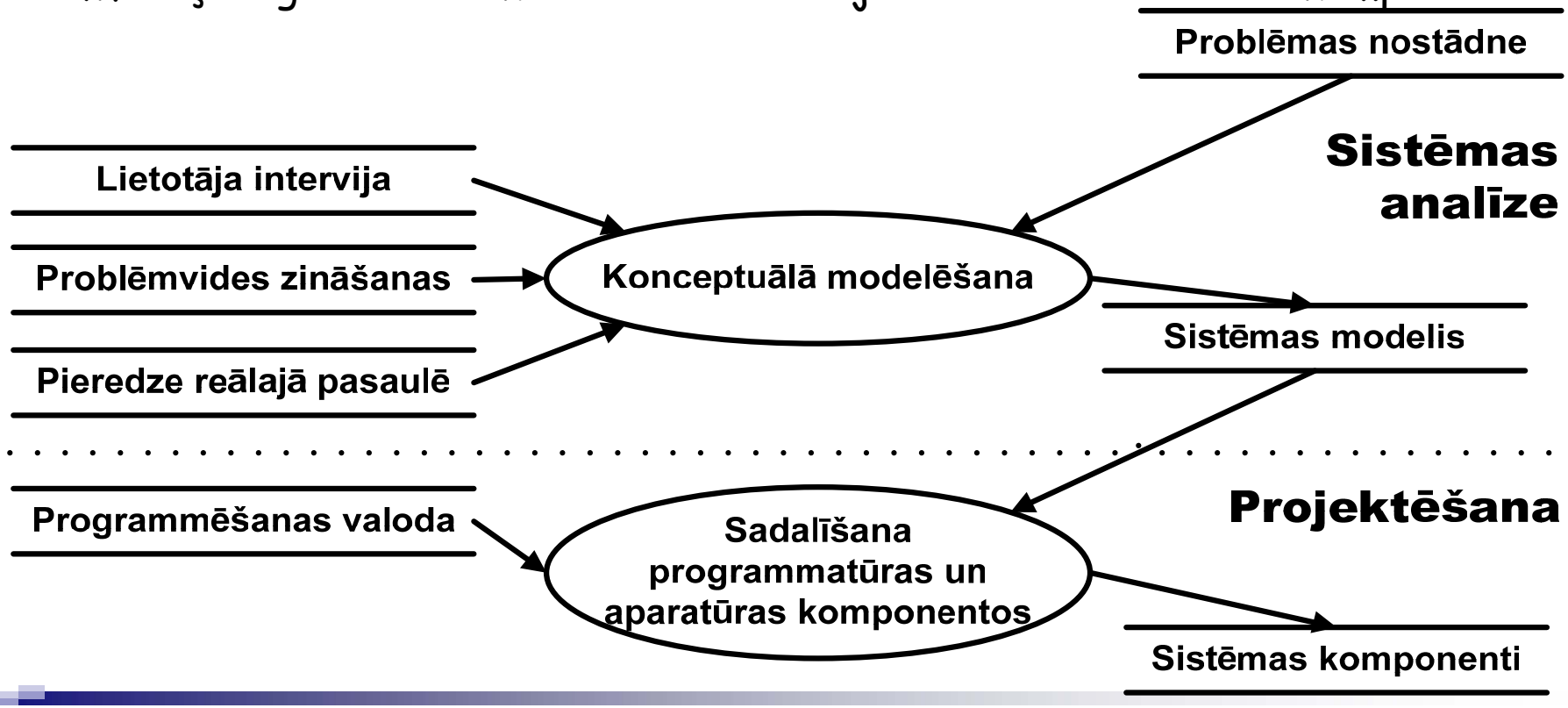
- Programmatūras dzīves cikls
  - Fāžu kopums programmatūras izstrādē un to izpildes secība.
- Dzīves cikla modelis
  - Konkrēta stratēģija programmatūras izstrādē
  - Fāžu sadalījums un to izpildes noteikumi
- Klasiskā fāžu organizācija:
  - Prasību ģenerēšana
  - Analīze & Projektēšana
  - Implementēšana
- "Pēc" fāze:
  - Uzturēšana

Prasību ģenerēšana

- Sistēmas funkcijas, nosacījumi un mērķi ir uzstādīti konsultācijās ar lietotāju.
- Prasības ir definētas prasību specifikācijas dokumentā, kurš ir saprotama gan sistēmas lietotājam, gan izstrādātājam.
- Tiek nostādīts programmatūras izstrādes uzdevums

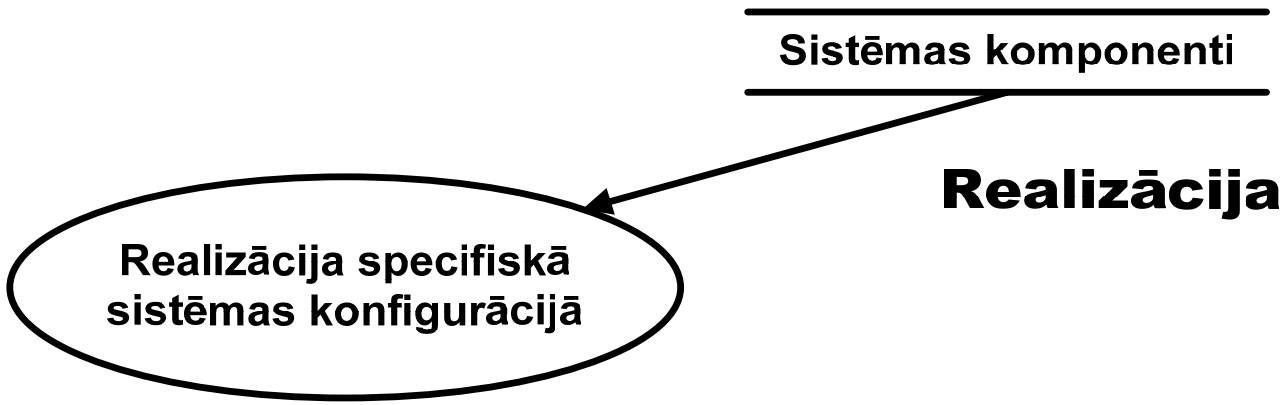
Sistēmanalīze un projektēšana

- Problēmvides saprašana
- Lietojumvides robežu definēšana
- Sistēmas formālā modeļa izstrāde
- Sistēmas modeļa bagātināšana ar realizācijas detaļām un optimizācija
- Modeļa sagatavošana kodēšanai izvēlētajā valodā un sadalīšana komponentos

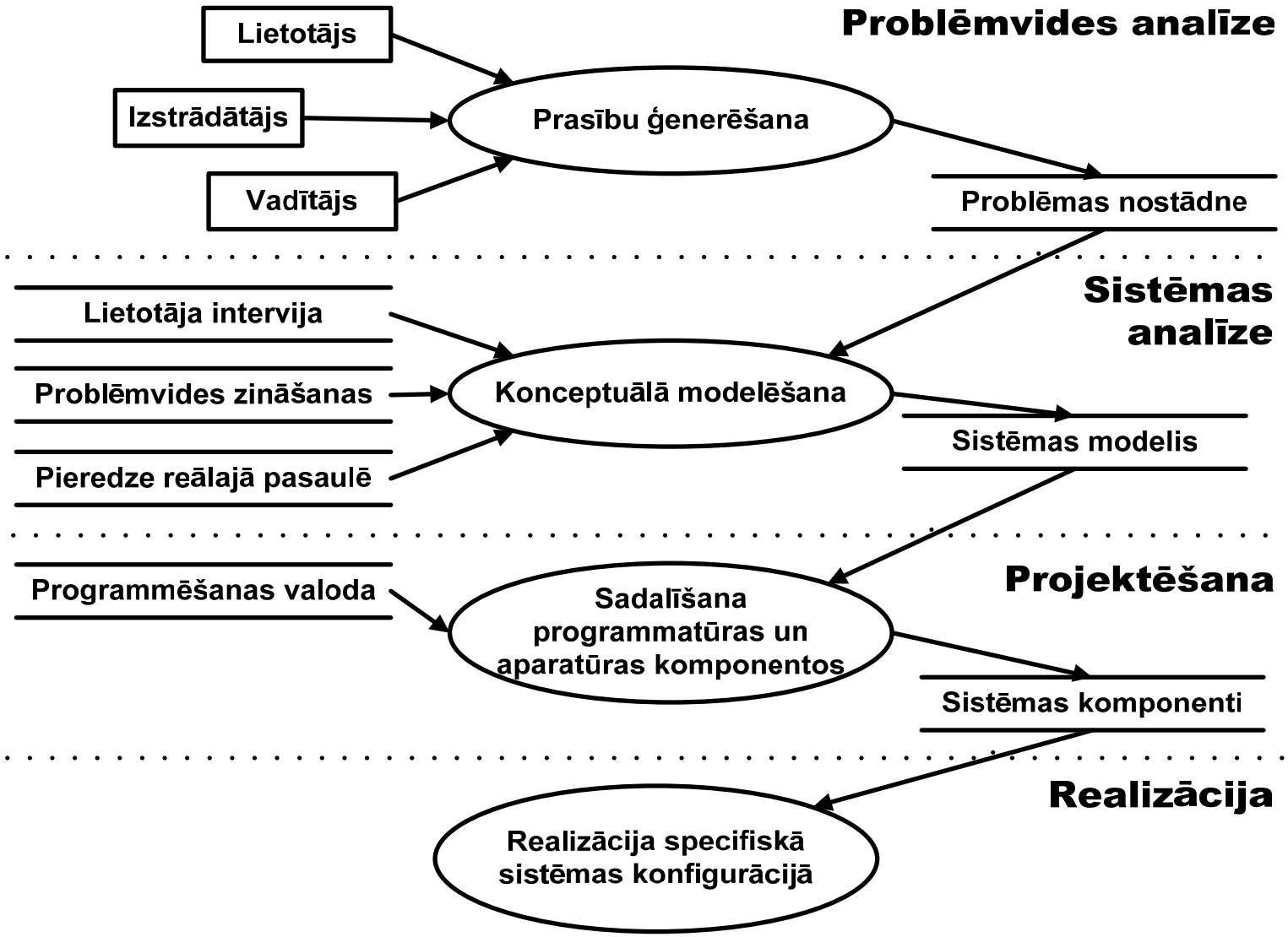


Implementēšana (jeb Realizācija)

- Programmatūras projekts ir realizēts kā programmu vai programmas moduļu kopums specifiskā sistēmas konfigurācijā
- Moduļu testēšana pārbauda vai katrs modulis atbilst tā specifikācijai.
  - verifikācija – pārbauda, vai programma ir uzrakstīta pareizi,
  - validācija – pārbauda, vai ir uzrakstīta pareiza programma – tā programma ko vajadzēja.
- Savienošana (integrācija) un sistēmas testēšana. Individuālās programmas vai moduļi ir savienoti un testēti kopējā sistēmā, lai pārliecinātos ka programmatūras prasības ir apmierinātas.



Programmatūras kodols

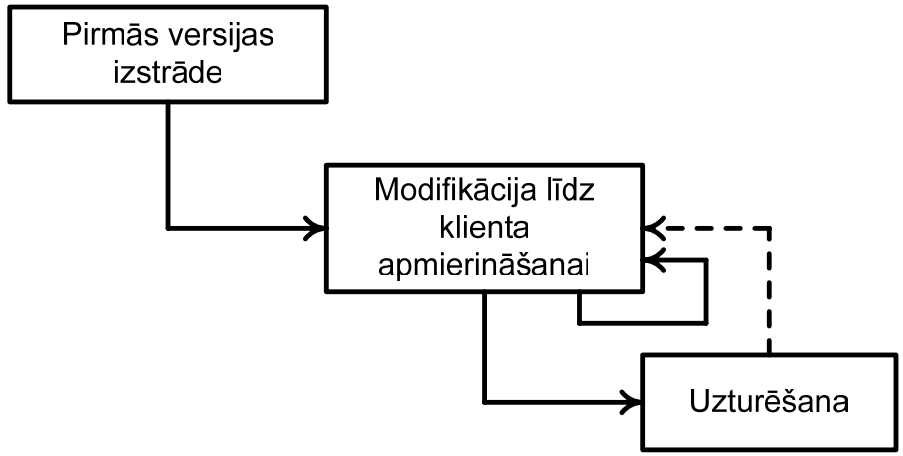


Programmatūras dzīves cikla modelis

- Nodrošinā ieteikumu kopu projekta vadībai
  - Uzdevumu izpildes secība
  - Pārbaudes punkti
  - Projekta attīstība
- Eksistē milzīgs dzīves cikla modeļu skaits un vēl vairākas kompānijas adaptē tos, veidojot pašu modifikācijas.
- Dzīves cikla attīstība:
  - Kodē un fiksē (*code and fix*)
  - "Disciplinētā" izstrāde
  - "Spēja" (*agile*) izstrāde

Kodē un fiksē (code-and-fix)

- Programmatūras izstrāde
  - Bez prasību specifikācijas un projektēšanas
  - Kodē, palaiž un skaties, kas notiek
  - Programmatūras attīstība, kamēr pasūtītājs nav apmierināts
- Pamata problēmas
  - Izmaiņas uzturēšanas gaitā
  - Dārgs lielām un sarežģītām sistēmām



Programmatūras izstrādes metodoloģija

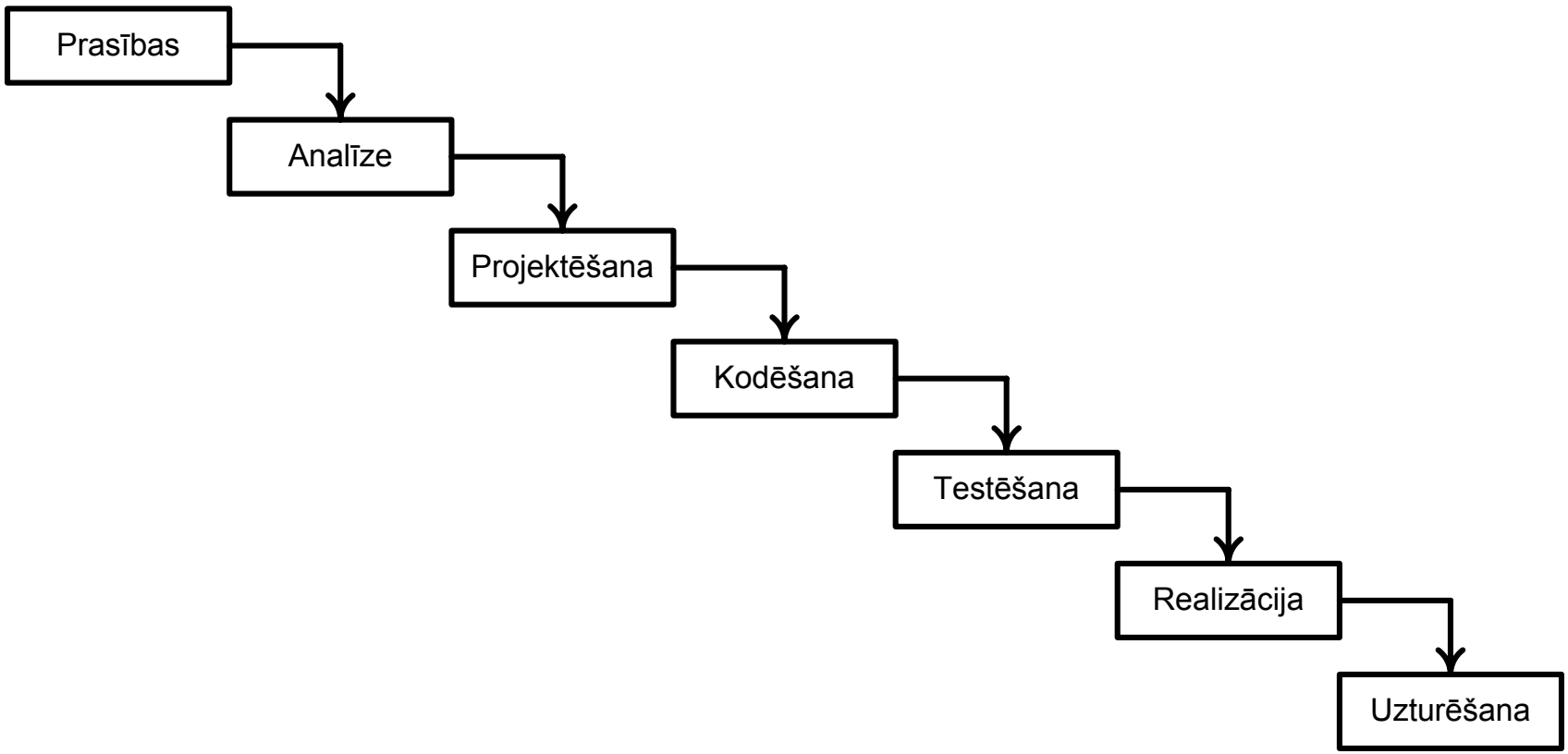
- Programmatūras izstrādes process balstās uz konkrētu programmatūras izstrādes metodoloģiju.
- **Metodoloģija ir organizēts process, kurā tiek lietots iepriekš definēto tehniku kopums un iepriekš definēta apzīmējumu sistēma.**
- Metodoloģija parasti ir attēlota kā soļu secība, kur katram solim atbilst noteikti paņēmieni to izpildīšanai.

Programmatūras izstrādes procesu klasifikācija

- Secīga pieeja – pēc noteiktas dzīves cikla fāzes pabeigšanas nav iespējas atgriezties uz kādu no iepriekšējiem soļiem;
- Iteratīva pieeja – pieeja paredz atgriešanos uz kādu no iepriekšējiem soļiem, tajā kaut ko izmainīt un tad izplatīt šīs izmaiņas tālāk;
- Rekursīva pieeja – izstrādāta soļu secība var būt pielietota katra dzīves cikla soļa iekšā, kā arī var būt atkārtoti pielietota gala produktam.
  - Visas rekursīvās pieejas ir iteratīvas, bet ne visas iteratīvās pieejas ir rekursīvas.

Secīgais "Ūdenskrituma" modelis Royce W.W. - 1970

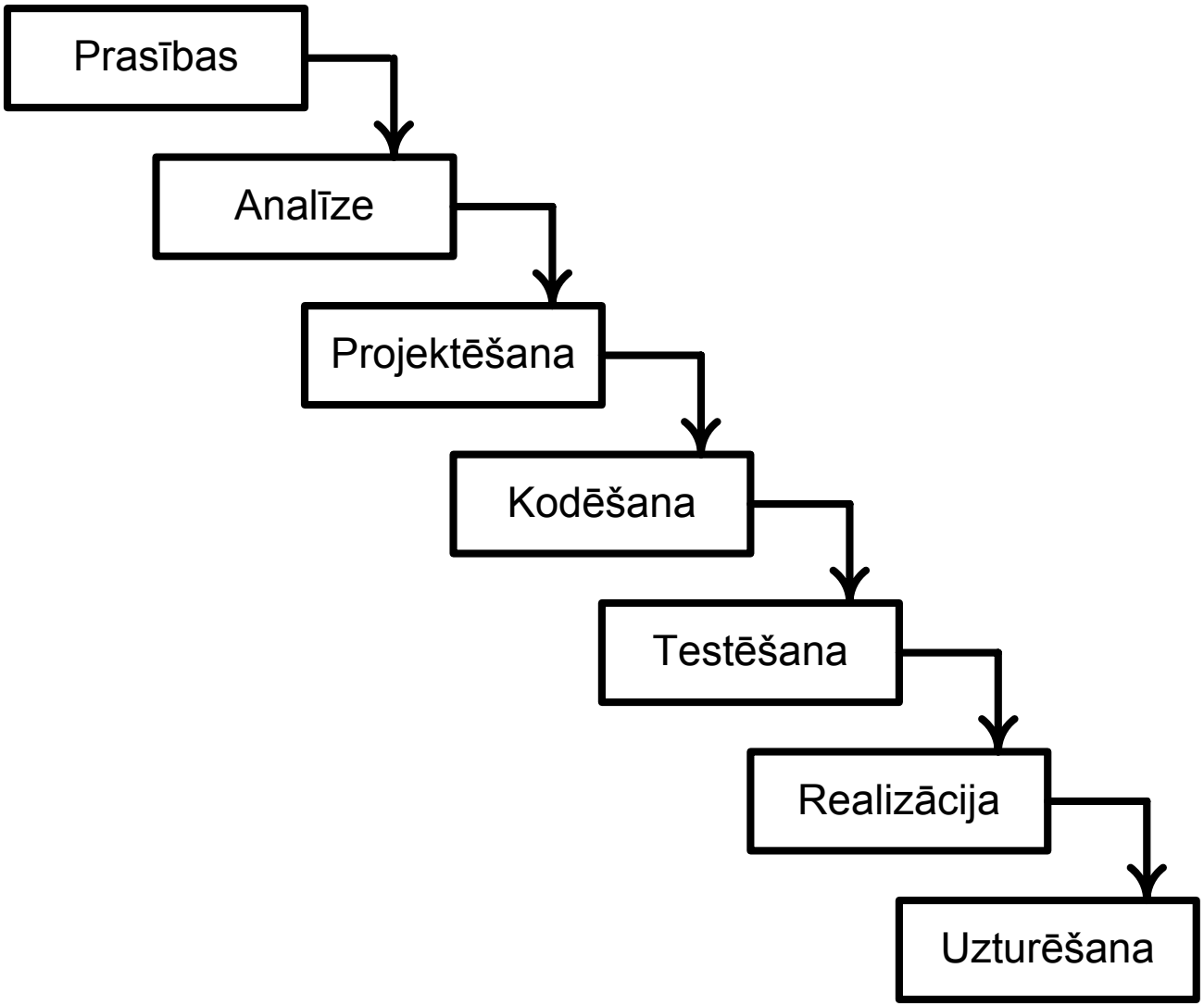
- Fāžu lineāra secība
- Iepriekšējas fāzes beigas pirms nākošās uzsākšanas
- Precīzā katras fāzes rezultātu definēšana



Pilnā "Ūdenskrituma" modeļa fāzes

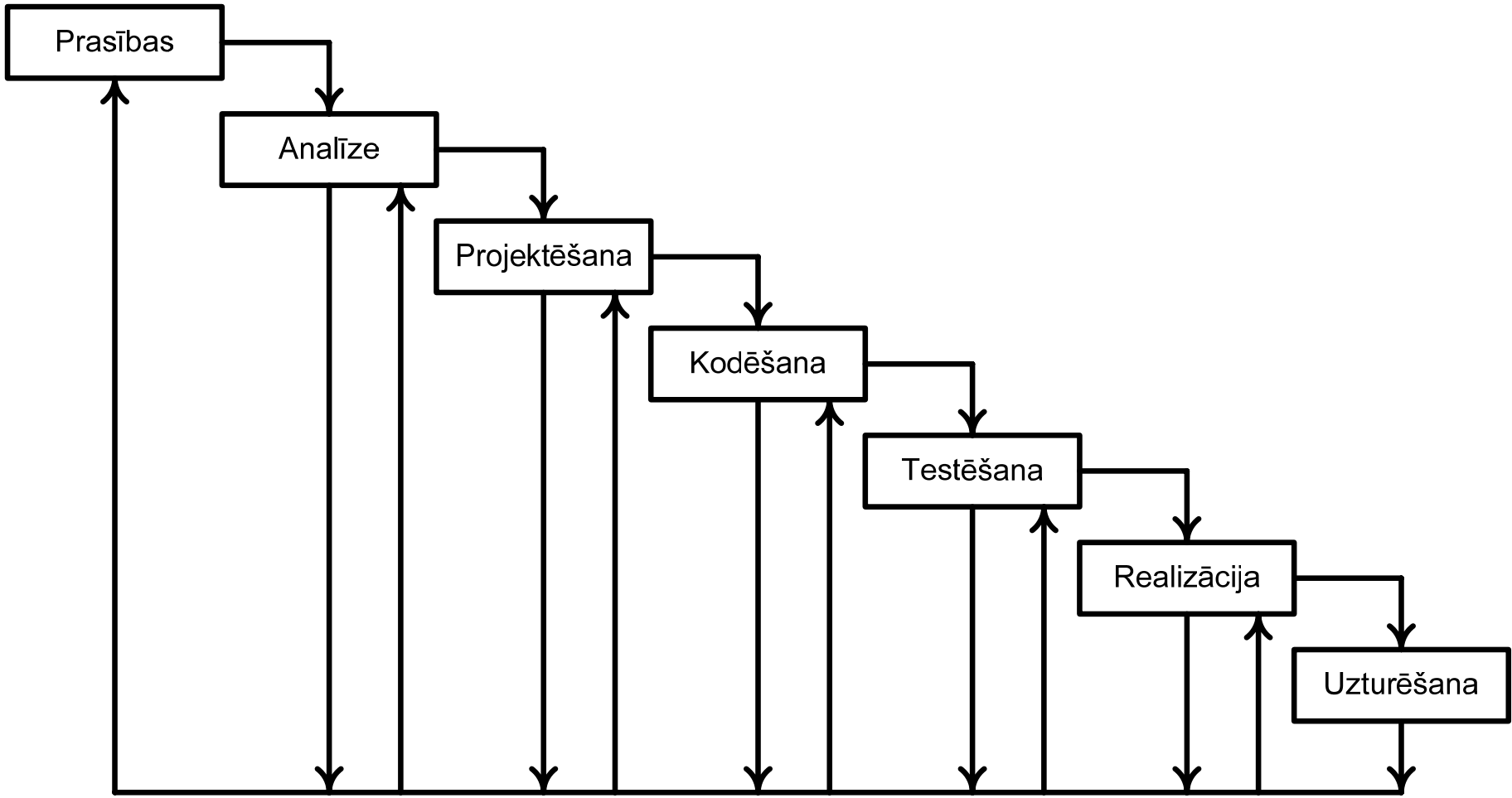
1. Ontoloģija
2. Viedokļu noskaidrošana
3. Mērķu noteikšana
4. Prasību noskaidrošana
5. Modelēšana
6. Sistēmas analīze
7. Projektēšana
8. Kodēšana
9. Testēšana
10. Realizācija
11. Uzturēšana
12. Modificēšana
13. Konservēšana
14. Iznicināšana

"Ūdenskrituma" modeļa modernāka modifikācija

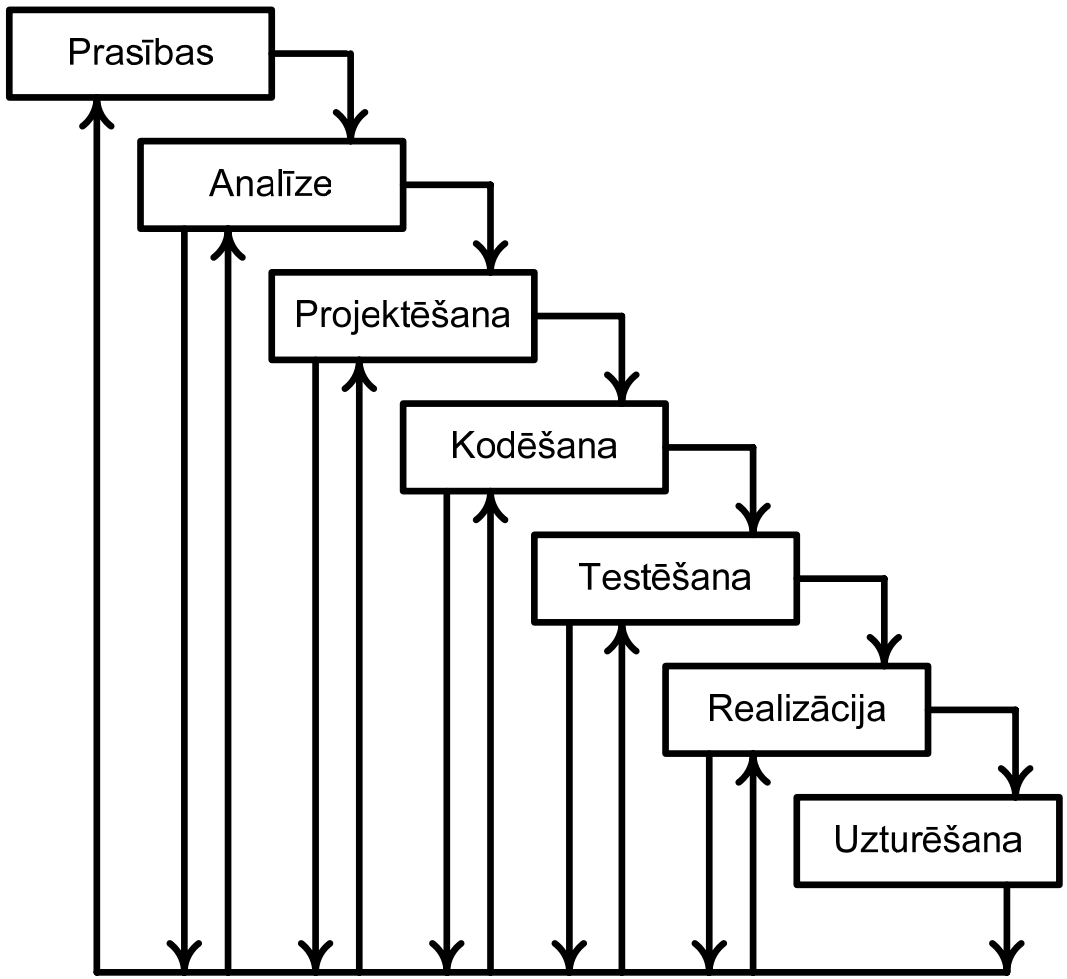


Iteratīvais "Ūdenskrituma" modelis

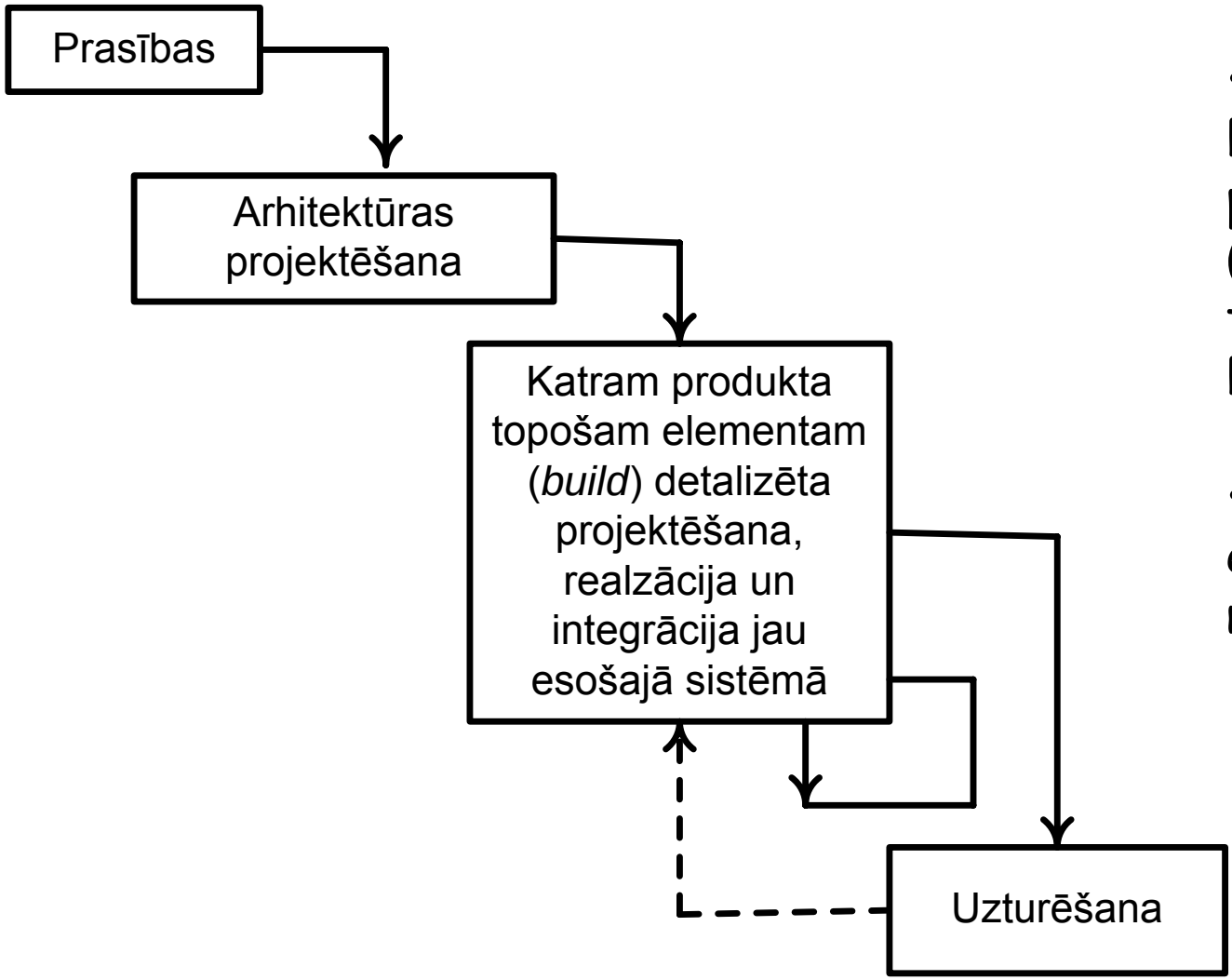
Royce W.W. - 1976



Iteratīvā "ūdenskrituma" modifikācija



Pieaugošais ("*incremental*") modelis



- Katras fāzes beigās ir jauns produkta elements (artefakts), kas ir topoša produkta bagātināšana
- Citādi saukts evolucionārs modelis

Pieaugošā modeļa modifikācija

- Priekšrocība
  - Parallēlitāte ekonomē laiku
- Riski
  - Ja nav kopēja risinājuma projektējuma sākumā → izstrādātie elementi var nesavienoties kopējā sistēmā