Ievads studiju nozarē

4. lekcija

Tēmu saraksts

1) Datorzinātnes un informācijas tehnoloģijas fakultātes misija un struktūra, pedagoģiskās un zinātniskās aktivitātes

2) Studiju saturs, studiju darbs un ārpus studiju aktivitātes

3) Studiju procesa organizācija, studentu tiesības un pienākumi

4) Sistēmu izstrādes projekta uzsākšana

5) Prasību analīzes posms sistēmu izstrādes procesā

6) Projektēšanas un izstrādāšanas posmi sistēmu izstrādes procesā

7) Testēšanas un uzturēšanas posmi sistēmu izstrādes procesā

8) Sistēmu izstrādes projekta noslēgšana

Plāns uz nodarbību

- Programmatūras izstrādes process,
- Līguma izstrāde,
- Projekta vadītāja loma.

PROGRAMMATŪRAS IZSTRĀDES PROCESS

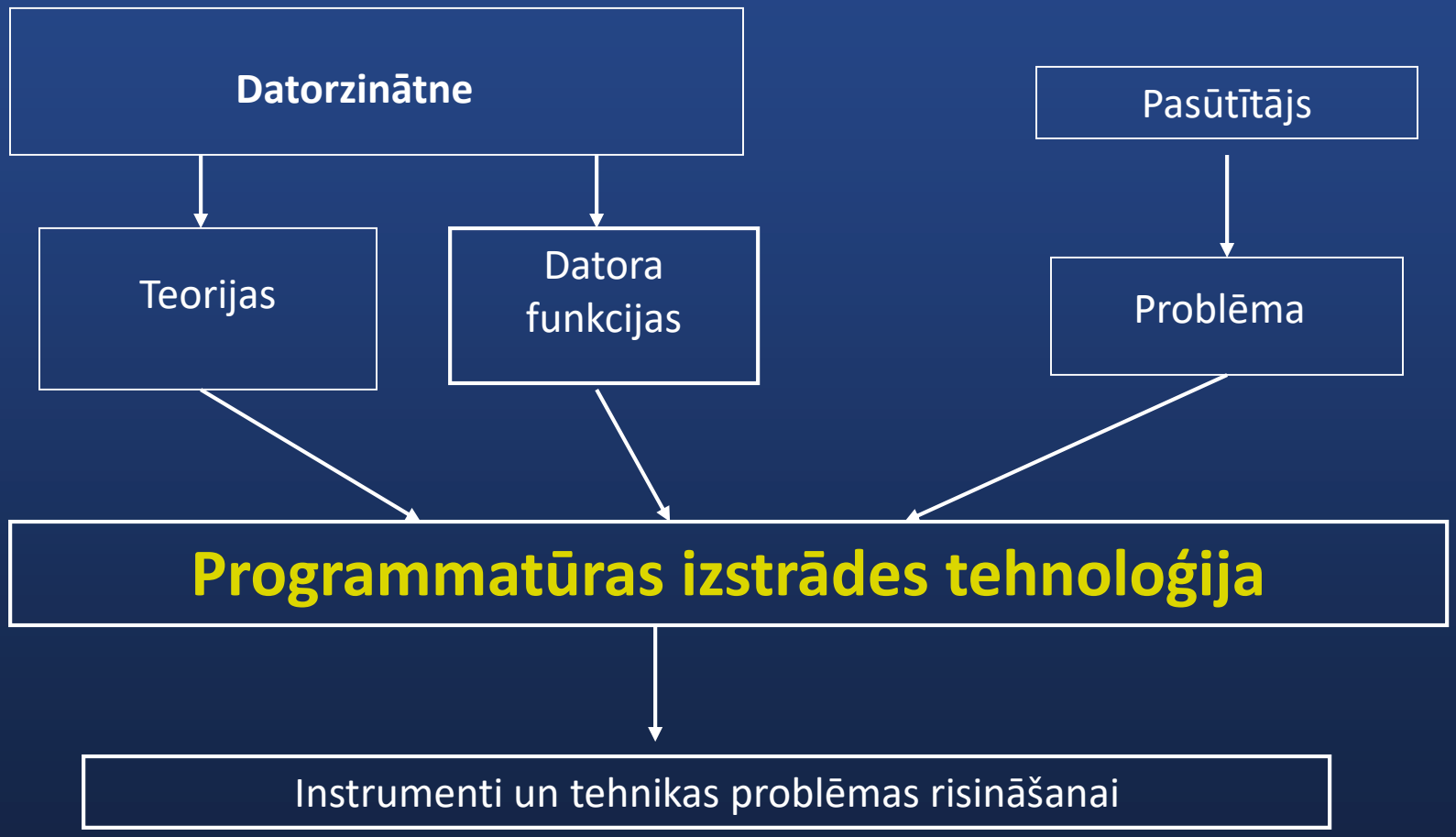
Programmatūras inženierija

- Sistematizēta pieeja zinātnisko un tehnoloģisko zināšanu, metožu un pieredzes izmantošanai funkcionāli efektīvas programmatūras izstrādāšanas, testēšanas un ieviešanas procesā;
- Inženierprincipu izstrādāšana un lietošana, lai taupīgi radītu programmatūru, kas ir droša un efektīvi strādā datoros.

Bāzlīnijas

- Ko nozīmē inženierprincipi, kas jālieto programmatūras izstrādei?
- Kā taupīgi izveidot drošu programmatūru?
- Kas ir nepieciešams programmu radīšanai, kuras efektīvi strādā dažādos datoros?

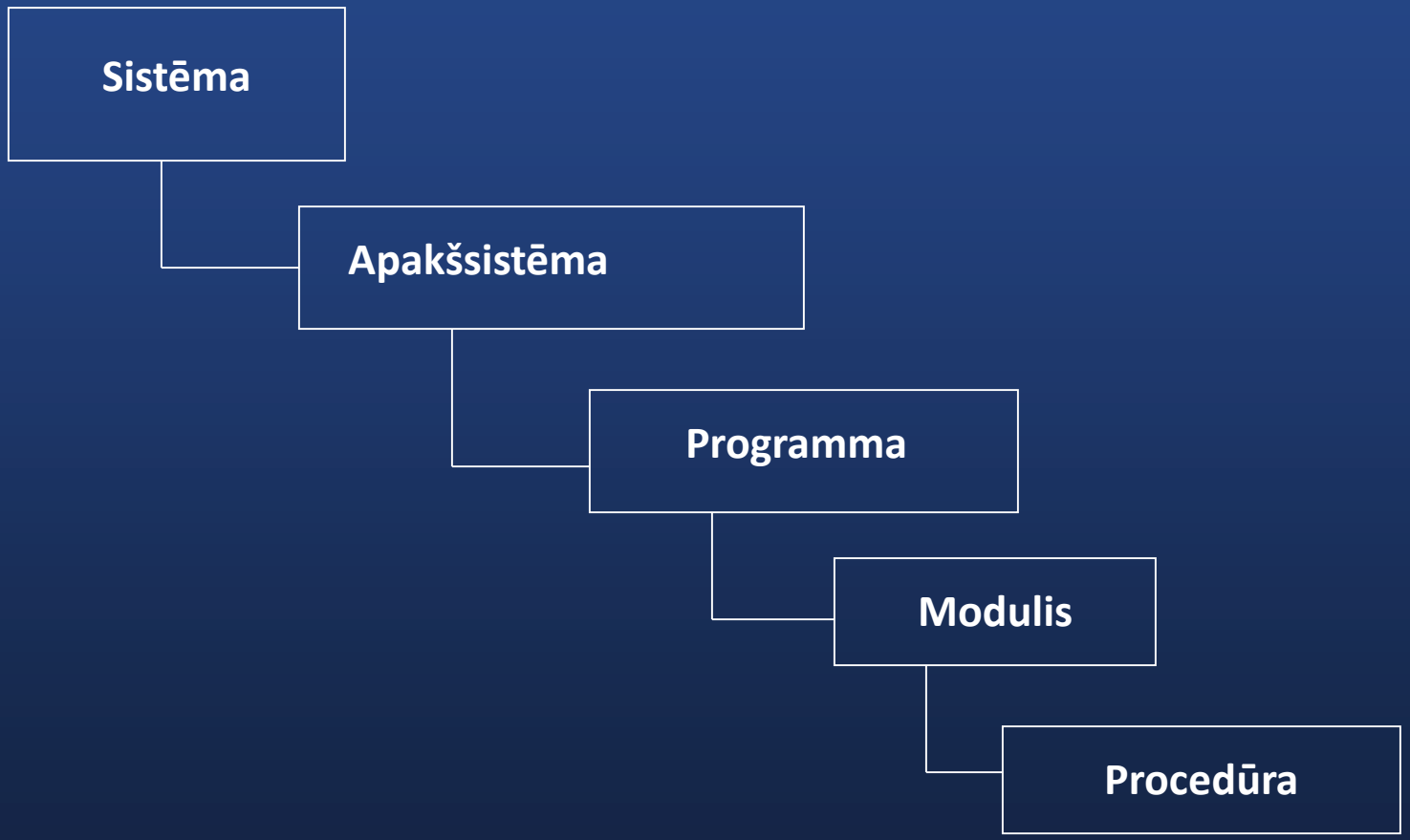
Programmatūras inženierija un datorzinātne sakarības



Programmatūras krīze

- Ātra aparatūras attīstība, programmatūra nespēja izmantot tās iespējas
- Jauno programmu izveides tempi neapmierināja pieprasījumus
- Programmatūras uzturēšanu apgrūtināja nepilnīgas konstrukcijas un līdzekļu trūkums

Programmatūras sastāvdaļas



Programmatūras sastāvdaļu definīcijas

Sistēma ir objektu, procedūru, vai paņēmienu kopums un to savstārpējas attiecības, kas funkcionāli veido vienoto veselumu.

Apakšsistēma ir sistēmas daļa, kas izpilda vienu vai vairākas tās pamat- vai palīgfunkcijas.

Programma ir instrukciju kopa, kas nosaka operāciju secību, ko izpilda dators datu apstrādes procesā.

Programmatūras sastāvdaļu definīcijas (II)

Modulis ir atsevišķa identificējamā programmas daļa, kuru var izveidot autonomi un izmantot, lai atvieglotu programmu sastādīšanu.

Procedūra ir darbību secība, kas jāveic, lai atrisinātu kādu uzdevumu. Šo terminu lieto, lai apzīmētu programmas daļu, kas veic kādu specifisku uzdevumu šīs programmas izpildes laikā.

Programmatūras klasifikācija

- Sistēmas programmatūra
- Reāllaika sistēmas
- Biznesa sistēmas
- Mākslīgā intelekta sistēmas
- Inženieru un zinātnes sistēmas
- Iegultās sistēmas
- Tīmekļa
- Personālo datoru programmatūra

Programmatūras klasifikācijas atšifrējums

Sistēmas programmatūra ir lietojumneatkarīga programmatūra, kas nodrošina lietojumprogrammu izpildi.

Reāllaika sistēmas veic datu apstrādi reālā laika; tām ir stingras laika prasības.

Biznesa sistēmas ir paredzētas biznesa informācijas apstrādei, kā arī palīdz pārvaldības lēmumu pieņemšanā.

Inženieru un zinātnes sistēmas zinātnisko pētījumu sistēmas.

Programmatūras klasifikācijas atšifrējums (II)

Mākslīgā intelekta sistēmas balstās uz zināšanām un ir paredzētas komplicētu problēmu risināšanai.

Iegultās sistēmas lieto produktu un sistēmu pārvaldībai.

Tīmekļa programmatūras piekļūšanai un datu vākšanai lieto pārlūkprogrammas.

Personālo datoru programmatūra paredzēta dažādu uzdevumu izpildei: tekstu apstrādei, grafikai un t.t.

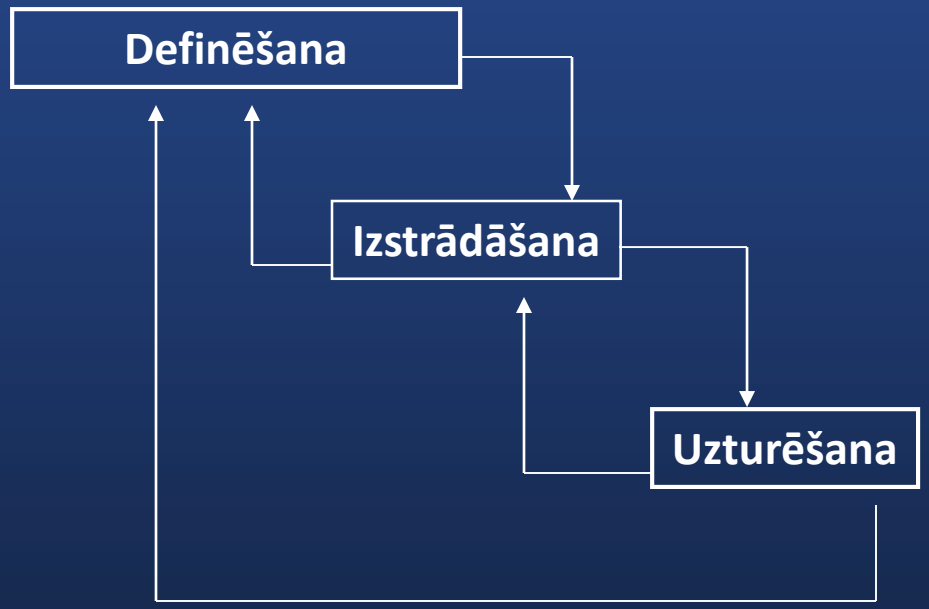
Pamatdefinīcijas

Programmatūras izstrāde ir process, kas sastāv no noteiktu aktivitāšu kopas, kuru izpilde noved pie programmatūras produkta radīšanas.

Programmatūras produkts ir lietotāja vajadzībām izstrādāta datorsistēma un ar to saistīta dokumentācija. Programmatūras izstrādes process balstās uz kādu no programmatūras dzīves cikla modeļiem.

Programmatūras dzīves cikls ir viss programmatūras produkta pastāvēšanas laiks no tā izstrādāšanas sākuma līdz brīdim, kad tas ir zaudējis savu praktisko vērtību.

Programmatūras dzīves cikla vienkāršota shēma



Kas jādara?

Sistēmas analīze
Plānošana, Prasību analīze

Kā izdarīt?

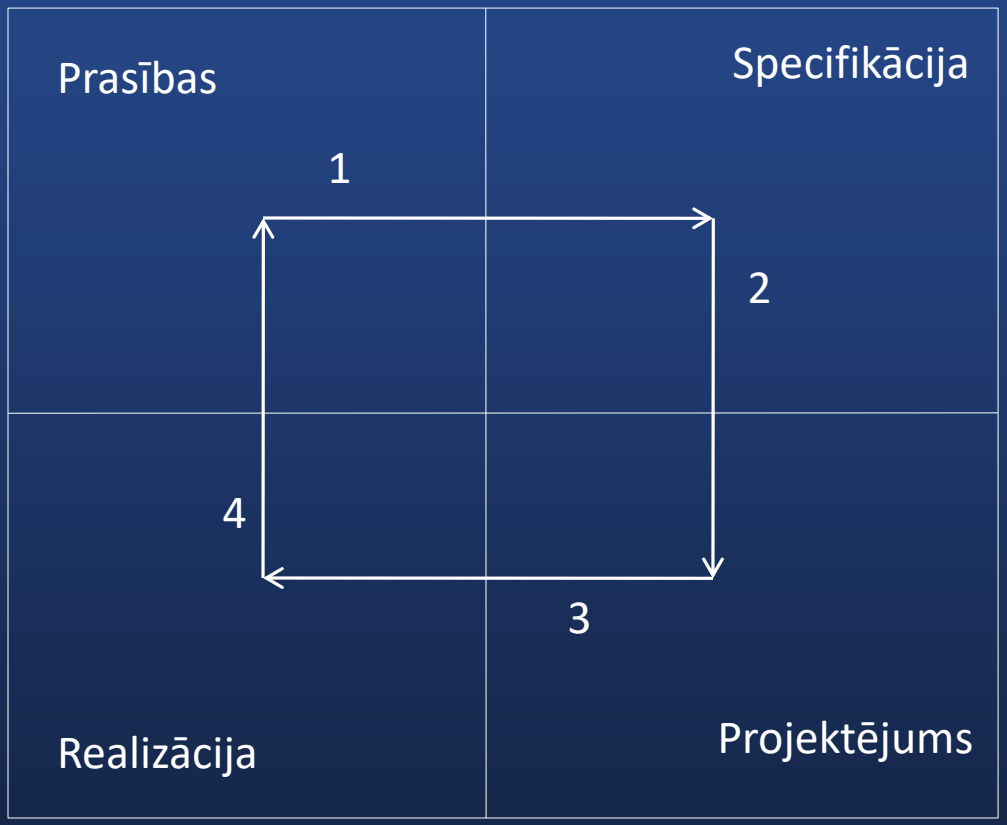
Projektēšana,
Implementēšana, Testēšana

Korekcija
Adaptācija
Uzlabošana

Programmatūras dzīves cikls diagrammas veidā

Reāla pasaule

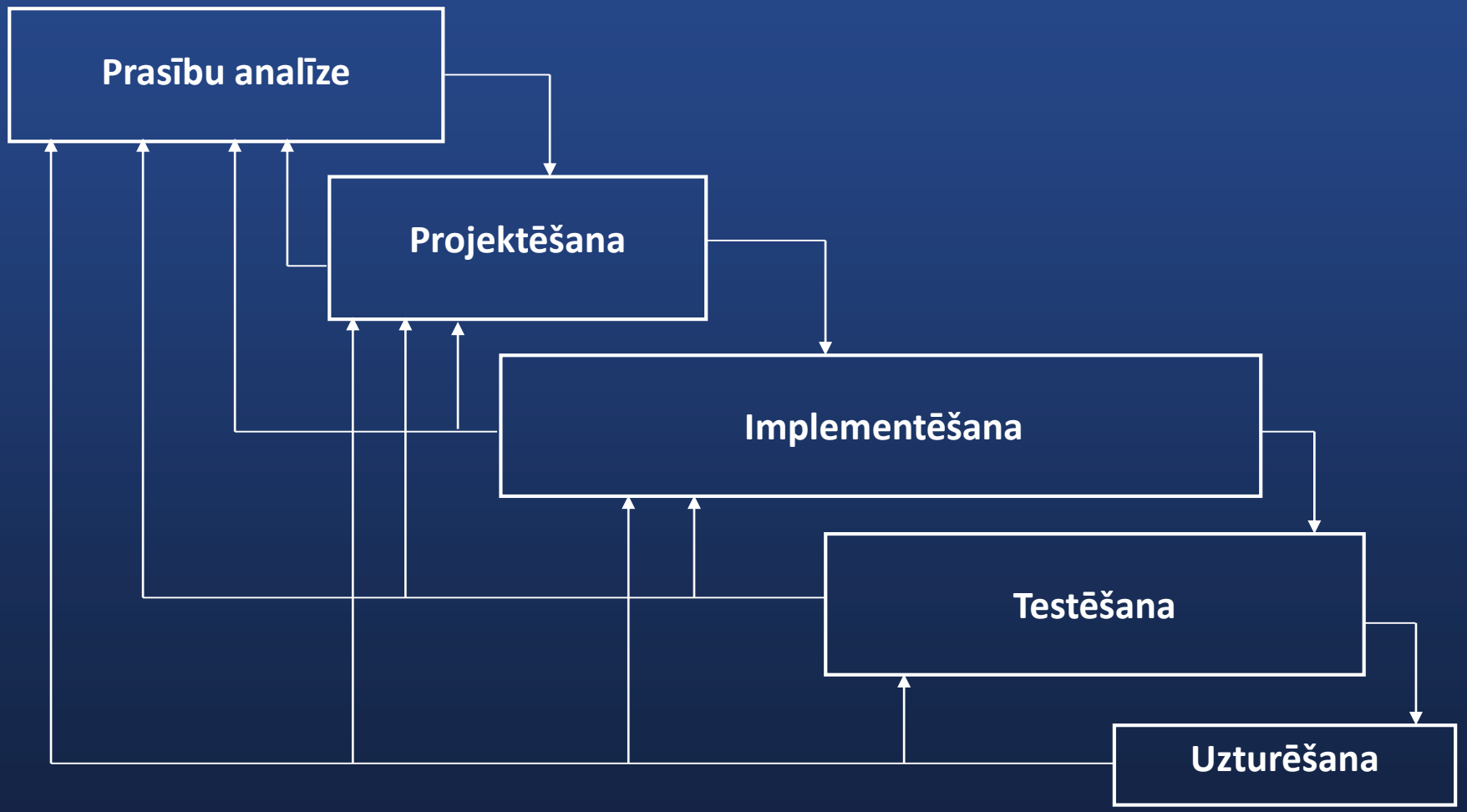
Abstrakcija



Programmatūras izstrādes modeļi

- Ūdenskrituma modelis
- Pētnieciskās programmēšanas modelis
- Prototipēšanas modelis
- Soļmodelis
- Formālā transformācija
- Sistēmas komplektēšana no atkārtotās lietošanas komponentiem

Ūdenskrituma modelis



Ūdenskrituma modeļa posmi

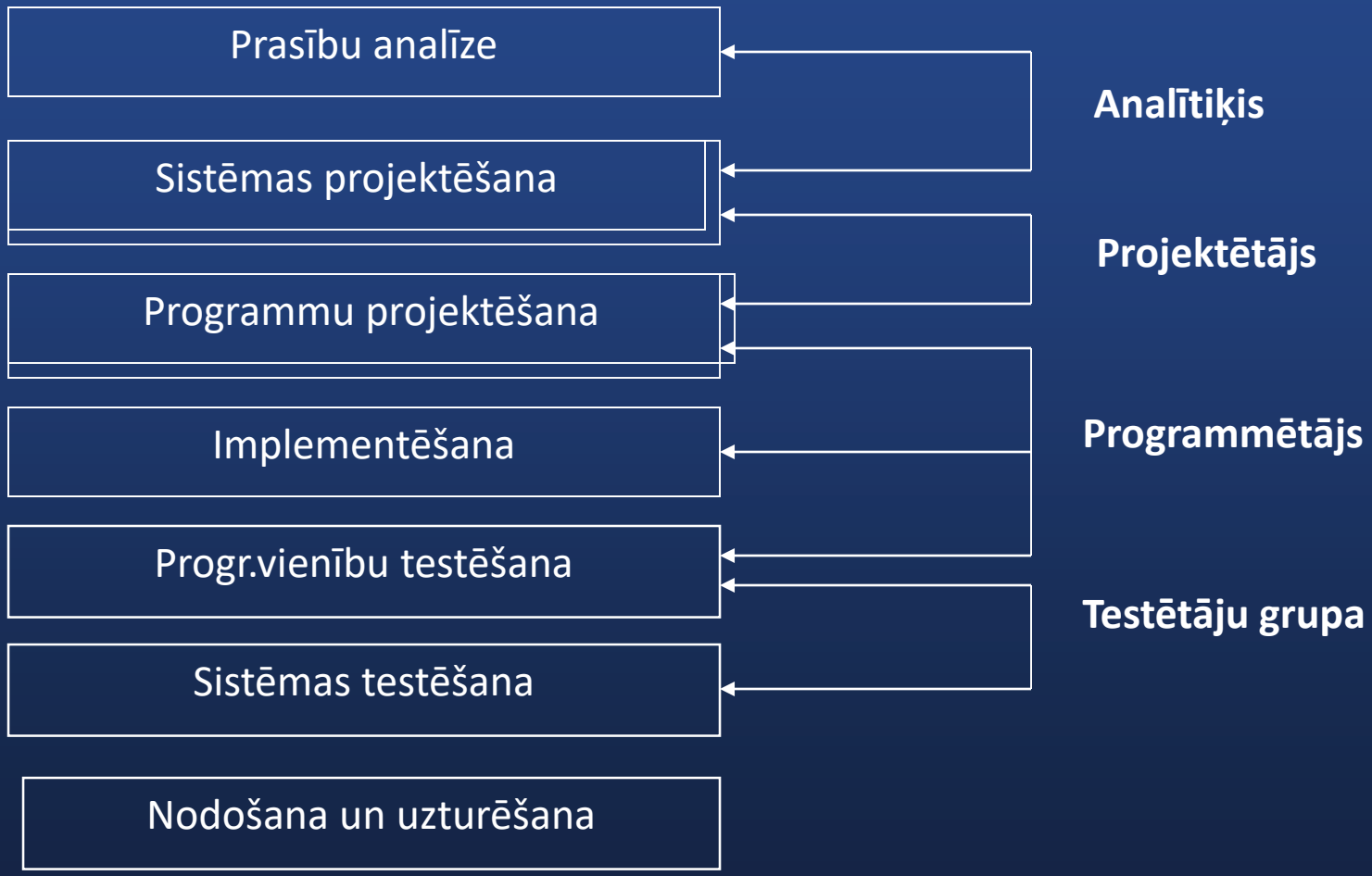
- **Prasību analīzes** posmā formulē programmatūras sistēmas izstrādes mērķi un nosaka prasības, kurām tai ir jāatbilst, t.i., izpildāmās funkcijas, darbības vidi, ātrdarbību, u.c. Iegūtās prasības apraksta formālā dokumentā, ko sauc par prasību specifikāciju. Speciālistu, kas darbojas programmatūras izstrādes posmā, sauc par sistēmanalītiķi.
- **Projektēšanas** posmā, pamatojoties uz iepriekš iegūtajām prasībām, definē sistēmas uzbūvi un datu glabāšanas struktūras, kā arī izstrādā sistēmas funkcionēšanas algoritmus. Visas nosauktās lietas apraksta dokumentā, ko sauc par projektējumu. Šajā programmatūras izstrādes posmā pamatā darbojas projektētājs.
- Pamatojoties uz izveidoto programmatūras sistēmas projektējumu, veic sistēmas realizāciju (kodēšanu) kādā programmēšanas valodā vai vidē. Šo darbu izpilda programmētājs. Tā rezultātā iegūst programmatūras sistēmas kodu.

Ūdenskrituma modeļa posmi (II)

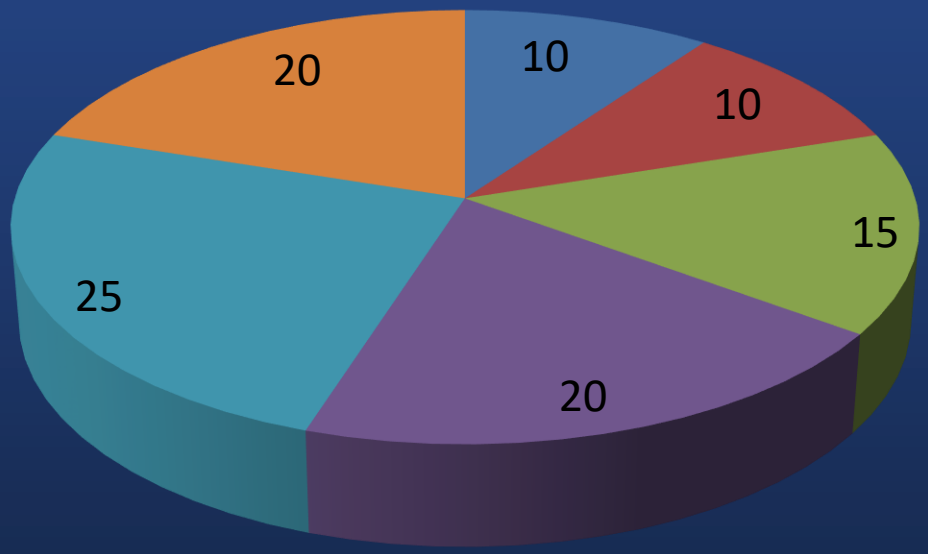
- Testēšanas posmā pārbauda izstrādāto sistēmu, lai noteiktu, vai tā atbilst sākotnēji izvirzītajām prasībām. Testēšanas rezultātus fiksē testēšanas dokumentācijā, kā arī sagatavo sistēmas lietotāja rokasgrāmatu, kas ir paredzēta sistēmas lietotājiem un viegli uztveramā valodā apraksta sistēmas izmantošanas iespējas. Speciālistu, kas darbojas šajā programmatūras izstrādes posmā, sauc par testētāju.

- Uzturēšanas posmā veic sistēmas uzstādīšanu un nodošanu ekspluatācijā pasūtītājam, kā arī izlabo jaunatklātas kļūdas, ievieš izmaiņas un adaptē sistēmu (ja tas ir nepieciešams) jauniem darba apstākļiem.

Posmu izpildītāji

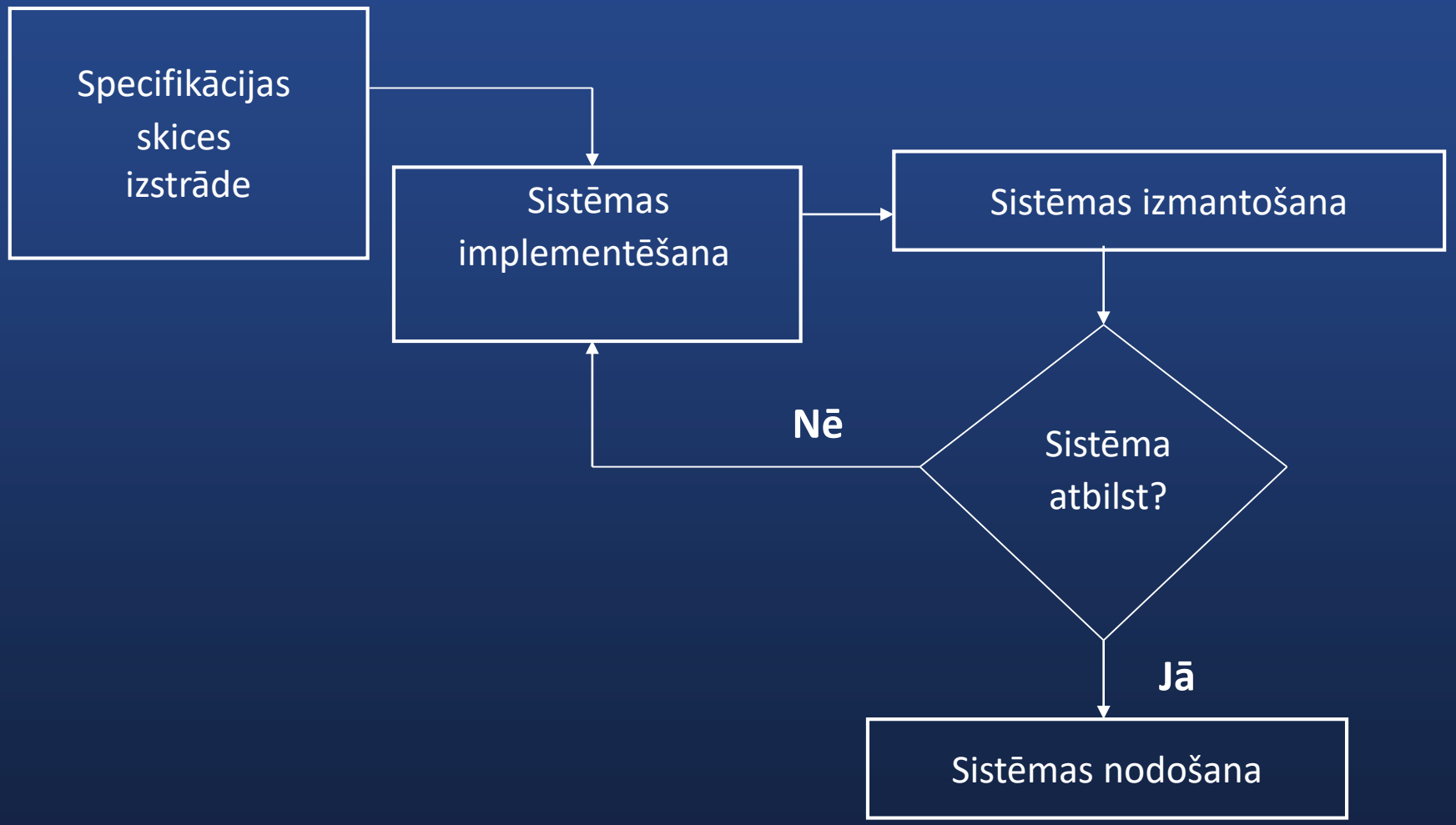


Laika patēriņa izstrādes procesā



- Prasību analīze
- Specifikāciju definēšana
- Projektēšana
- Kodēšana
- Programmvienību testēšana
- Sistēmas testēšana

Pētnieciskās programmēšanas modelis



Pētnieciskās programmēšanas pielietošana

Tā labi noder:

- Relatīvi mazu sistēmu izstrādei;
- Sistēmas izstrādei ar īso programmatūras dzīves ciklu;
- Apakšsistēmu izstrādei, kad nevar definēt sistēmas specifikācijas (piem., mākslīgā intelekta sistēmas).

Pētnieciskās programmēšanas ierobežojumi

- Programmatūras izstrādes process ir grūti kontrolējams
- Sistēmas parasti slikti strukturētas
- Speciālie kadri bieži ir vajadzīgi

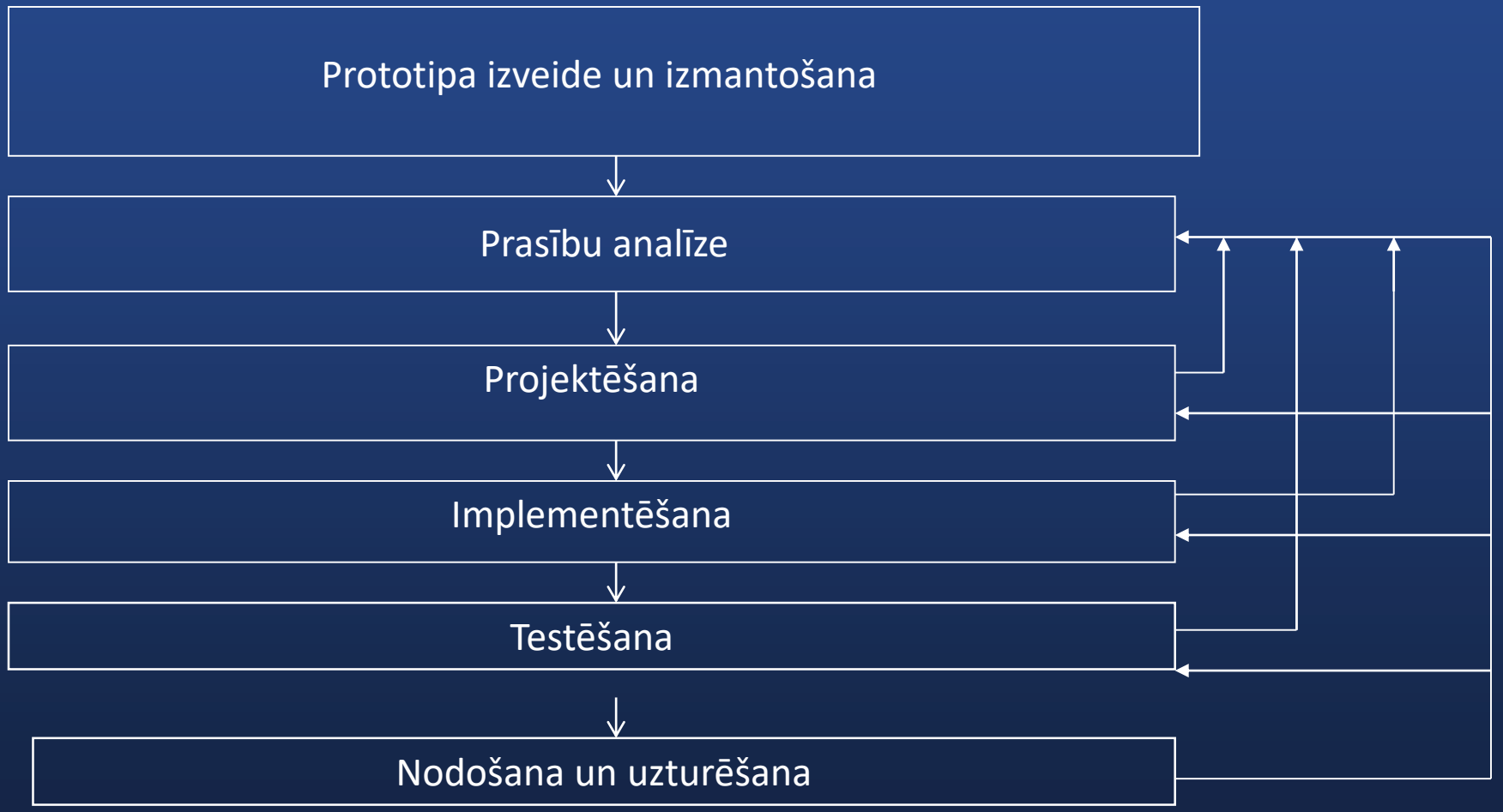
Prototipēšanas modelis

Lieto, kad lietotāji *nespēj* formulēt savas prasības un definēt kā sistēmai ir jāfunkcionē.

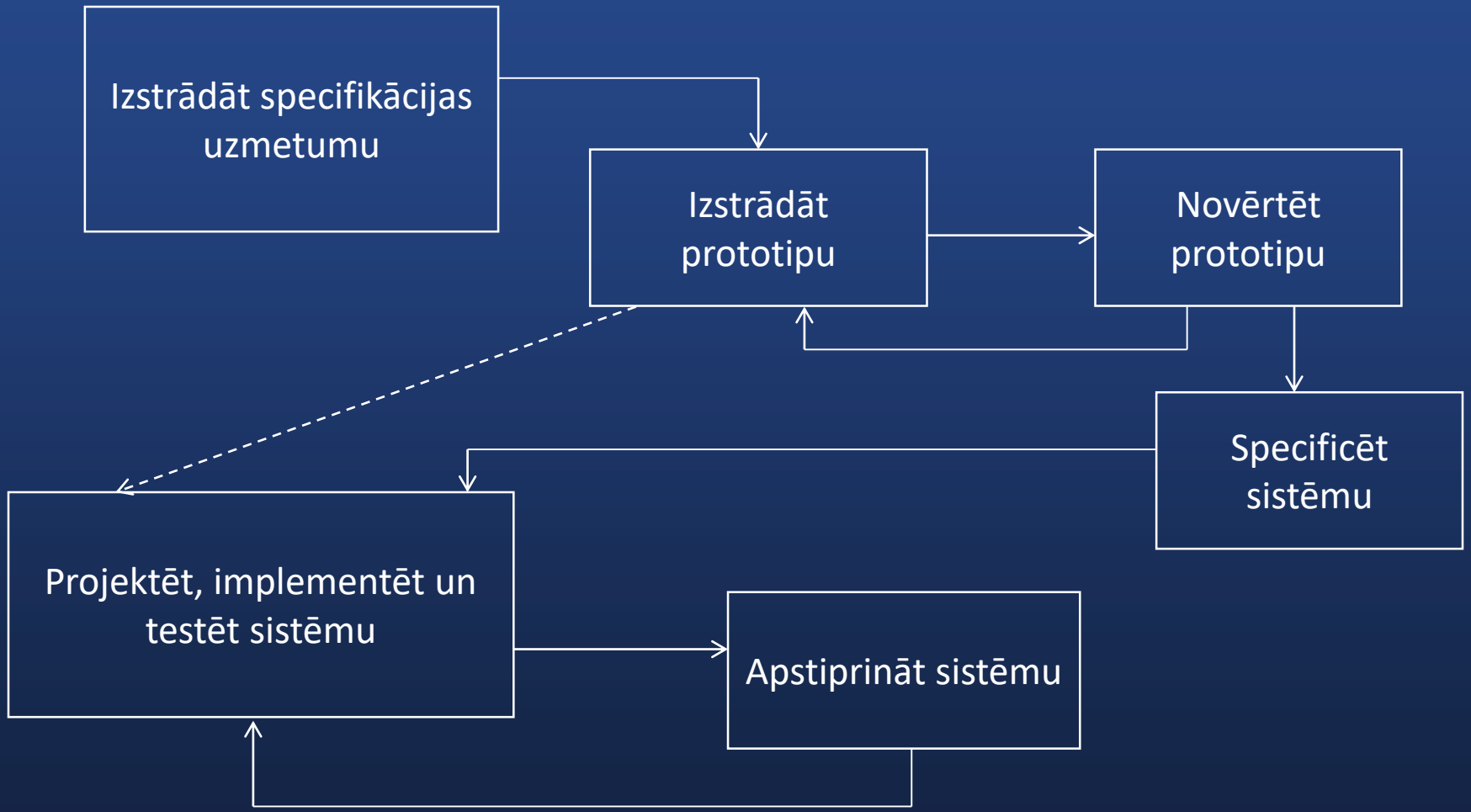
Prototipa izstrāde pamatojas uz sistēmas specifikācijas skici.

Mērķis: atklāt sistēmas prasības un specifikācijas.

Prototipēšanas modeļa lietošanas shēma



Prototipēšanas modeļa shēma



Soļmodelis

Paredz pakāpenisku soļsecīgu programmatūras izstrādi.

Nodrošina iespēju lietot, pārbaudīt un novērtēt katru sastāvdaļu izveides laikā.

Lietderīgi izmantot, kad:

- Nav pietiekošu darbinieku sistēmas izstrādei;
- Ir plānots novērtēt programmatūras tehniskus riskus.

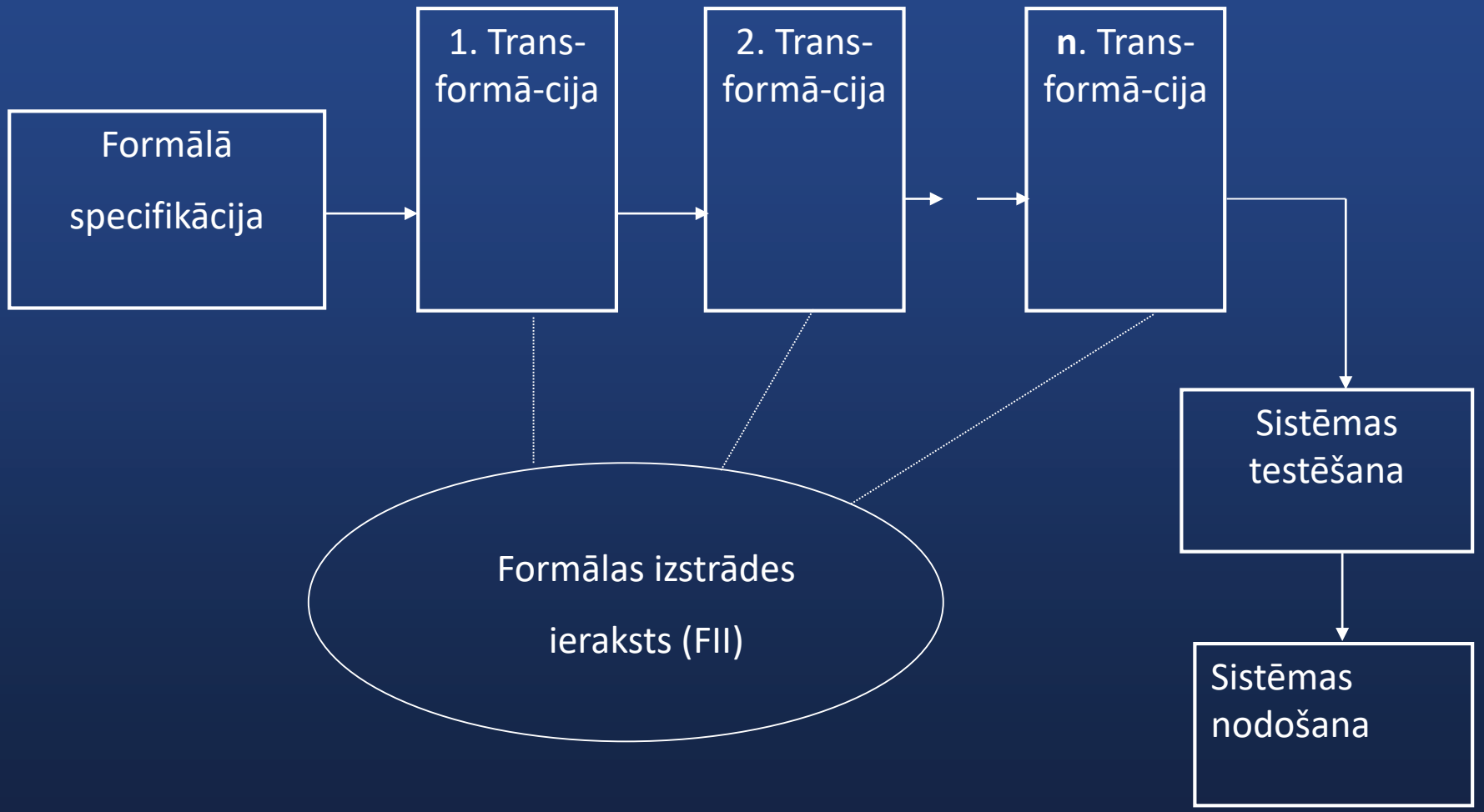
Soļmodeļa shēma

Formālās transformācijas modelis

Paredz sistēmas formālās specifikācijas izstrādi, kura pēc tam tiks transformēta programmā ar speciālo rīku palīdzību.

Formālā specifikācija ir specifikācija kura uzraktīta valodā ar formāli definētu vārdnīcu, sintaksi un semantiku.

Formālās transformācijas modeļā shēma



Formālās transformācijas modeļa trūkumi

1. Prasa vairāk laika;
2. Saskarne ar klientiem ir apgrūtināta, jo tie nepārzina formālās specifikācijas tehniku;
3. Programmatūras dažas klases grūti specificējamas;
4. Trūkst metožu un rīku šīs tehnikas atbalstam;

. . .

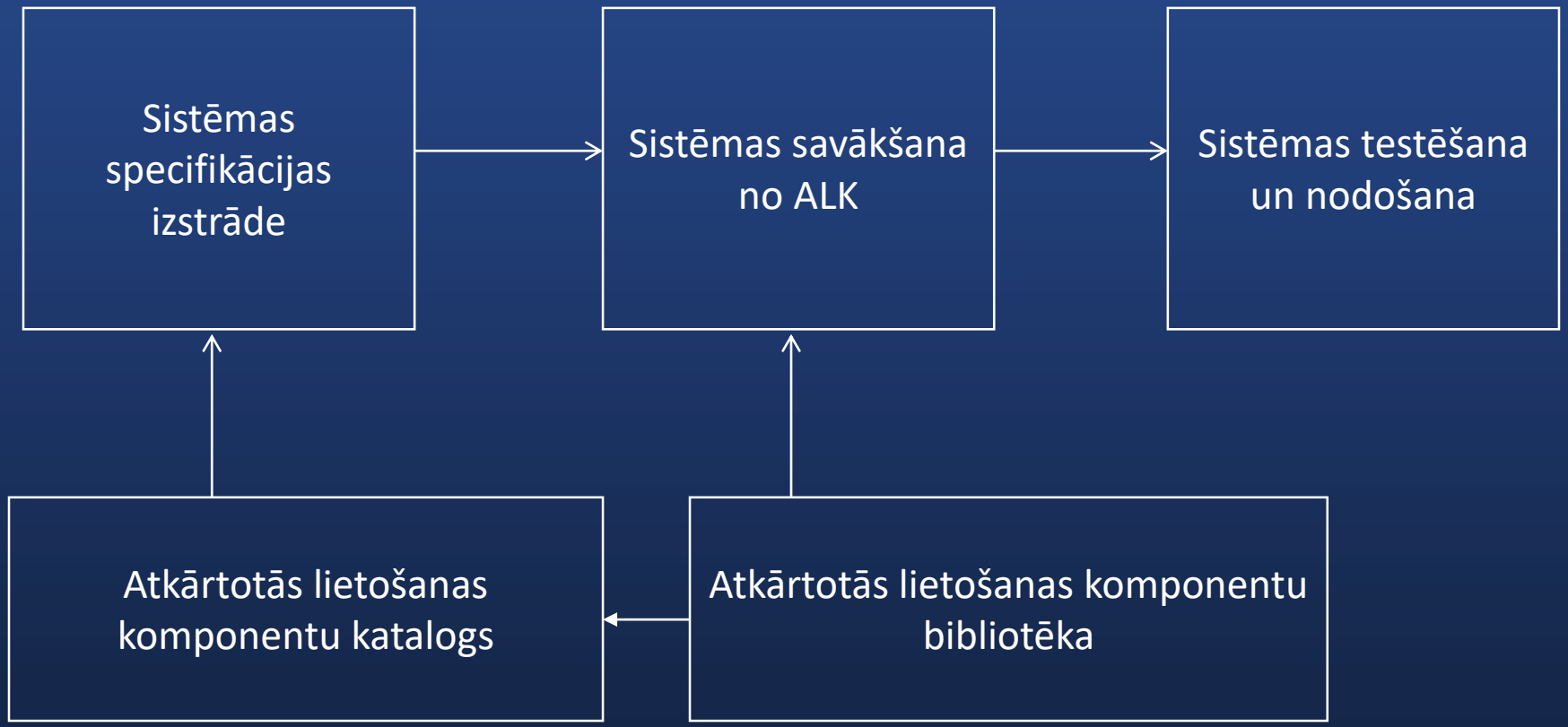
Sistēmas komplektēšanas no komponentiem modelis

Paredz izmantot atkārtotās lietošanas komponentu (ALK) bibliotēku.

Ļauj samazināt programmatūras izstrādes izmaksas.

Ir nepieciešama ALK bibliotēka un katra komponenta detalizēta dokumentācija.

Sistēmas komplektēšana no komponentiem



Līguma izstrāde

Līguma mērķi

1. Skaidri definēt pasūtītāja un izstrādātāja pienākumus, lai abas iesaistītās puses zinātu, kas ir jādara un ko viņi par to saņems.
2. Noteikt kalendāro grafiku, saskaņā ar kuru tiks nodoti un izstrādāti atsevišķi līgumā noteiktie produkti un pakalpojumi.
3. Skaidri definēt procedūras, saskaņā ar kurām katra puse var ieviest izmaiņas un noteikt, cik lielā mērā izmaiņas var ietekmēt plānu un izmaksas.
4. Definēt terminus uz kuru pamata pasūtītājs pieņems produktu pēc tā izstrādes pabeigšanas un apmaksu.
5. Definēt, kādā veidā tiks izskatīti strīdīgi jautājumi.

Līguma priekšmets

Līgumā nepieciešams noteikt:

- programmatūras funkcionalitāti;
- standartus, kuriem atbilst programmatūra;
- vidi, kurā darbosies programmatūra;
- programmatūras veiktspēju šajā vidē.

Izstrādes iespējas

Pasūtot programmatūras izstrādi, pastāv divas iespējas:

- pasūtītājs definē vispārīgas prasības un slēdz līgumu prasību specifikācijas, projektējuma un programmatūras izstrādei;
- pasūtītājs definē detalizētas prasības un slēdz līgumu projektējuma un programmatūras izstrādei.

Programmatūras izstrādes veidi

Divi programmatūras veidi:

- tāda, kas veidota atbilstoši pasūtītāja prasībām;
- standartprogrammatūra, kas ir pielāgota pasūtītāja prasībām.

Kam piederēs programmatūra?

- veidojot programmatūru īpašām pasūtītāja prasībām, programmatūra kļūst par pasūtītāja īpašumu un tiek lietota tikai šajā uzņēmumā;
- programmatūra pieder izstrādātājam, pasūtītājam tiek izsniegta licence darbam ar programmatūru, un izstrādātājam ir tiesības izsniegt licences arī citiem uzņēmumiem;
- programmatūra pieder pasūtītājam, bet izstrādātājam tiek dota licence programmatūras izmantošanai mārketinga vajadzībām.

Kā tiek veikta apmaksa?

Programmatūras izstrāde var būt veikta par :

- fiksētu cenu

  izstrādātājs riskē vairāk, bet centīsies lielākā mērā kontrolēt izstrādes procesu;

- vai balsties uz laiku un izmaksām

  elastīgāka, projekts tiek vājāk kontrolēts; nepieciešams vienoties par resursu izmaksu likmēm un resursu uzskaites procedūrām.

Citi jautājumi

Līgumā var būt apskatīti vēl šādi jautājumi:

- kādus dokumentus ir nepieciešams izveidot, atbilstoši kādiem standartiem un kas nodrošinās šos standartus;
- kas un par kādu maksu uzstādīs programmatūru;
- kādas darbības ir nepieciešams izpildīt, lai programmatūra tiktu pieņemta (t.i., kādu testus ir nepieciešams izpildīt), un kurš to darīs;
- kādus datus būs nepieciešams pārveidot (no vecās sistēmas uz jauno) un kas to darīs;
- cik lielā mērā un kādā laika posmā tiks nodrošināta kļūdu labošana;
- kādi pasākumi ir nepieciešami kvalitātes nodrošināšanai un kas tos veiks.

Projekta vadītāja loma

Projekta vadītāja pienākumi

- veikt projekta iegūšanas aktivitātes, kas ir saistītas ar potenciālā pasūtījuma specifikas izpēti, iespējamo tehnisko risinājumu izvērtēšanu, darbietilpības prognozēšanu, izmaksu analīzi un piedāvājuma dokumentu sagatavošanu;
- plānot projekta resursus (laiku, budžetu, personālu, u.c.) un izvērtēt iespējamus riskus;
- nodrošināt projekta kvalitātes pārvaldību;
- vadīt projekta personālu, sadalot individuālus darba uzdevumus, organizējot personāla apmācību un vērtējot darbinieku prasmes un spējas;
- kontrolēt projekta izpildi, veicot nodevumu apskates, pārskatot plāna izpildi, vērtējot pasūtītāja apmierinātību un kontrolējot gala produkta atbilstību pasūtītāja prasībām;
- atskaitīties par projekta gaitu uzņēmuma vadībai un pasūtītājam.

Projekta vadītāja nepieciešamās zināšanas un prasmes

Projekta vadītājam jāprot:

- organizēt, plānot un noturēt sapulces,
- prezentēt informāciju dažādām auditorijām saprotamā valodā,
- pamatot savu viedokli un pārliecināt citus,
- komunicēt vismaz angļu valodā pārrunām ar pasūtītāju un tehniskās literatūras lasīšanai.

Projekta vadītāja nepieciešamās zināšanas un prasmes (II)

Projekta vadītājam jāspēj:

- sastādīt projekta kalendāro plānu;
- vērtēt riskus;
- plānot budžetu;
- novērtēt projekta darbietilpību;
- strādāt ar IT nozares standartiem un speciālo tehnisko literatūru;
- veidot un vadīt projekta komandu, motivēt darbiniekus, deleģēt pienākumus, pieņemt lēmumus un risināt konfliktus.

PALDIES PAR UZMANĪBU!