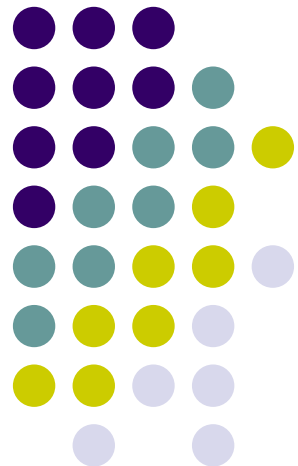
Rational Unified Process

PAT



Vladimirs ŅIKUĻŠINS

3.kursa doktorants

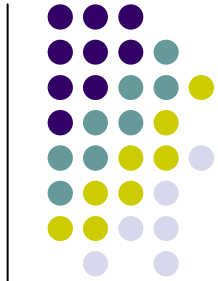
RTU DITF LDI

Lietišķo datorzinātņu katedra

Lekcijas plāns

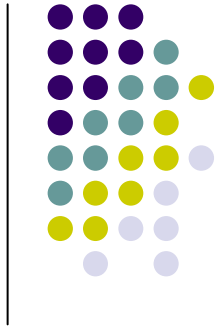
- Kas ir RUP
- RUP struktūra
- RUP darbplūsmas
  - Sākuma fāze
  - Izvēršanas fāze
  - Konstruēšanas fāze
  - Pārejas fāze
- RUP disciplīnas
    - Biznesa modelēšana
    - Prasību definēšana
    - Analīze un projektēšana
    - Realizācija
    - Testēšana
    - Izvietošana
    - Konfigurāciju un izmaiņu vadība
    - Projekta vadība
    - Vide

Kas ir RUP?



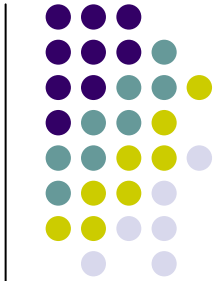
- Iteratīva, uz arhitektūru un lietošanas gadījumu virzīta programmatūras izstrādes pieeja
- Labi strukturēts programmatūras inženierijas process, kurā ir skaidri noteikts, kurš par ko ir atbildīgs, kad un kādas darbības ir jāveic
- Procesa produkts, kurš nodrošina programmatūras izstrādātāju ar viegli mērogojamo procesa karkasu

RUP vēsture



- Programmatūras izstrādes metodoloģija, kura attīstās kopš 1991. gada
- Pirmo reizi tika piedāvāta 1994. gadā kā kompānijas Microsoft labāko prakšu kopa
- 2003. gadā tika izlaista versija 3.0
- Pašlaik aktuālākā versija ir 4.0, kura tika izlaista 2005. gada novembrī
- MSF 4.0 pilnīgi integrēta ar programmatūras izstrādes rīku Microsoft Visual Studio 2005 Team System

RUP labas prakses



- Izstrādāt programmatūru iteratīvi
- Pārvaldīt prasības
- Izmantot komponenšu bāzētas arhitektūras
- Vizuāli modelēt programmatūru
- Pārbaudīt programmatūras kvalitāti
- Kontrolēt programmatūras izmaiņas

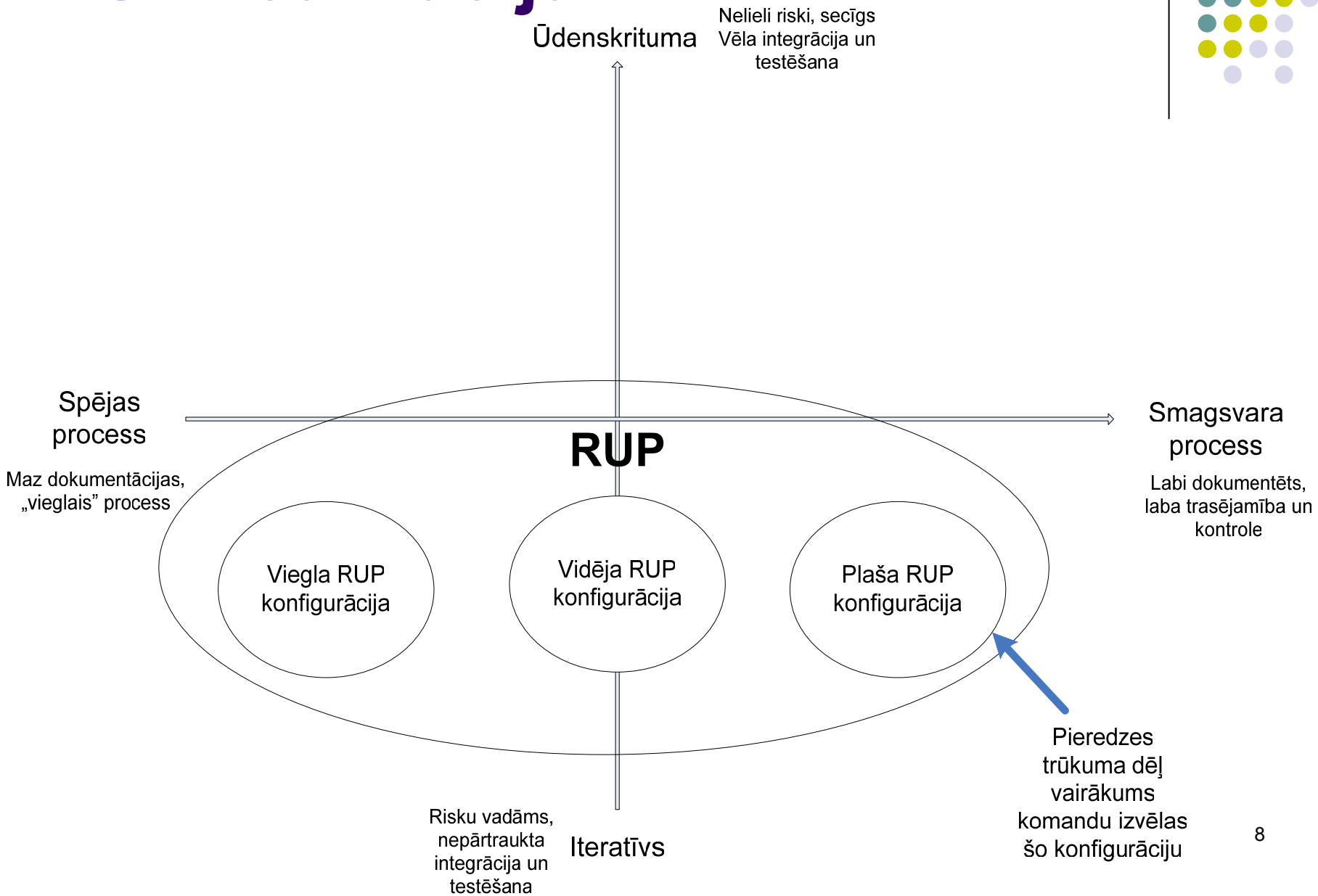
Komandas organizācija

- Analītiķu lomu kopa - prasību noskaidrošana un pētīšana
- Izstrādātāju lomu kopa - programmatūras projektēšana un implementēšana
- Testētāju lomu kopa - testēšana
- Vadītāju lomu kopa - programmatūras inženierijas procesa vadība un konfigurācija
- Papildu lomu kopa - izpilda dažādas atbalsta darbības

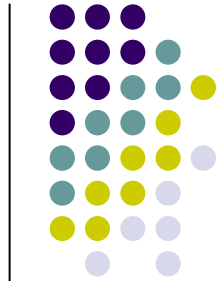
RUP pielietošanas stratēģijas

- RUP procesa konfigurācija
  - Adoptēt procesa produktu
- RUP procesa realizācija
  - Izmainīt organizācijas biznesa procesus

RUP realizācija



Realizācijas efekti



- Procesa izmainīšana ir sarežģītāka par programmatūras nomainīšanu
- Ietekmes objekti:
  - Cilvēki un viņu kompetence, zināšanas, motivācija un attieksme
  - Rīki
  - Programmatūras izstrādāšanas process
  - Programmatūras izstrādāšanas procesa izskaidrojums

Risku identificēšana (1)

- Pirmā iterācija ir pārāk ambicioza un cilvēki nekoncentrējas uz pašu svarīgāko
- Katrā iterācijā ir nepareizi realizēti vai notestēti artefakti. Lai samazinātu riskus ir jāveic testēšana katrā iterācijā
- Dažas ieinteresētas puses nesaprot iteratīvu pieeju
- Plānošana notiek soli pa solim, augsta līmeņa plānošanas vietā
- Daudz artefaktu, kas ir jāpārstrādā pēc iepriekšējas iterācijas izpildīšanas

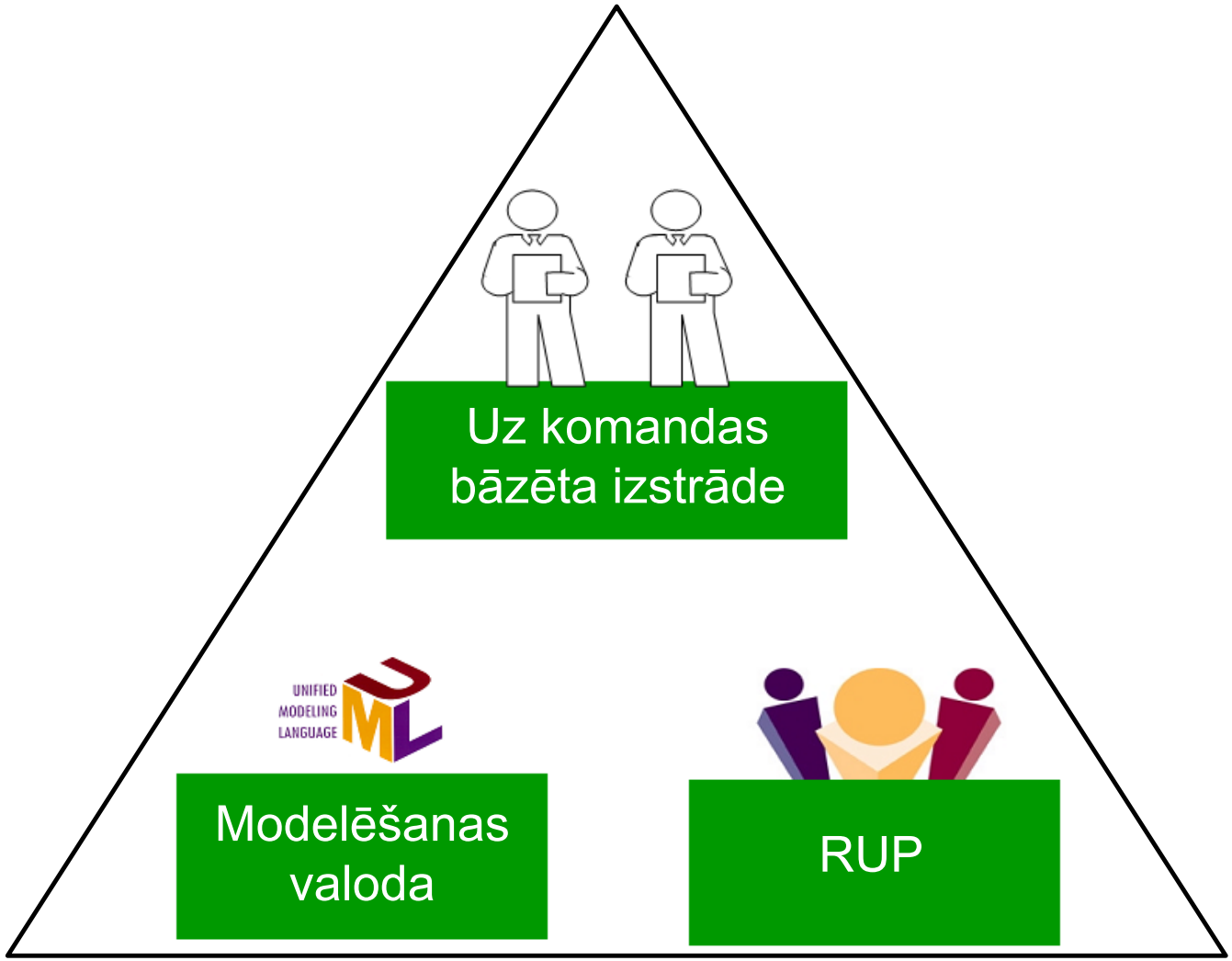
Risku identificēšana (2)

- Izmaiņas netiek kontrolētas ar domu, ka jebko var izmanīt nākošās iterācijas
- Cilvēku apmācīšana tiek izdarīta pārāk ātri un pirms cilvēki sāk strādāt viņi jau aizmirst ko viņi ir iemācījušies
- Tiek piesaistīti pārāk daudz cilvēku, cilvēki nesaprot savus pienākumus, vēlāk tie var būt noslogoti nelielā apjomā
- Cilvēki nevar izmanot rīkus (rīki nav piemēroti vai cilvēkiem pietrūkst zināšanas)
- Ir kļūda novērtēšana un tiek veidoti nevajadzīgi produkti

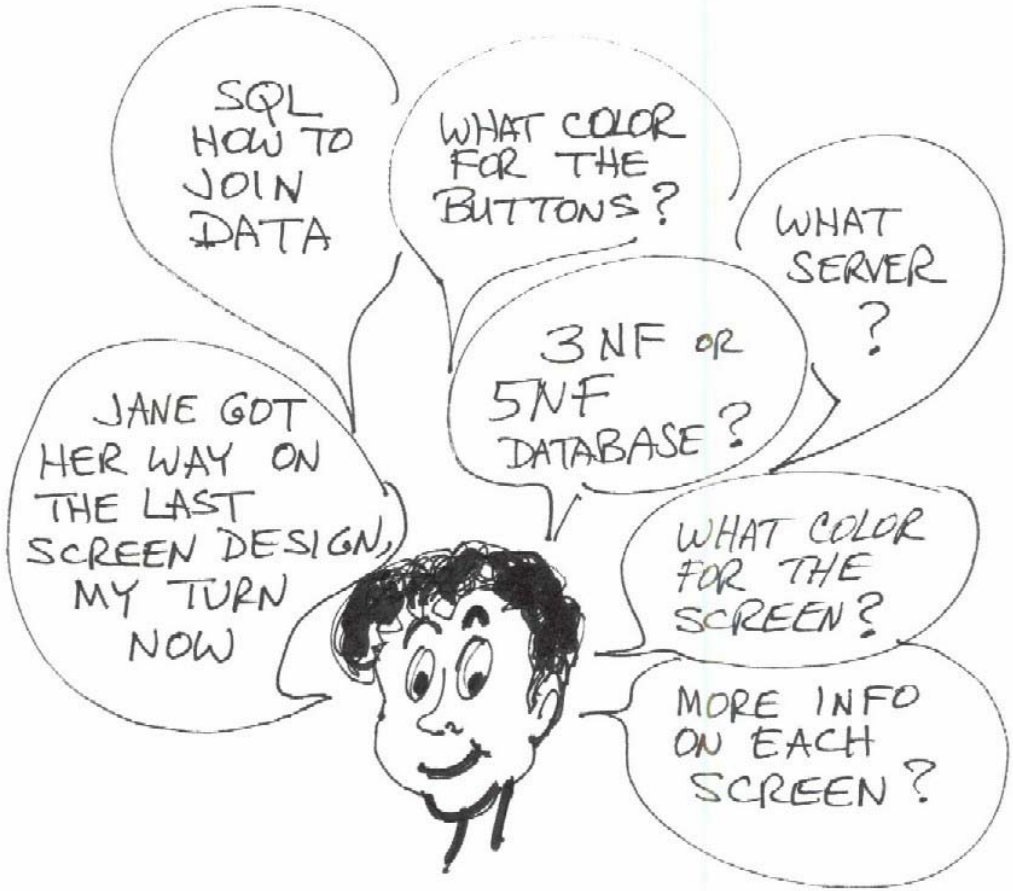
Vizuālā modelēšana

- Vienota modelēšanas valoda (Unified Modelling Language, UML)
- UML diagrammas:
  - Klašu diagramma
  - Lietošanas gadījumu diagramma
  - Aktivitāšu diagramma
  - Stāvokļu diagramma
  - Secību diagramma
  - Sadarbības diagramma
  - Komponenšu diagramma
  - Izvērsuma diagramma

Vizuālā modelēšana un RUP



Kāpēc jālieto lietošanas gadījumi?

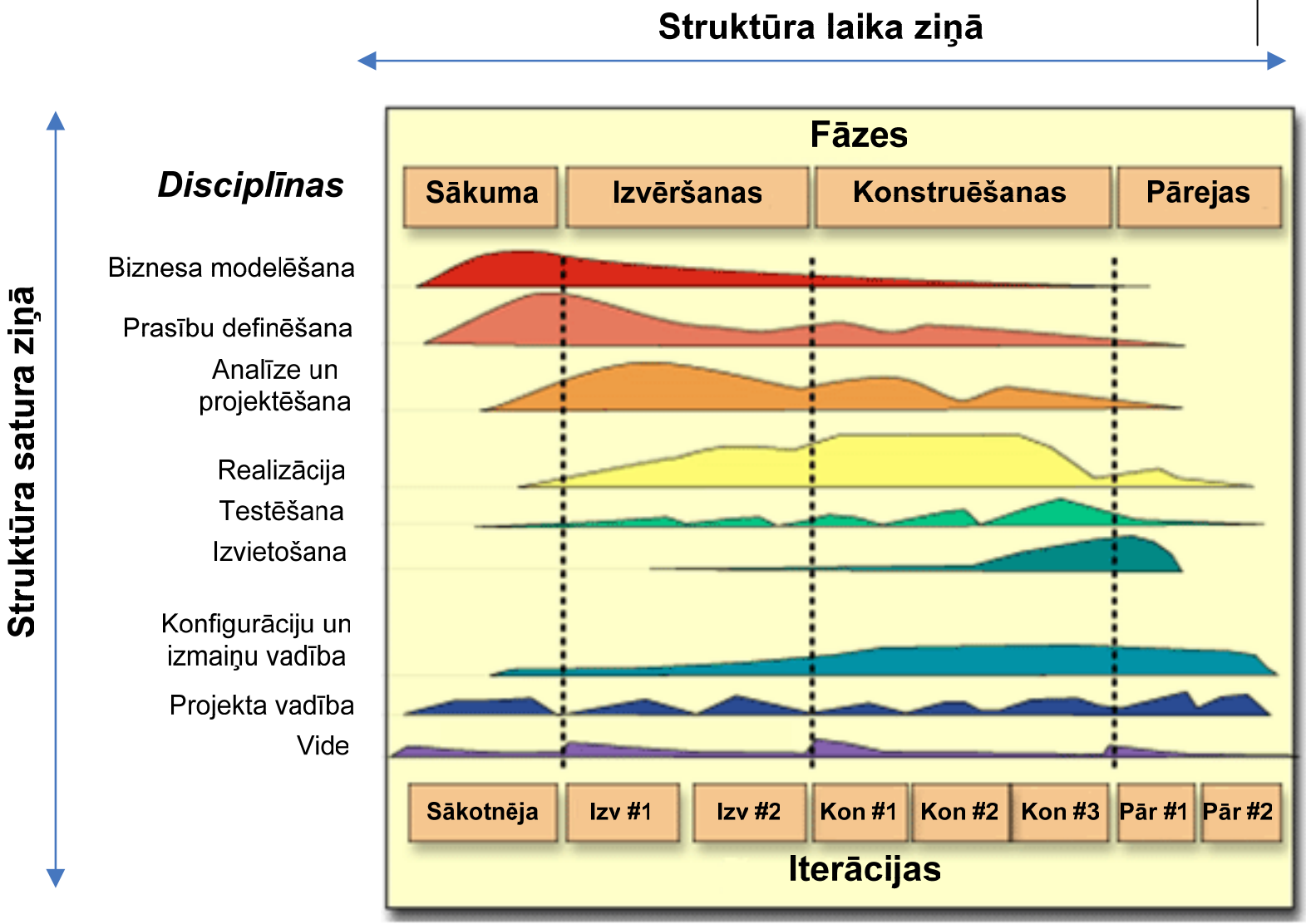


Nefunkcionālas prasības < - > Kādiem mērķiem lietot lietojumprogrammu darbā?

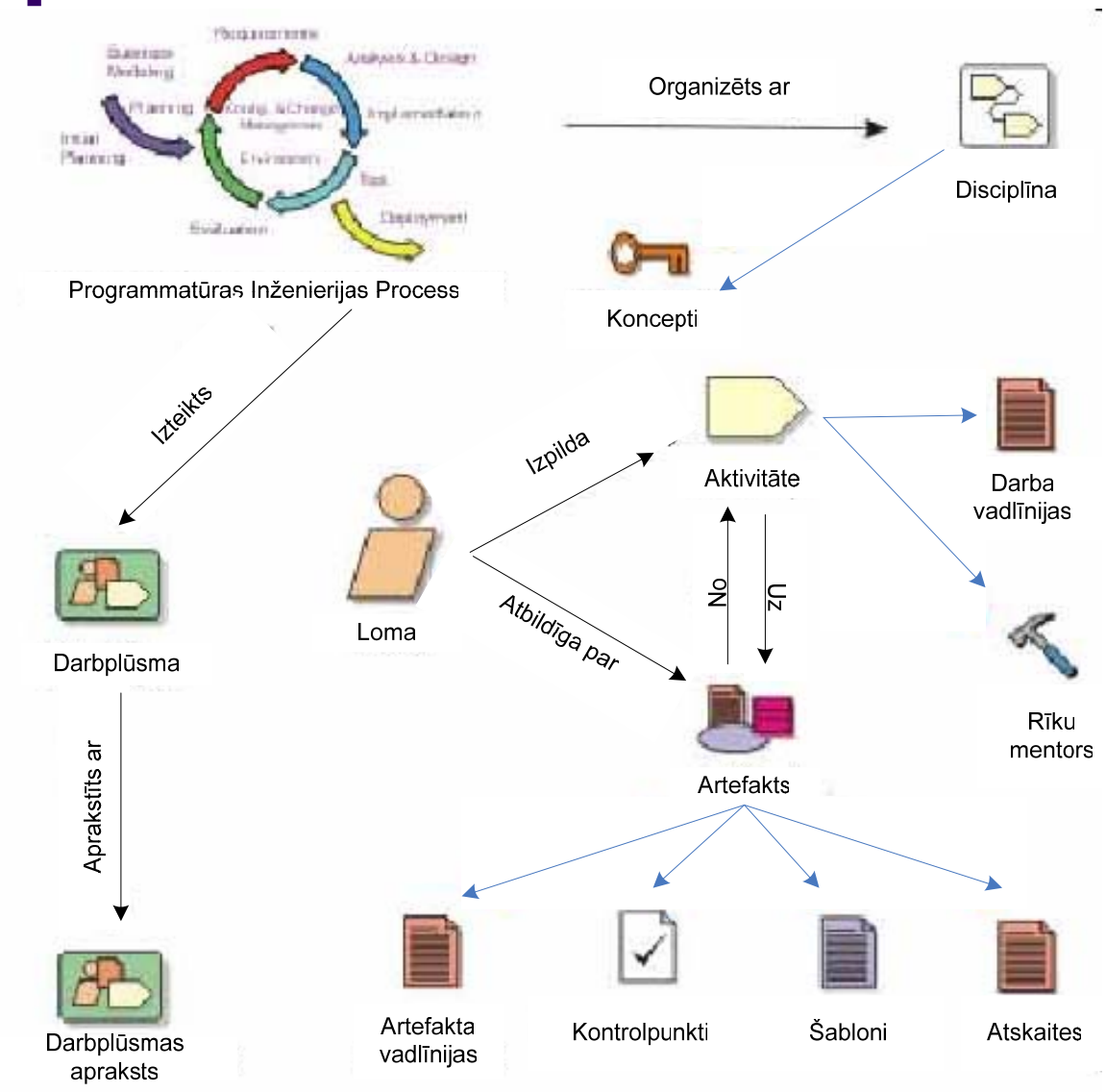
RUP dimensijas

- *Rational Unified Process* sastāv no divām struktūrām jeb dimensijām
  - Dinamiskā struktūra
    - Laiks (fāzes un iterācijas)
      - cikli, fāzes, iterācijas, pagrieziena punkti (*milestones*)
  - Statiskā struktūra
    - Saturs (disciplīnas)
      - procesa komponentes, aktivitātes, darbplūsmas, artefakti, darbinieki

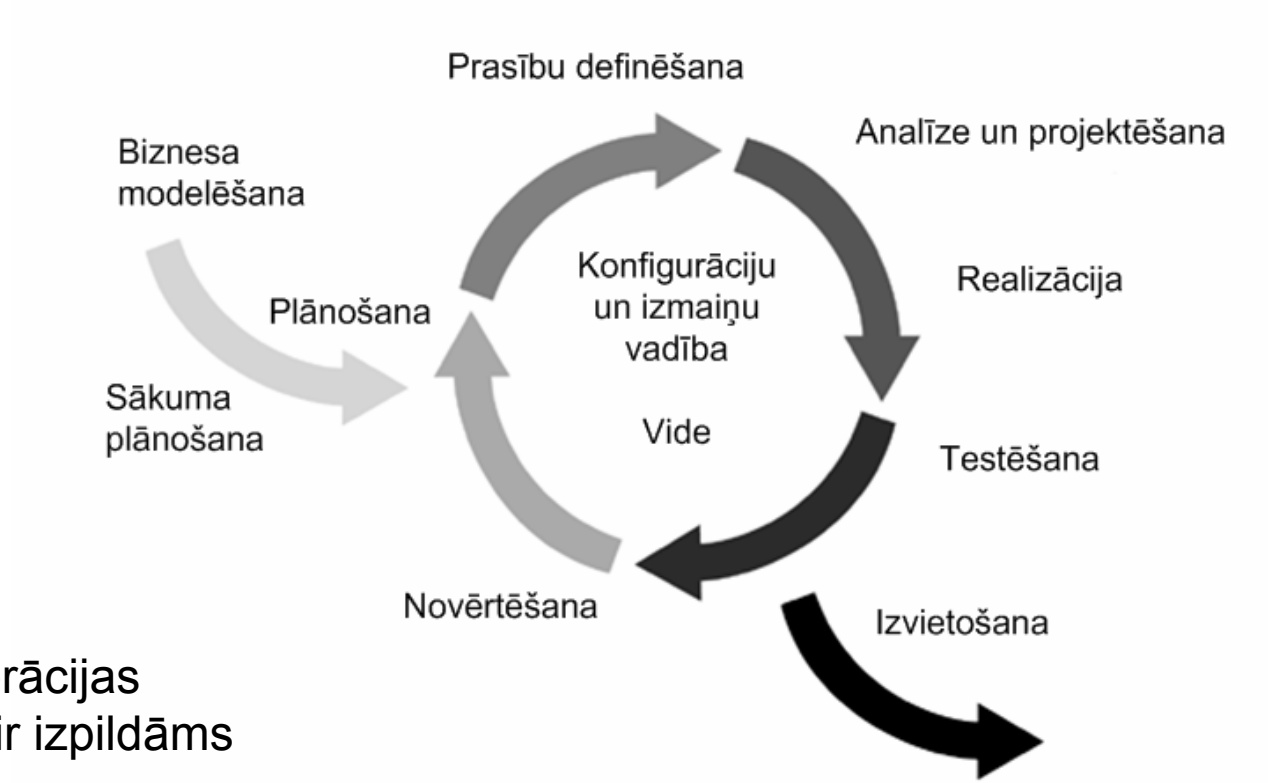
RUP procesa kopēja uzbūve



RUP pamatelementi



Iteratīvs procesa modelis (1)

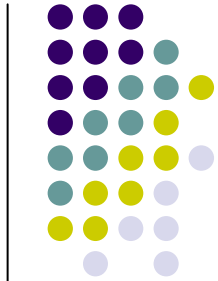


Katras iterācijas rezultāts ir izpildāms laidiens

Iteratīvs procesa modelis (2)

- Ļauj ievērot prasību izmaiņas un operatīvi reaģēt uz tām
- Visi elementi tiek pakāpeniski integrēti sistēmā vairāku iterāciju laikā
- Riski tiek identificēti un samazināti daudz agrāk
- Atvieglo komponenšu daudzkārtēju izmantošanu
- Nodrošina stabilāku arhitektūru
- Ļauj racionālāk izmantot personālu
- Process var būt uzlabots un pilnveidots dzīves cikla laikā

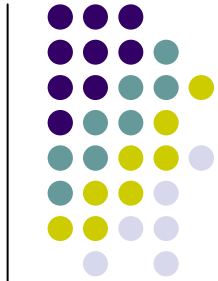
Statiskā struktūra (1)



Pamatelementi:

- Loma – indivīda vai indivīdu grupas uzvedības un atbildību definējums
- Aktivitāte – darba veids ar noteiktu mērķi
- Artefakts – informācijas vienība, kuru izveido, maina vai izmanto process
- Darbplūsma – aktivitāšu sekvence, kura dod pamanāmu vērtību

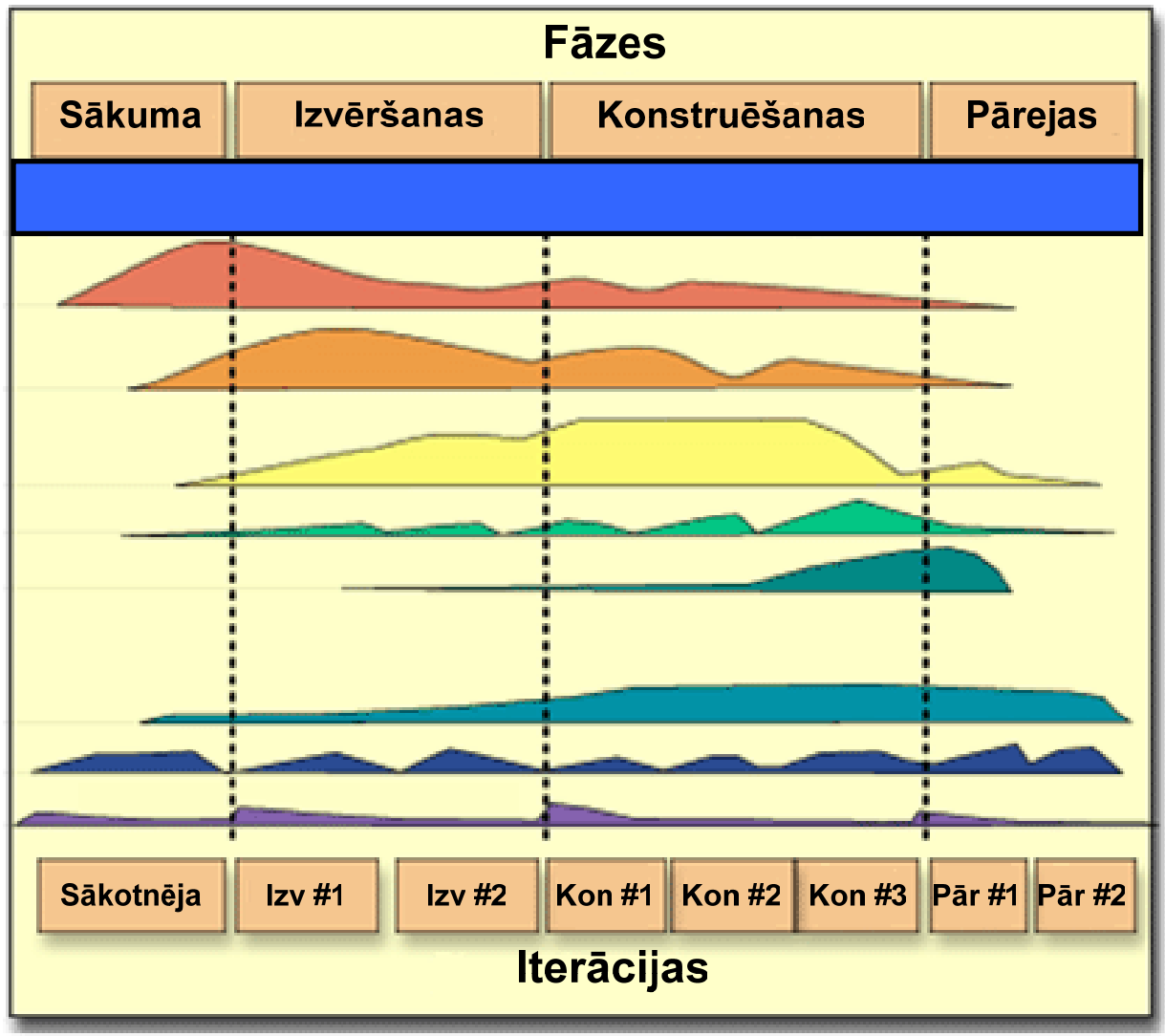
Statiskā struktūra (2)



Disciplīnas:

- Biznesa modelēšana
- Prasību definēšana
- Analīze un projektēšana
- Realizācija
- Testēšana
- Izvietošana
- Konfigurāciju un izmaiņu vadība
- Projekta vadība
- Vide

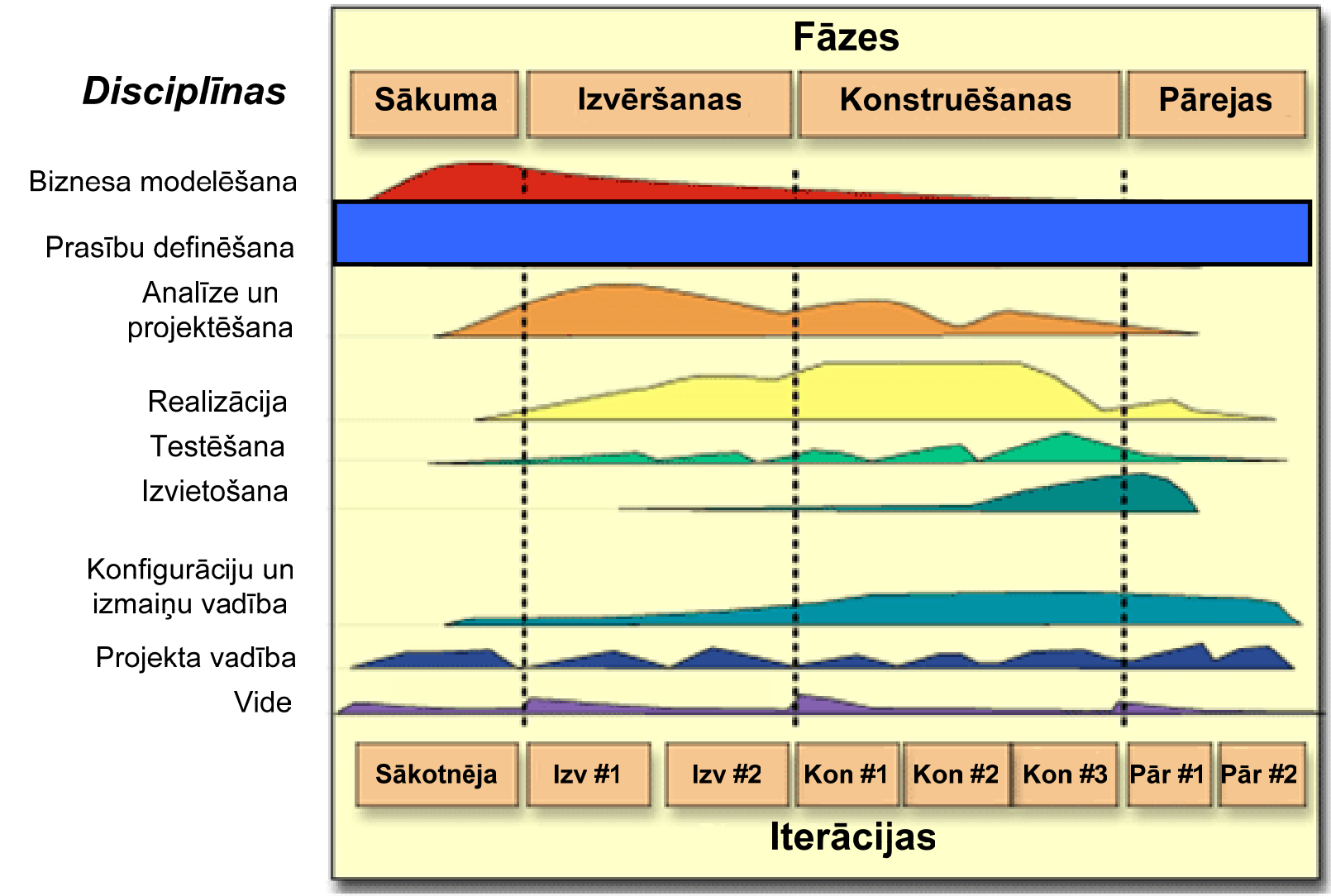
RUP disciplīnas: biznesa modelēšana (1)



RUP disciplīnas: biznesa modelēšana (2)

- Biznesa modelēšanas disciplīnā biznesa procesi tiek dokumentēti izmantojot t.s. biznesa lietošanas piemērus (use cases). Tas ļauj visām ieinteresētajām pusēm kopīgi saprast, kā un kādus organizācijas biznesa procesus ir nepieciešams atbalstīt

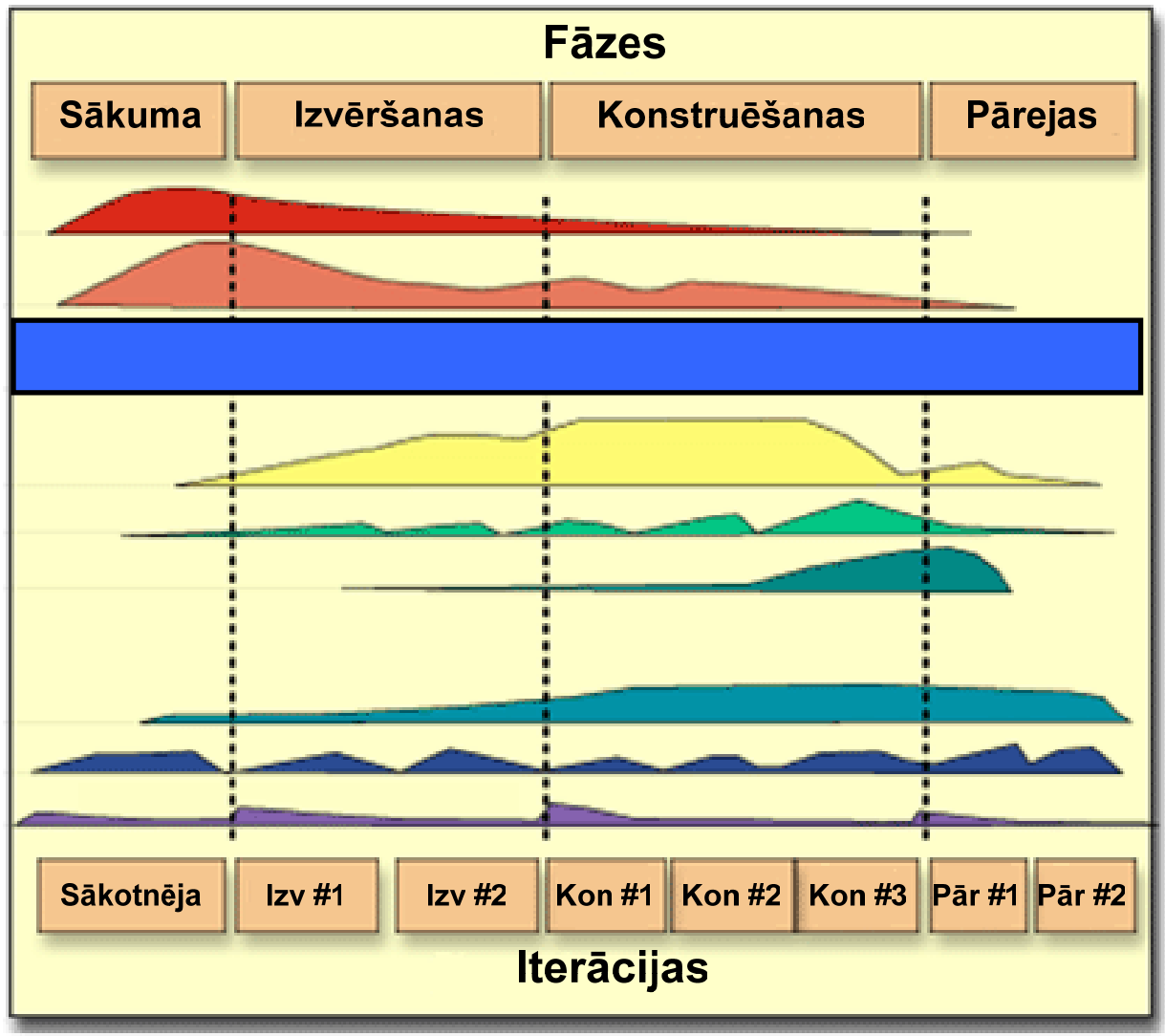
RUP disciplīnas: prasību definēšana (1)



RUP disciplīnas: prasību definēšana (2)

- Disciplīnas mērķis ir aprakstīt, kas ir jādara sistēmai, dot izstrādātājiem un klientiem iespēju vienoties par šo aprakstu.
- Tiek izveidots vīzijas dokuments, kurā ir noskaidrotas visu ieinteresēto pušu vajadzības; tiek identificēti visi aktieri, kuri atbilst sistēmas lietotājiem, kā arī visas citas sistēmas, kuras var mijiedarboties ar izstrādājamo sistēmu; tiek izveidoti lietošanas piemēri, kuri attēlo sistēmas uzvedību.

RUP disciplīnas: analīze un projektēšana (1)



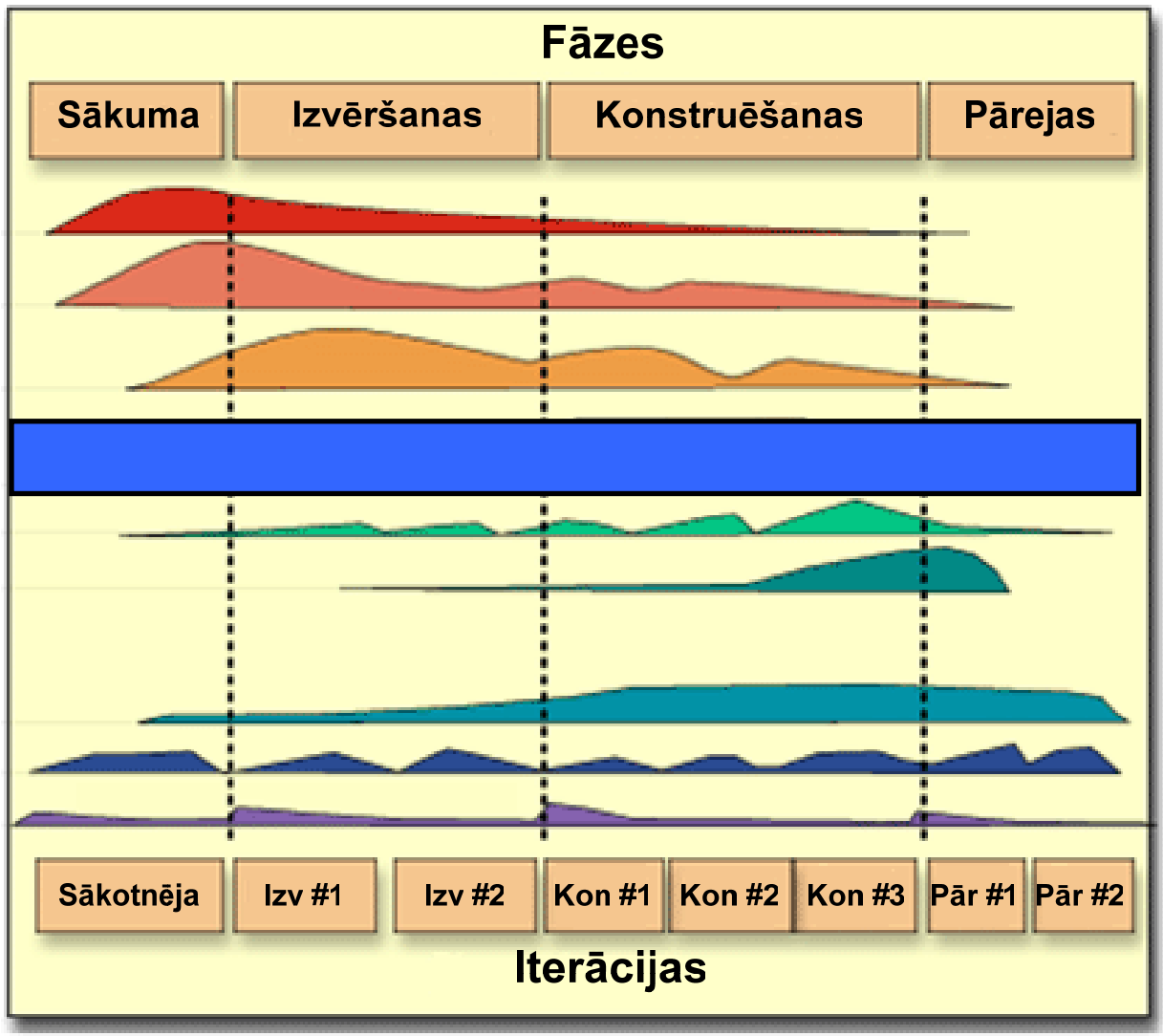
RUP disciplīnas: analīze un projektēšana (2)

- Šīs fāzes galvenais mērķis ir parādīt, kādā veidā sistēma tiks implementēta realizācijas fāzē
- Analīzes un projektēšanas rezultāti tiek attēloti projektēšanas modelī (design model). Projektēšanas modelis tiek izmantots kā programmatūras pirmkoda abstrakcija jeb plāns, kas parāda, kā pirmkods tiek uzrakstīts un strukturēts.

RUP disciplīnas: analīze un projektēšana (3)

- Projektēšanas modelis sastāv no klasēm, kuras:
  - Ir strukturētas
  - Ir ievietotas pakotnēs un apakšsistēmās ar skaidri noteiktiem interfeisiem, kas, savukārt parāda, kuras no tām kļūs par komponentēm
  - Parāda, kā šo klašu objekti sadarbosies savā starpā, lai veiktu lietošanas piemērus
  - Ir cieši saistītas ar sistēmas arhitektūru

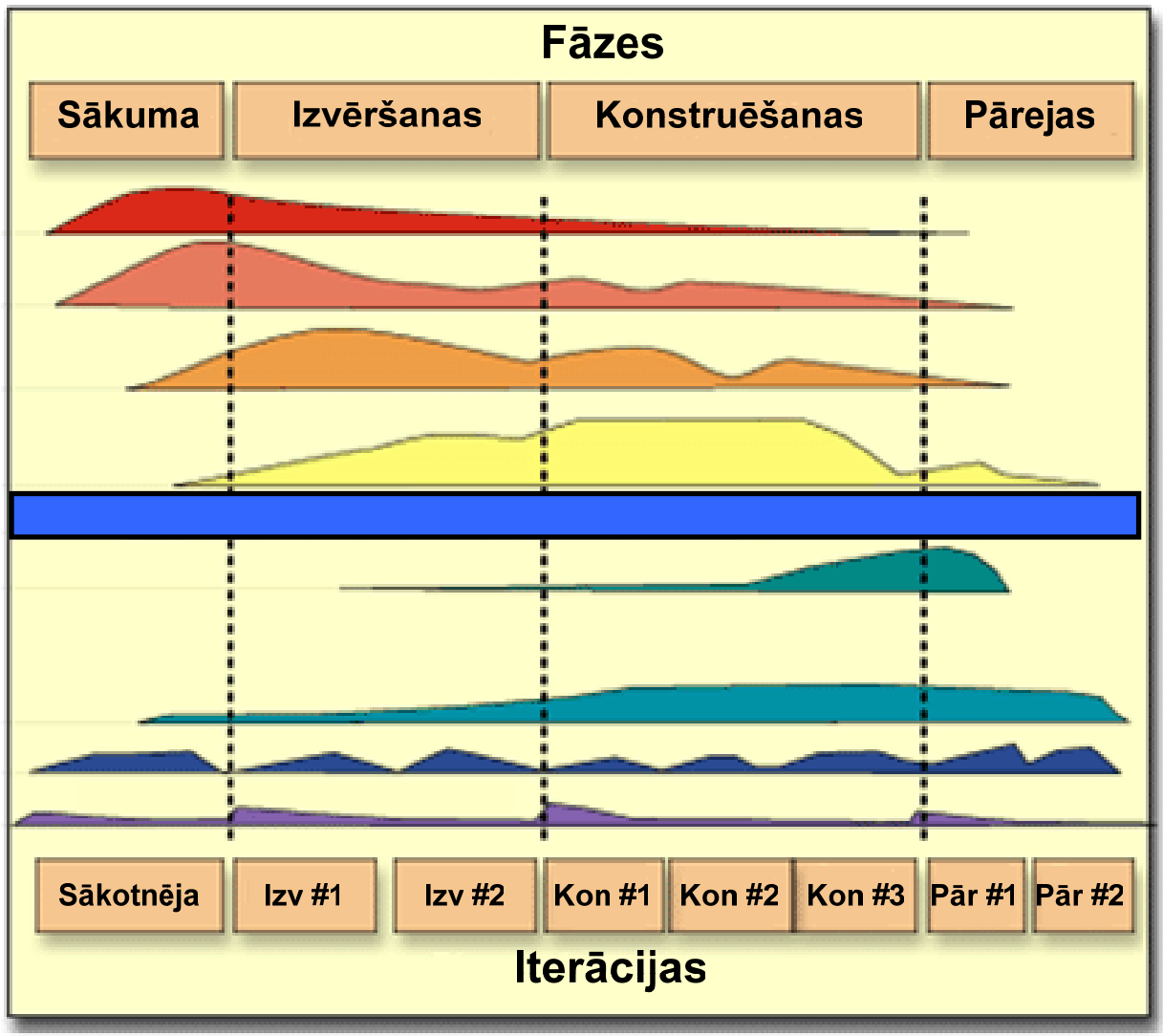
RUP disciplīnas: realizācija (1)



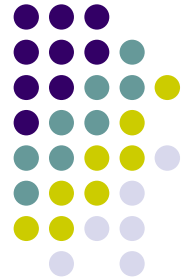
RUP disciplīnas: realizācija (2)

- Kādā veidā ir jārealizē jaunas vai atkārtoti jāizmanto eksistējošās komponentes, lai izstrādātu viegli uzturamu sistēmu. Visas komponentes ir strukturētas.
- Mērķi:
  - Definēt pirmkoda struktūru kā apakšsistēmu implementāciju slāņu veidā
  - Realizēt klases un objektus kā komponentus (binārie faili, pirmkoda faili, izpildāmie faili u.c.)
  - Testēt izstrādātas komponentes
  - Integrēt visu izstrādātāju vai to komandu iegūtus rezultātus galvenajā izpildāmajā sistēmā

RUP disciplīnas: testēšana (1)

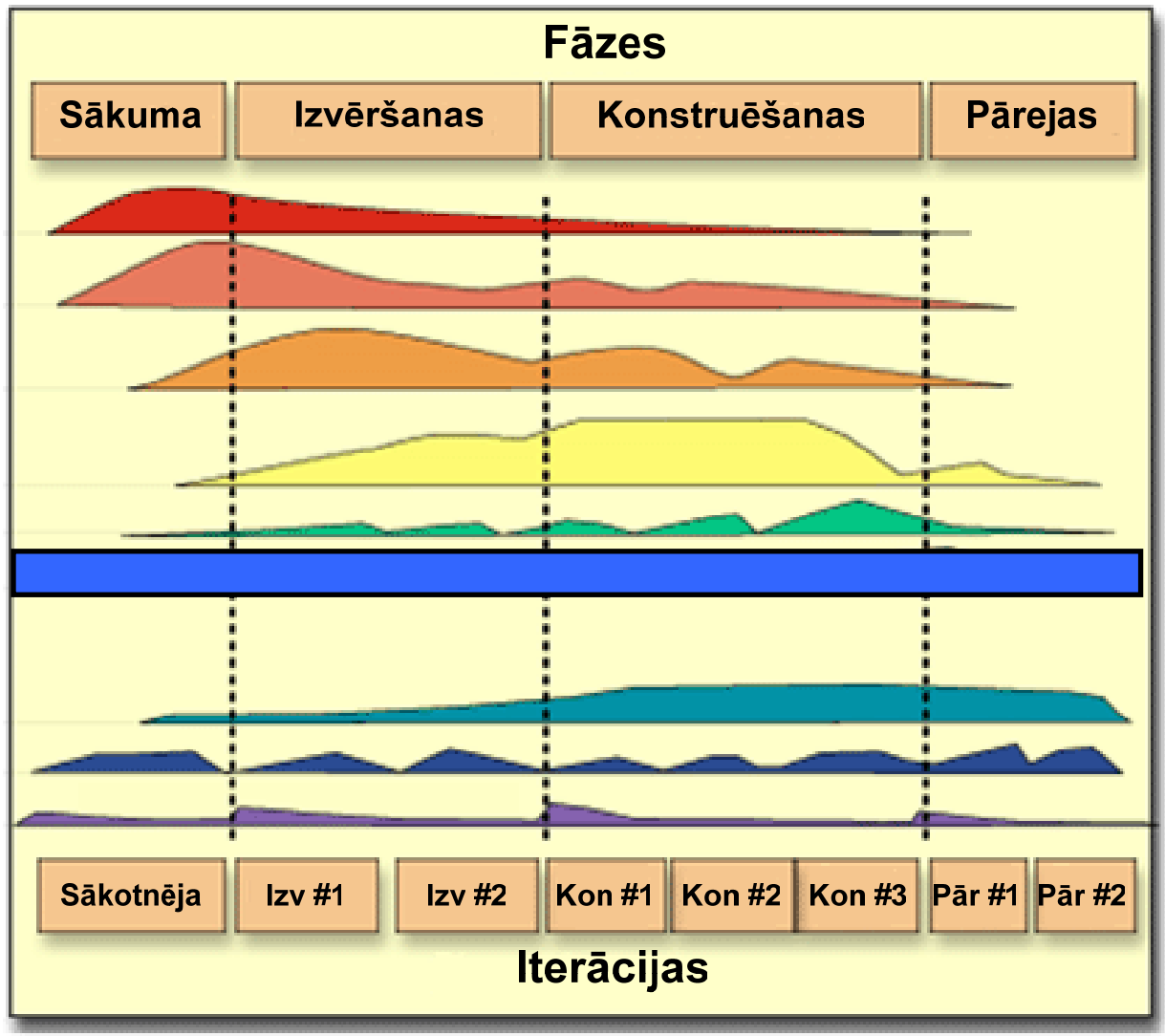


RUP disciplīnas: testēšana (2)

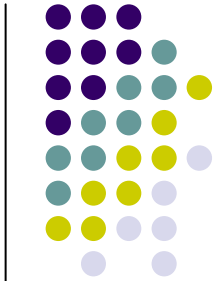


- Mērķi:
  - Pārbaudīt objektu mijiedarbību
  - Pārbaudīt programmatūras komponenšu integrāciju
  - Pārbaudīt, lai visas prasības būtu korekti realizētas
  - Identificēt visus defektus pirms programmatūras izvietošanas
- Testu automatizācija
- Testu kvalitātes dimensijas: drošums, funkcionalitāte, programmas veiktspēja un sistēmas veiktspēja

RUP disciplīnas: izvietošana (1)

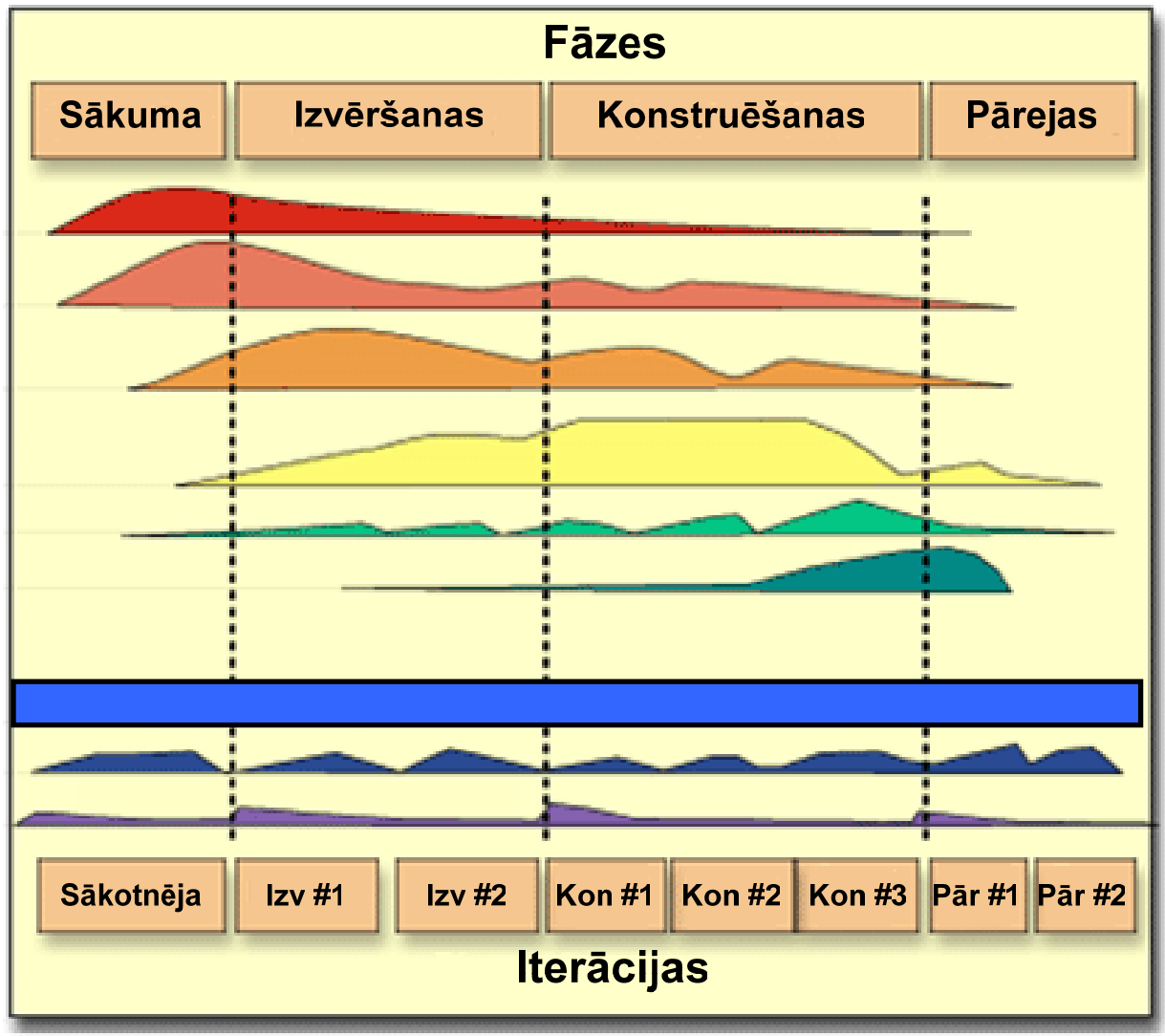


RUP disciplīnas: izvietošana (2)



- Pamata aktivitātes:
  - Produkta ārējo laidienu izveidošana
  - Programmatūras iepakošana
  - Programmatūras izplatīšana
  - Programmatūras instalēšana
  - Lietotāju atbalsta un palīdzības nodrošināšana
- Papildus aktivitātes:
  - Beta testu plānošana un vadīšana
  - Eksistējošas programmatūras vai datu migrācija
  - Formālā akceptēšana.

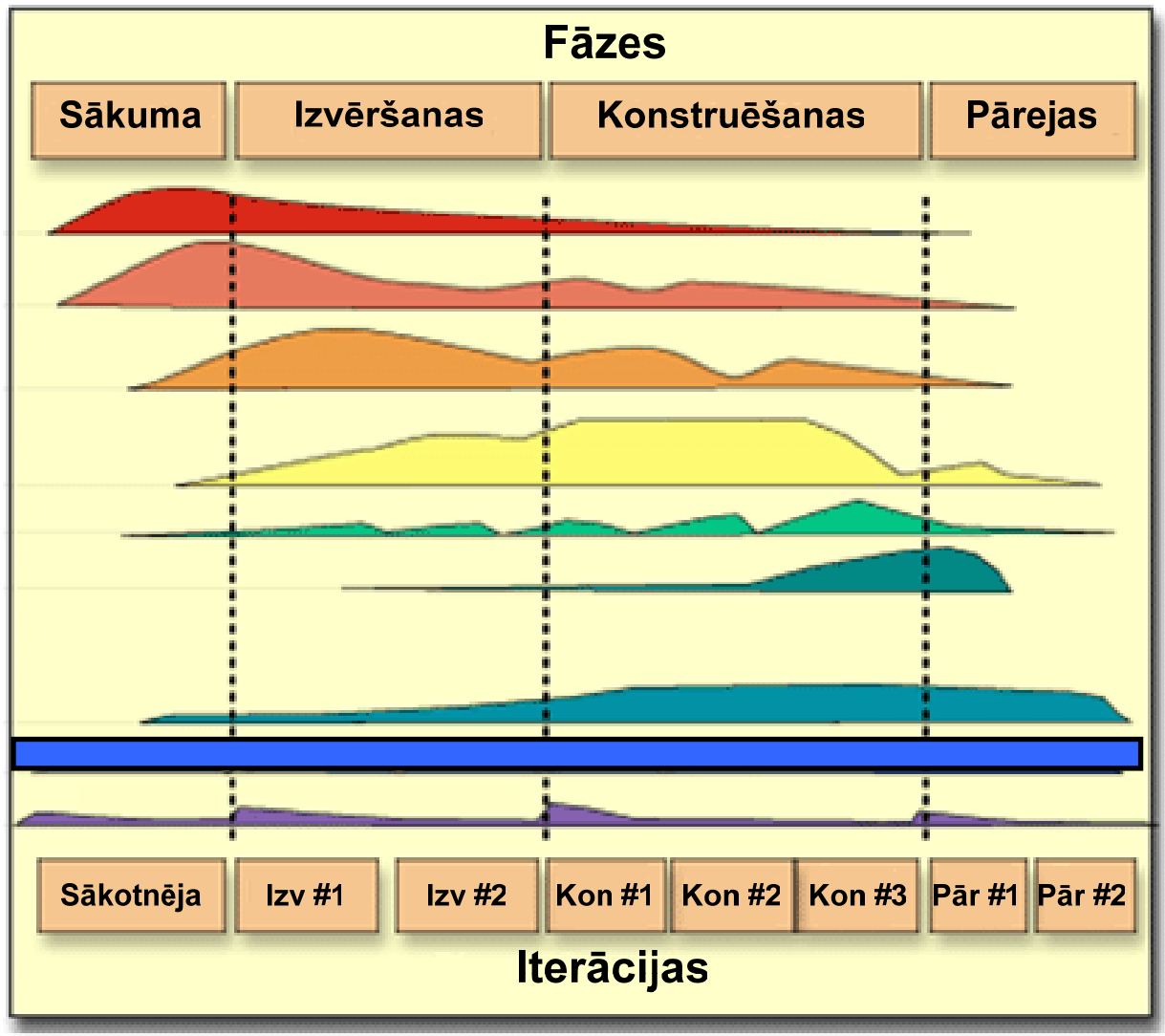
RUP disciplīnas: konfigurāciju un izmaiņu vadība (1)



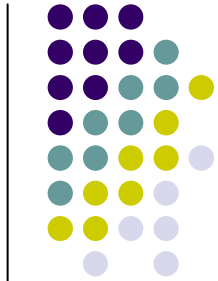
RUP disciplīnas: konfigurāciju un izmaiņu vadība (2)

- Šī disciplīna apraksta, kā kontrolēt lielu artefaktu skaitu, kurus izveido vairāki cilvēki, kas piedalās vienā projektā. Kontrole ļauj izvairīties no lieliem izdevumiem un garantē, ka artefakti nekonfliktē viens ar otru
- Tiek ievērota:
  - Vienlaicīga atjaunošana
  - Ierobežota paziņošana
  - Saliktas versijas

RUP disciplīnas: projekta vadība (1)



RUP disciplīnas: projekta vadība (2)

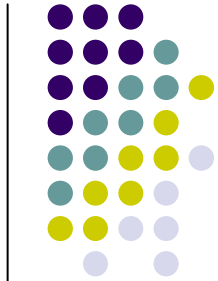


- Šī disciplīna fokusējas galvenokārt uz iteratīvā izstrādāšanas procesa specifiskiem aspektiem
- Disciplīnas mērķi ir nodrošināt:
  - Programmatūras projektu pārvaldīšanas karkasu
  - Praktiskās vadlīnijas projektu plānošanu, izpildīšanu un uzraudzību, kā arī personāla komplektēšanu
  - Risku pārvaldības karkasu izveidošanu

RUP disciplīnas: projekta vadība (3)

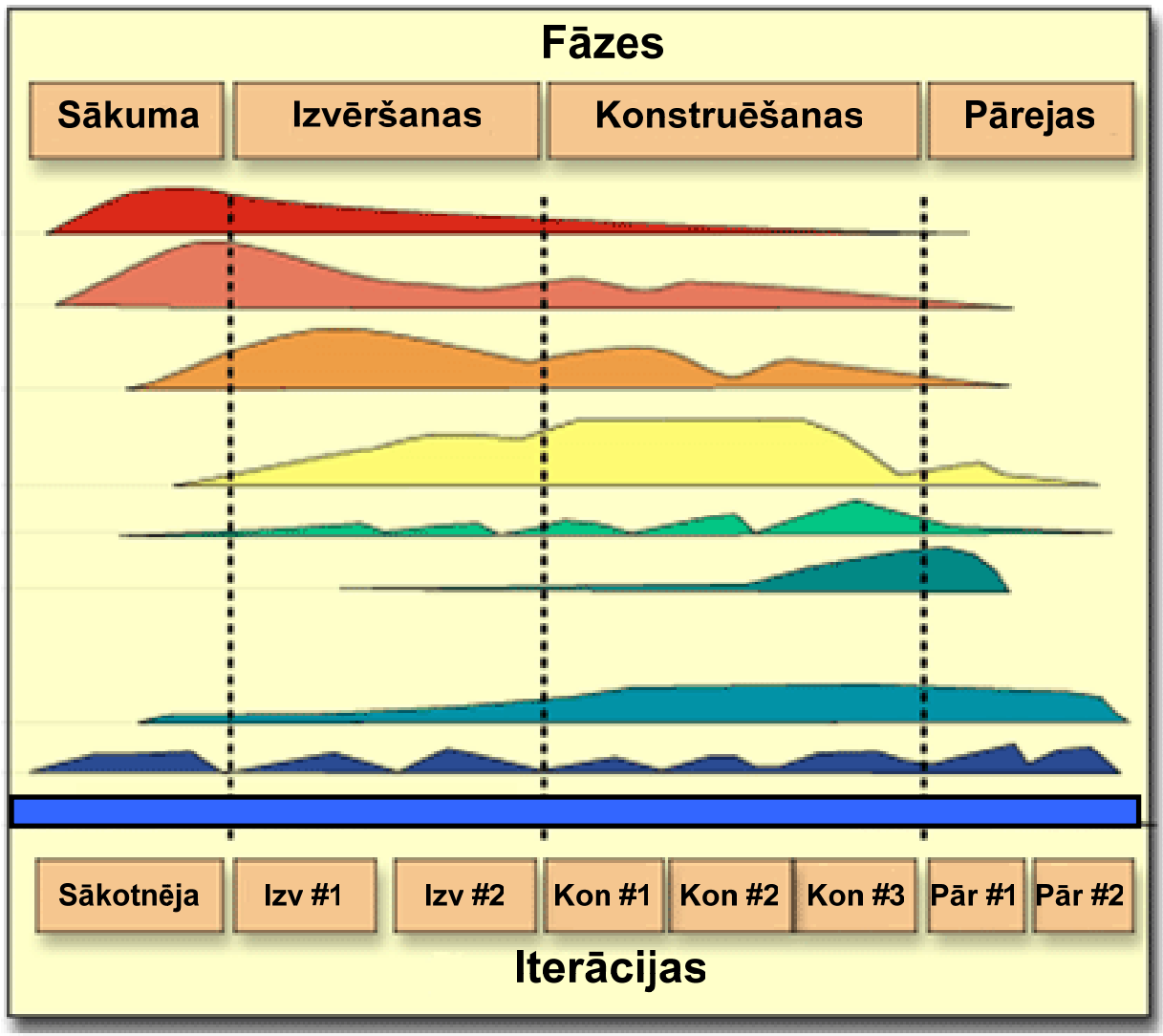
- Disciplīnas mērķis nav apskatīt visus projekta vadības aspektus, piemēram:
  - Personāla vadība (salīgšana, apmācība)
  - Budžeta vadība (definēšana, piešķiršana)
  - Kontraktu vadība (ar piegādātājiem, pasūtītājiem utt.)
- Projekta vadības plāni:
  - Fāžu plāns
  - Iterāciju plāns

RUP disciplīnas: projekta vadība (4)



- Fāžu plāns
  - Mērījumu plāns (izmanto metrikas progresa novērošanai)
  - Risku vadības plāns
  - Risku saraksts
  - Problēmu risināšanas plāns
  - Produkta pieņemšanas plāns
- Iterāciju plāns
  - Pašreizējas iterācija plāns
  - Nākamas iterācijas plāns

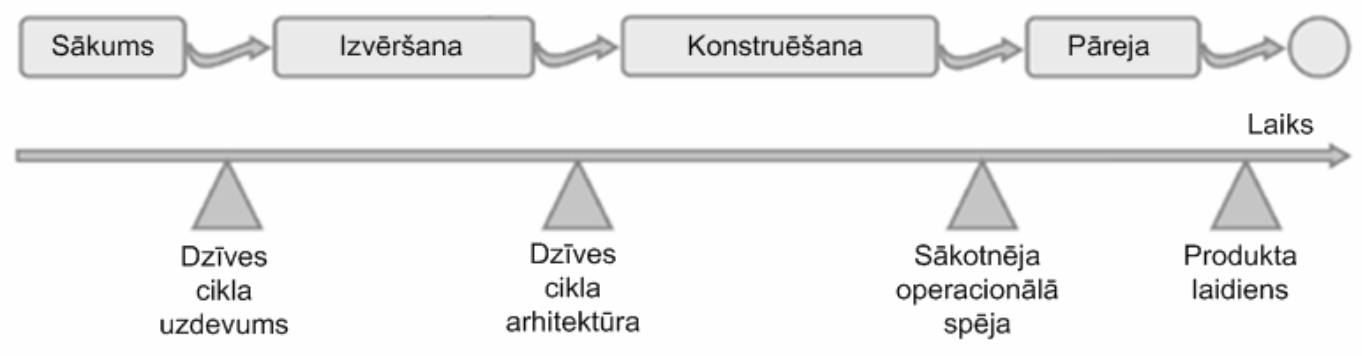
RUP disciplīnas: vide (1)



RUP disciplīnas: vide (2)

- Šīs disciplīnas mērķis ir nodrošināt programmatūras izstrādes organizāciju ar programmatūras izstrādāšanas vidi – t.i. ar procesiem un rīkiem, kuri ir nepieciešami projekta komandas atbalstīšanai.
- Vides disciplīna fokusējas uz:
  - Procesa atbalstīšanu
  - Vadlīniju izstrādāšanu

Dinamiskā struktūra (1)

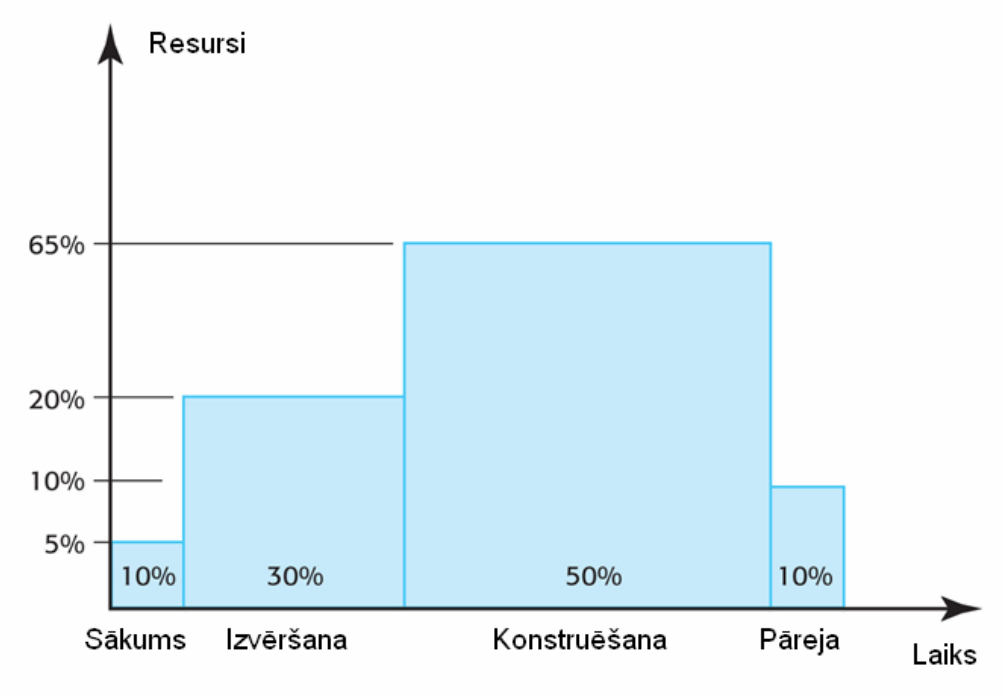


Pamatelementi:

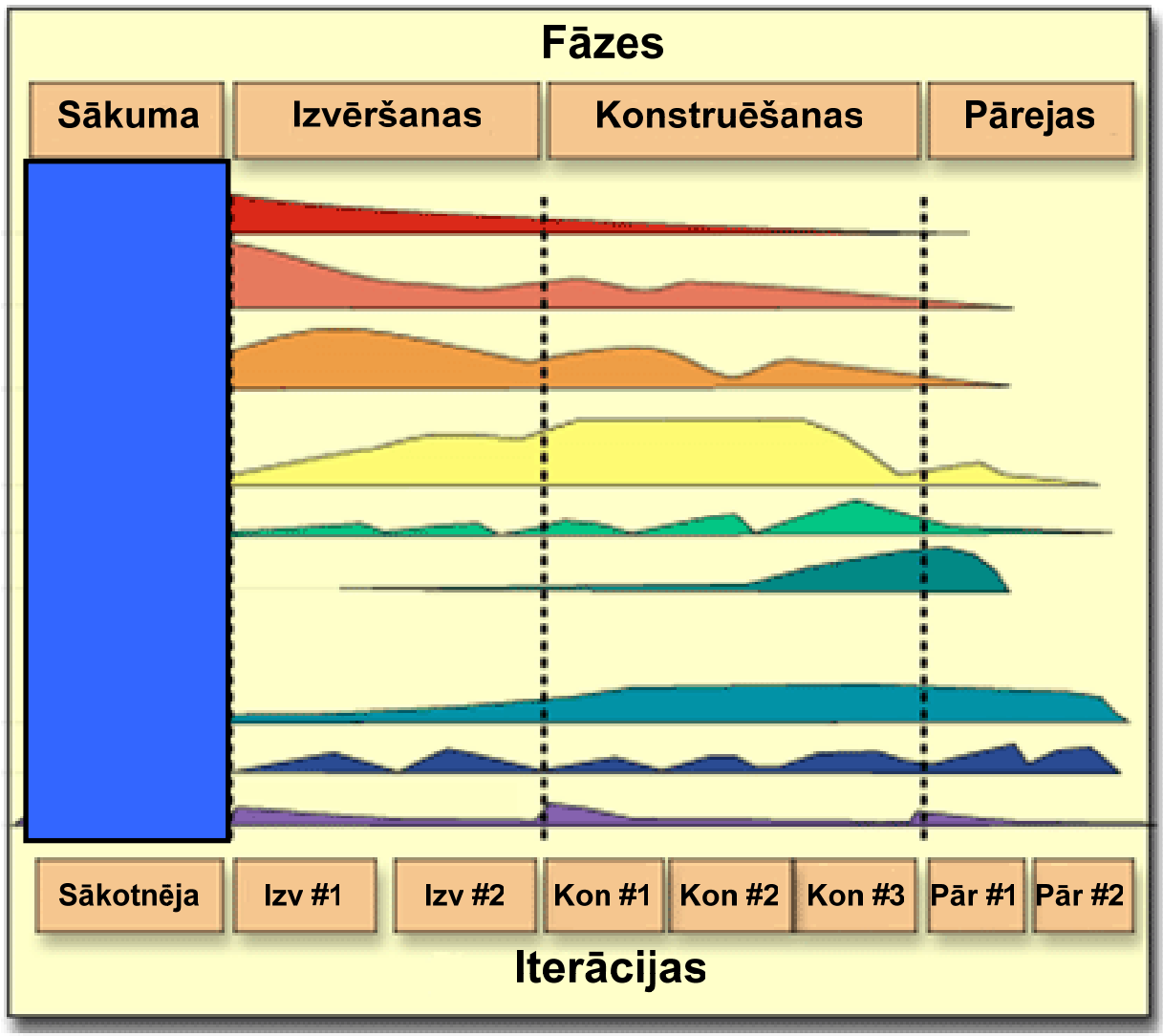
- Fāze - laika periods starp diviem pagrieziena punktiem, kurā tiek sasniegti noteikti mērķi un pieņemts lēmums par jaunās fāzes uzsākšanu
- Pagrieziena punkts - iterācijas vai fāzes formālās beigas, kas atbilst produkta laidienam

Dinamiskā struktūra (2)

Fāžu plānošana



RUP fāzes: sākuma fāze (1)



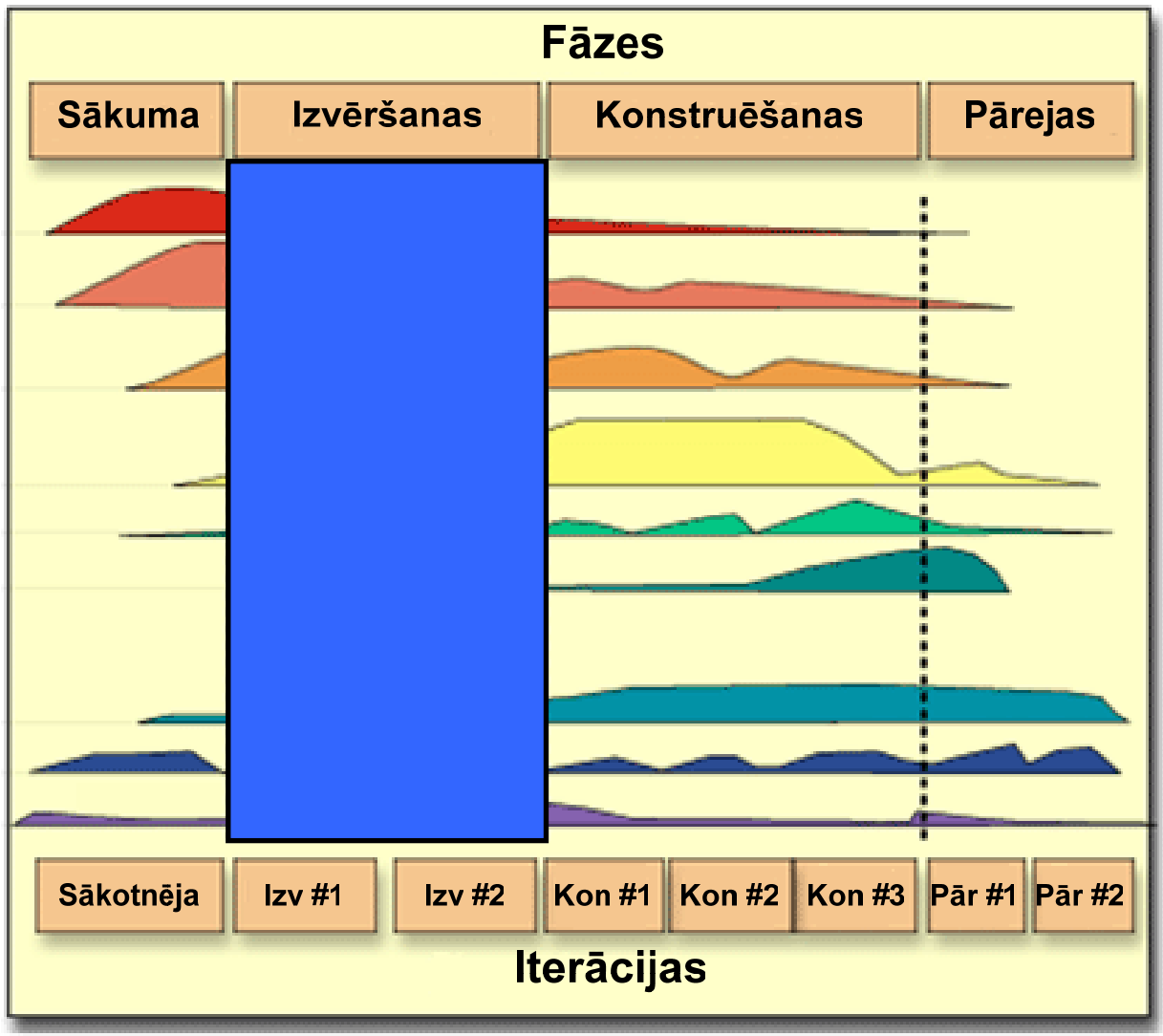
RUP fāzes: sākuma fāze (2)

- Mērķi:
  - Biznesa plāna sastādīšana
  - Risku novērtēšana
- Jāatbild uz jautājumiem:
  - Vai programmatūras ieviešanas peļņa pārsniegs programmatūras izstrādes izmaksas?
  - Vai programmatūra tiks ieviesta tirgū pietiekami ātri, lai šo peļņu saņemtu?
- Sākuma fāzes darbības:
  - Sistēmas konteksta saprašana
  - Sistēmas lietošanas apgabala definēšana
  - Sistēmas funkcionālo un nefunkcionālo prasību uzmetums
  - Iespējamo risku samazināšana
  - Sākotnēja projekta plāna izstrāde

RUP fāzes: sākuma fāze (3)

- Sākuma fāzes rezultāts:
  - Sistēmas konteksta modelis (biznesa procesi un konceptuālais modelis)
  - Sistēmas realizācijas un testēšanas uzmetumi, kas var turpmāk būt izmainīti
  - Sistēmas darbības konceptuālais prototips
  - Bāzes risku saraksts un prasību prioritāšu saraksts
  - Biznesa plāns ar projekta vides izstrādes aprakstu un veiksmes kritēriju novērtēšanu (prognozējama peļņa, tirgus izpēte, projekta novērtēšana)

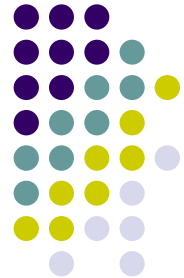
RUP fāzes: izvēršanas fāze (1)



RUP fāzes: izvēršanas fāze (2)

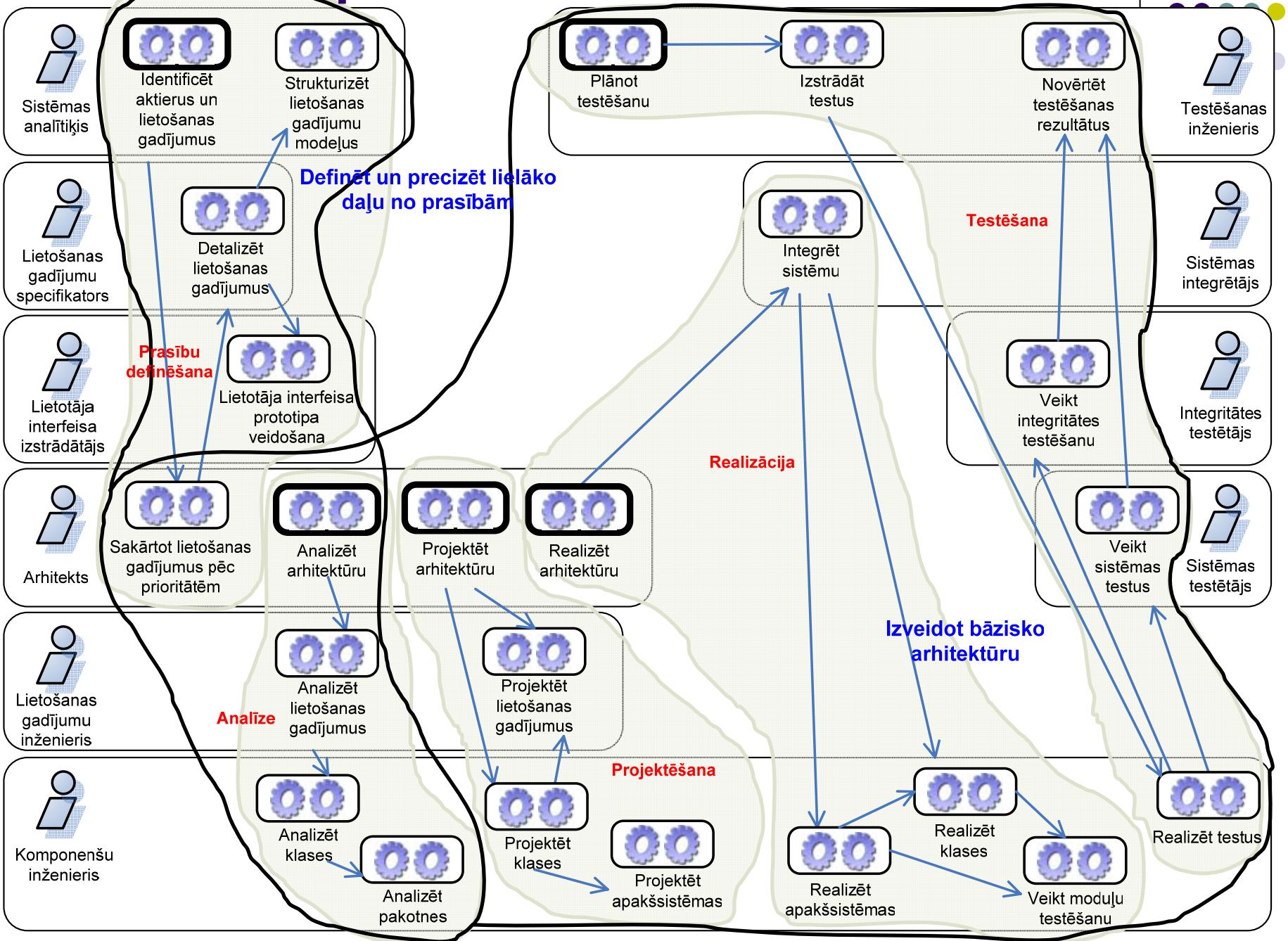
- Mērķi:
  - Noteikt pārejas prasības (kas nebija atklātas sākuma fāzē)
  - Uzsākt arhitektūras veidošanu
  - Turpināt novērtēt riskus
- Izvēršanas fāzes darbības:
  - Projekta izstrādes komandas veidošana Izstrādes vides modificēšana (pielāgošana) atbilstoši projekta būtībai
  - Prasību apstrāde:
    - Vai visas prasības ir nodefinētas?
    - Vai to detalizācijas pakāpe atbilsts vajadzībām?
  - Iespējamo risku samazināšana
  - Biznesa plāna atbilstības pārbaude

RUP fāzes: izvēršanas fāze (3)

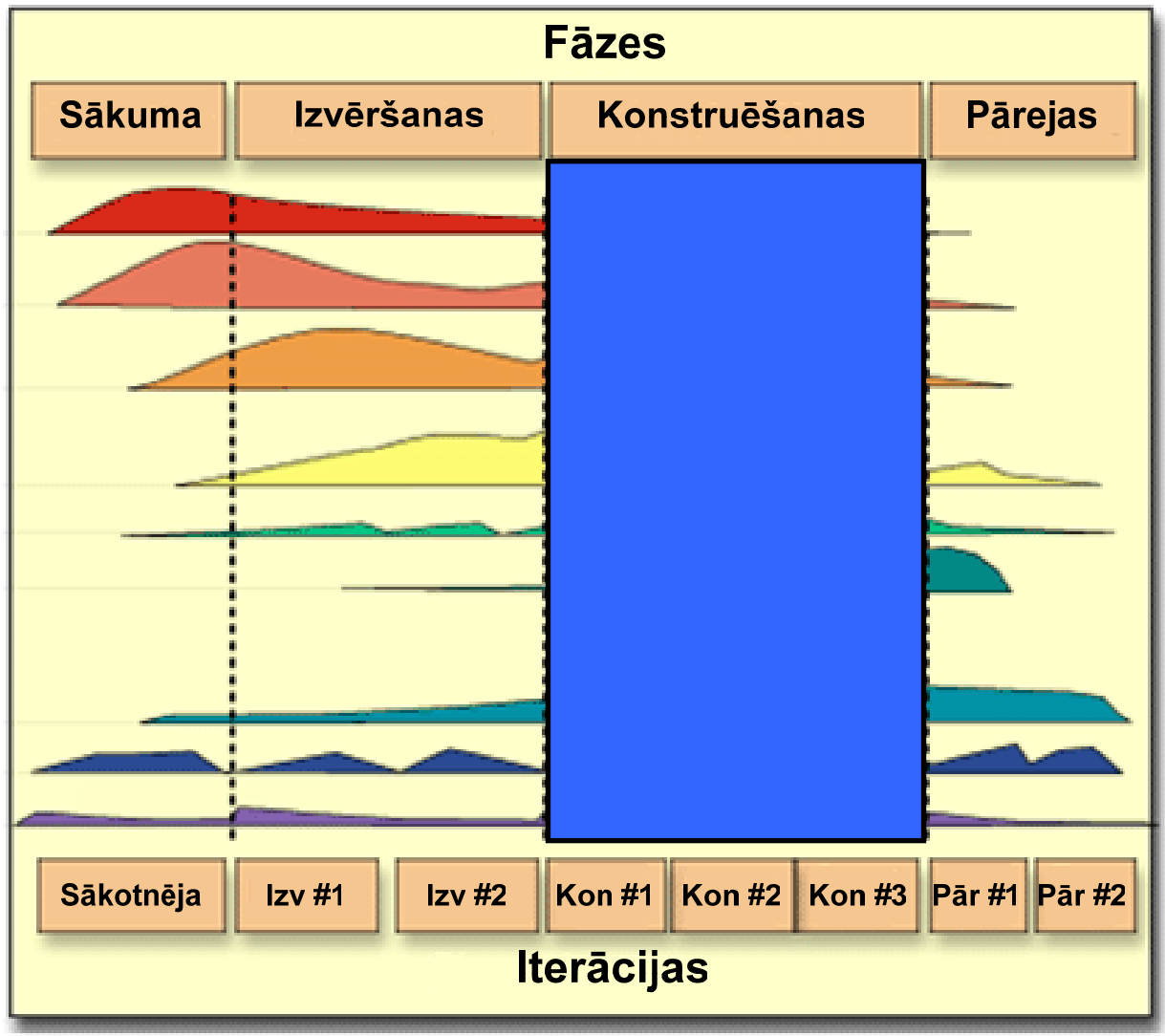


- Izvēršanas fāzes rezultāts
  - Sistēmas konteksta pilns modelis (biznesa procesi un konceptuālais modelis)
  - Sistēmas analīzes un projektēšanas jauns modelis (dažādu sistēmas aspektu diagrammas UML valodā)
  - Sistēmas realizācijas un testēšanas uzmetumi, kas var turpmāk būt izmainīti
  - Sistēmas lietošanas sākotnēja interfeisa prototips
  - Sākotnēja sistēmas lietošanas instrukcija
  - Pārstrādāts risku saraksts
- Pabeigts biznesa plāns ar projekta vides izstrādes aprakstu un veiksmes kritēriju novērtēšanu (prognozējama peļņa, tirgus izpēte, projekta novērtēšana)

RUP fāzes: pamatdarbības izvēršanas fāzē



RUP fāzes: konstruēšanas fāze (1)



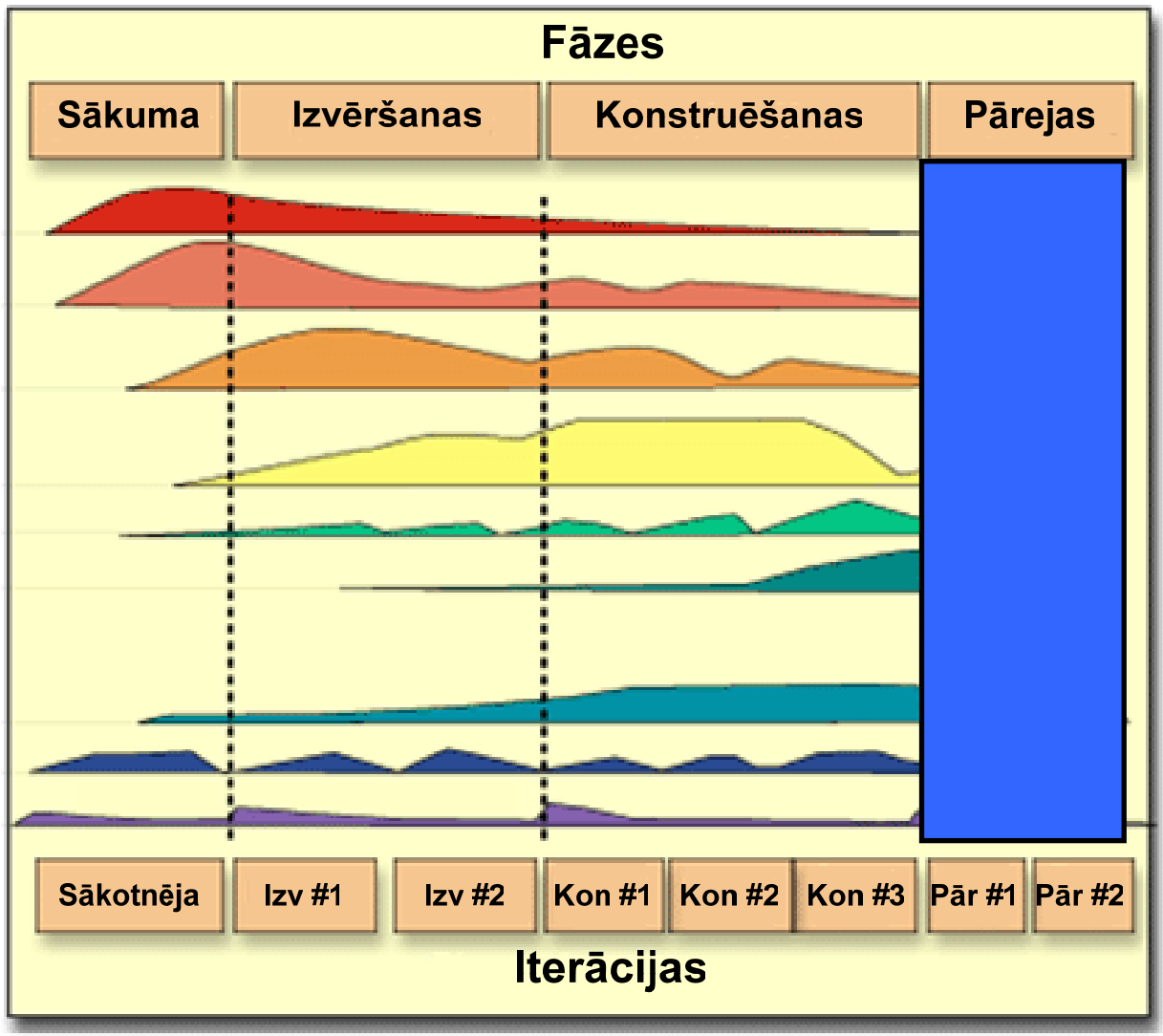
RUP fāzes: konstruēšanas fāze (2)

- Mērķi:
  - Radīt programmas produktu, kas ir gatavs sākotnējai ieviešanai un ß-testēšanai
- Konstruēšanas fāzes darbības:
  - Atlikušo prasību daļas definēšana (parasti ap 20% no visām prasībām)
  - Interfeisa prototipa veidošana
  - Sistēmas analīzes un projektēšanas modeļa bagātināšana
  - Sistēmas arhitektūras projektēšana
  - Prioritāro prasību funkcionēšanas realizācija
  - Testēšanas uzdevumu izpilde
  - Kopējas sistēmas testēšana
  - Testa rezultātu novērtēšana

RUP fāzes: konstruēšanas fāze (3)

- Konstruēšanas fāzes rezultāts:
  - Ieviešanas plānošana
  - Pati programmatūra – versija, kas ietver bāzes funkcionēšanu
  - Uzturams un minimāli modificējams sistēmas arhitektūras apraksts
  - Lietošanas instrukcija ß-versijas testēšanai

RUP fāzes: pārejas fāze (1)



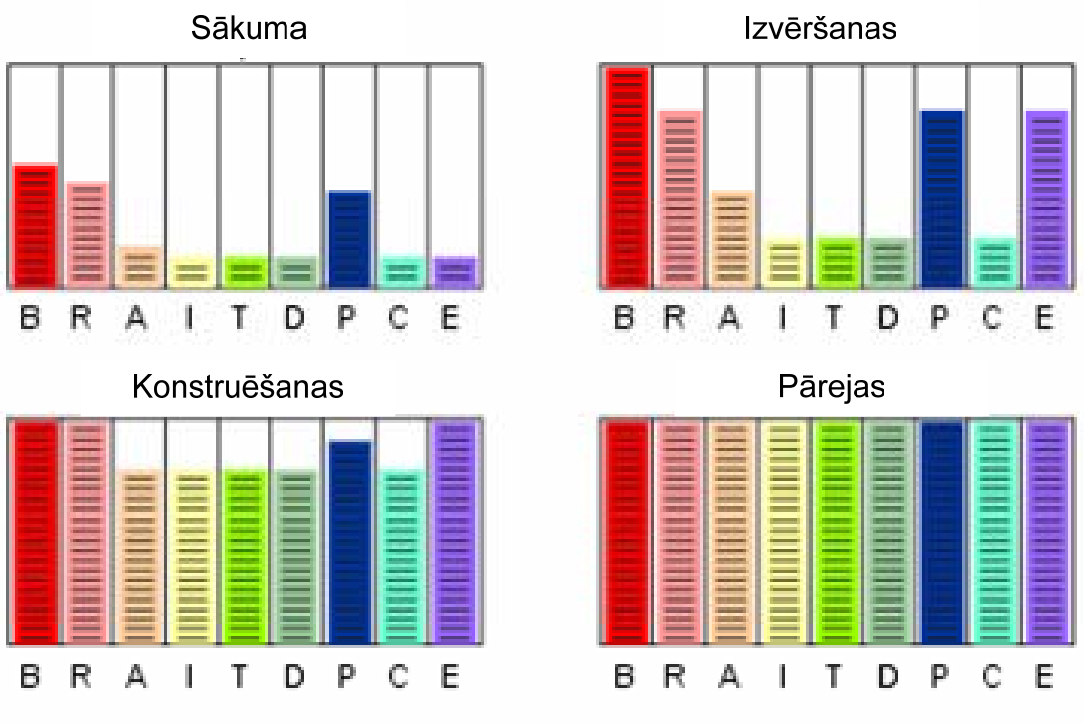
RUP fāzes: pārejas fāze (2)

- Mērķi:
  - Salīdzināt sistēmas funkcionēšanu ar prasībām un noskaidrot lietderīguma pakāpi
  - Apskatīt visus jautājumus, kas ir nepieciešami lietotājiem, lai turpmāk strādātu ar programmatūru, ieskaitot ß-testēšanas rezultātus
  - Sistēmas ieviešana tiek klasificēta divos veidos:
    - Produkts izplatīšanai tirgū, kas var būt ļoti liela mēroga
    - Sistēmas izstrāde pēc pasūtījuma
  - Pieņemt, ka produkti tiek ieviesti ar zināmu defektu daļu, un paredzēt produkta attīstību turpmākās versijās – uzturēšanas aktivitātes

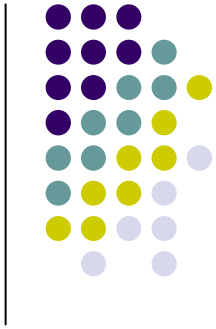
RUP fāzes: pārejas fāze (3)

- Konstruēšanas fāzes rezultāts:
  - Programmatūra ar instalācijas iespējām
  - Juridiskā dokumentācija – līgumi, licences piekrišanās, atteikumi no pretenzijām, garantijas
  - Pilns sistēmas analīzes un projektēšanas modelis
  - Pilns un izlabots sistēmas arhitektūras apraksts
  - Lietotāju, operatoru, sistēmas administratoru instrukcijas, kā arī mācību materiāli
  - Lietotāju atbalsta adreses un Internet lapas, kur var iegūt papildus informāciju par produktu, paziņot par kļūdām un iegūt jaunas programmatūras versijas

Izstrādāto artefaktu daudzums

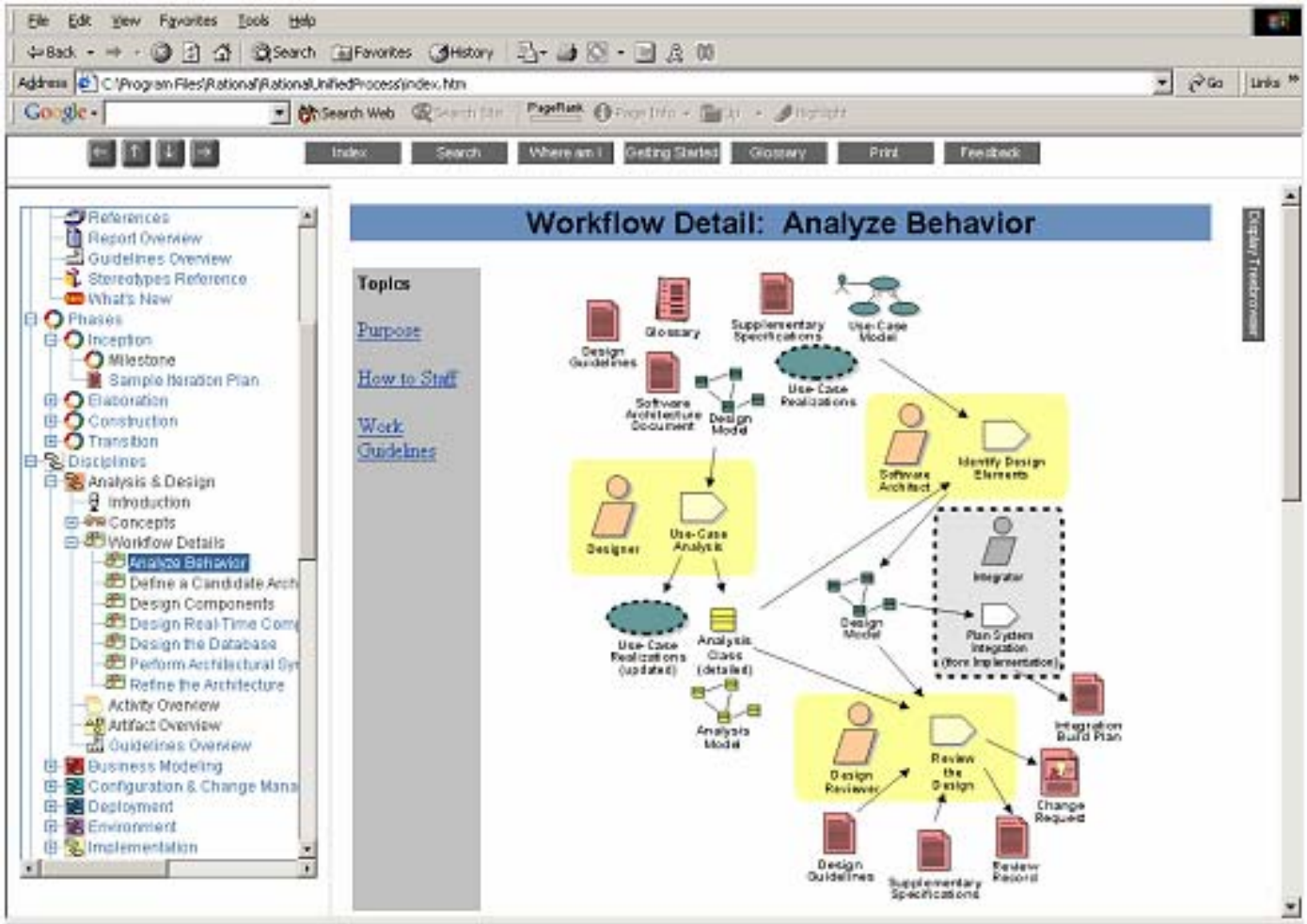


RUP sastāvdaļas



- Tīmekļa zināšanu bāze
  - Plašas vadlīnijas
  - Tool Mentor
  - Rational Rose šabloni un piemēri
  - SoDA šabloni (programmatūras dokumentēšanas automatizācijas rīks)
- Microsoft Project plāni
- Izstrādes darba rīku komplekts
- Pieeja resursu vietnei

RUP tīmekļa zināšanu bāzes piemērs



RUP priekšrocības

- Fokusējas uz izstrādājamā produkta un tā kvalitātes
- Projekta izstrādes risku agrā identificēšana un samazināšana
- Izmaiņu un prasību vadība visa projekta izstrādes laikā
- Viegli modificējams process, kurš var būt formāls vai elastīgs
- Integrēts ar Rational Suite produktu kopu