

levads operāciju pētīšanā
Studiju darbs

Darba mērķis-

**Sprintu plānošanas optimizācija spējās izstrādes
pieejā**

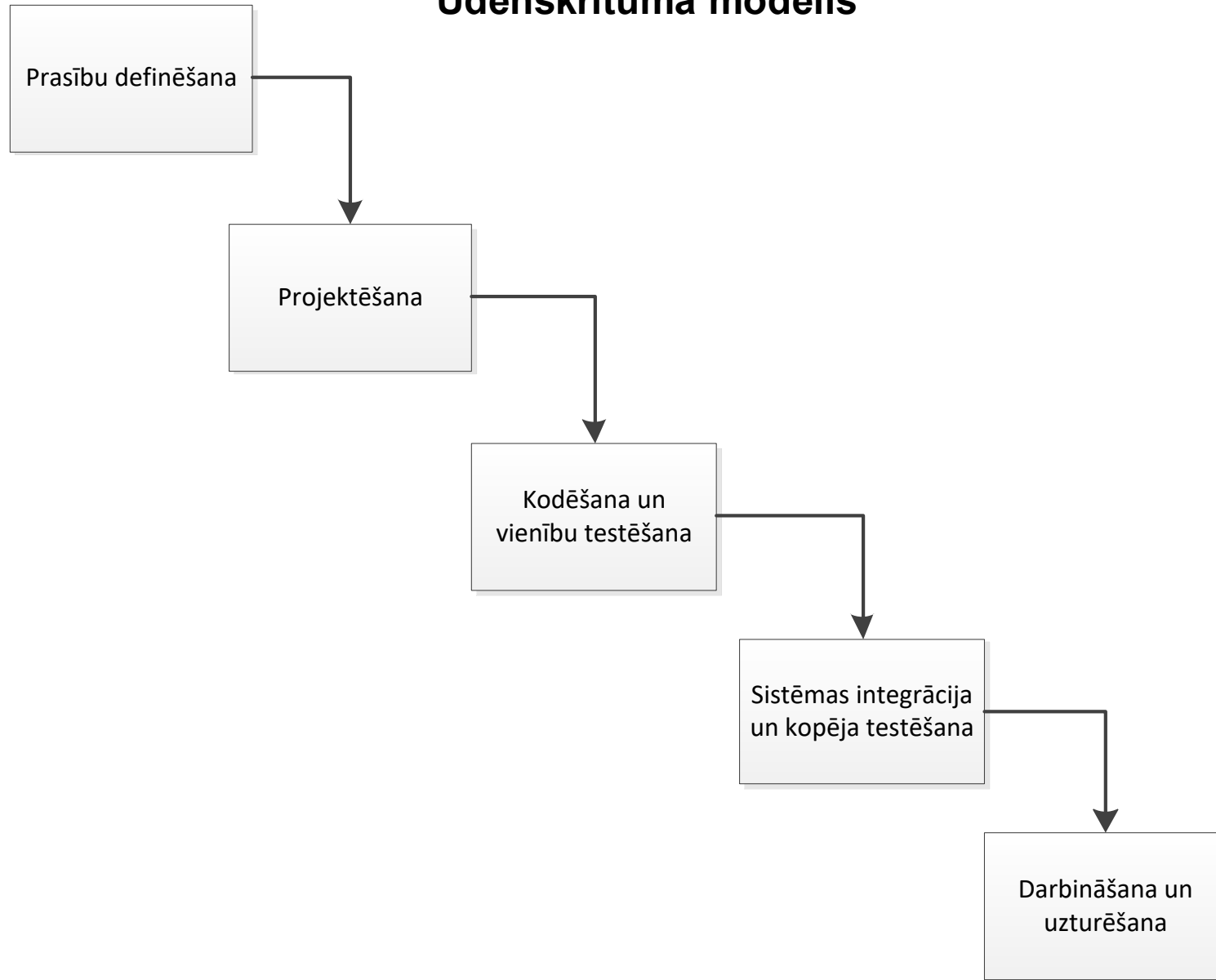
Sprintu plānošanas optimizācija spējās izstrādes pieejā

20. gadsimta vidū, kad sāka attīstīties programmatūras izstrāde, to projekti pārņēma līdzšinējās izstrādes, tā sauktās „klasiskās”, pieejas, kas tika izmantotas ražošanas un celtniecības nozarēs.

Šāda pieeja paredz, ka projekta gaita ir sadalīta saturiski nošķirtos posmos, ko nepieciešams pabeigt, lai uzsāktu nākošo posmu.

Tā kā posmi tiek definēti un saplānoti uzsākot projektu, tas nozīmē ierobežotas, dārgas, laikietilpīgas vai pat nereālas iespējas veikt izmaiņas projekta gaitā.

Ūdenskrituma modelis



90. gadu vidū , neliela daļa pieredzējušu industrijas līderu, kas sekmīgi vadīja savus uzņēmumus sāka piedāvāt inovatīvus risinājumus programmatūras izstrādē, kas pamatā bija balstīti uz sekmīgu izmaiņu vadību.

Šie notikumi ir uzskatāmi par spējās izstrādes pirmsākumiem.

Plašākai sabiedrībai spējās izstrādes termins parādījās 2001. gadā, pēc šo līderu sanāksmes, kuras rezultātā tapa spējās izstrādes manifests, jeb *Agile manifesto*.

Drīz tika laistas klajā pirmās spējās izstrādes principu aprakstošās grāmatas, spējās izstrādes pieejas ieguva lielu piekrišanu un ap 2005. gadu lielās kompānijas sāka ieviest šos principus savā darbībā

Agile manifests

Radot un palīdzot citiem radīt programmnodrošinājumu, mēs nepārtraukti turpinām meklēt un atrodam labākus izstrādes veidus. Savā darbā novērtējam:

- **Cilvēkus un viņu mijiedarbi** augstāk par rīkiem un procesiem
- **Strādājošu programmatūru** augstāk par detalizētu dokumentāciju
- **Kopsadarbību ar pasūtītāju** augstāk par līguma sarunām
- **Reaģēšanu uz izmaiņām** augstāk par sekošanu plānam

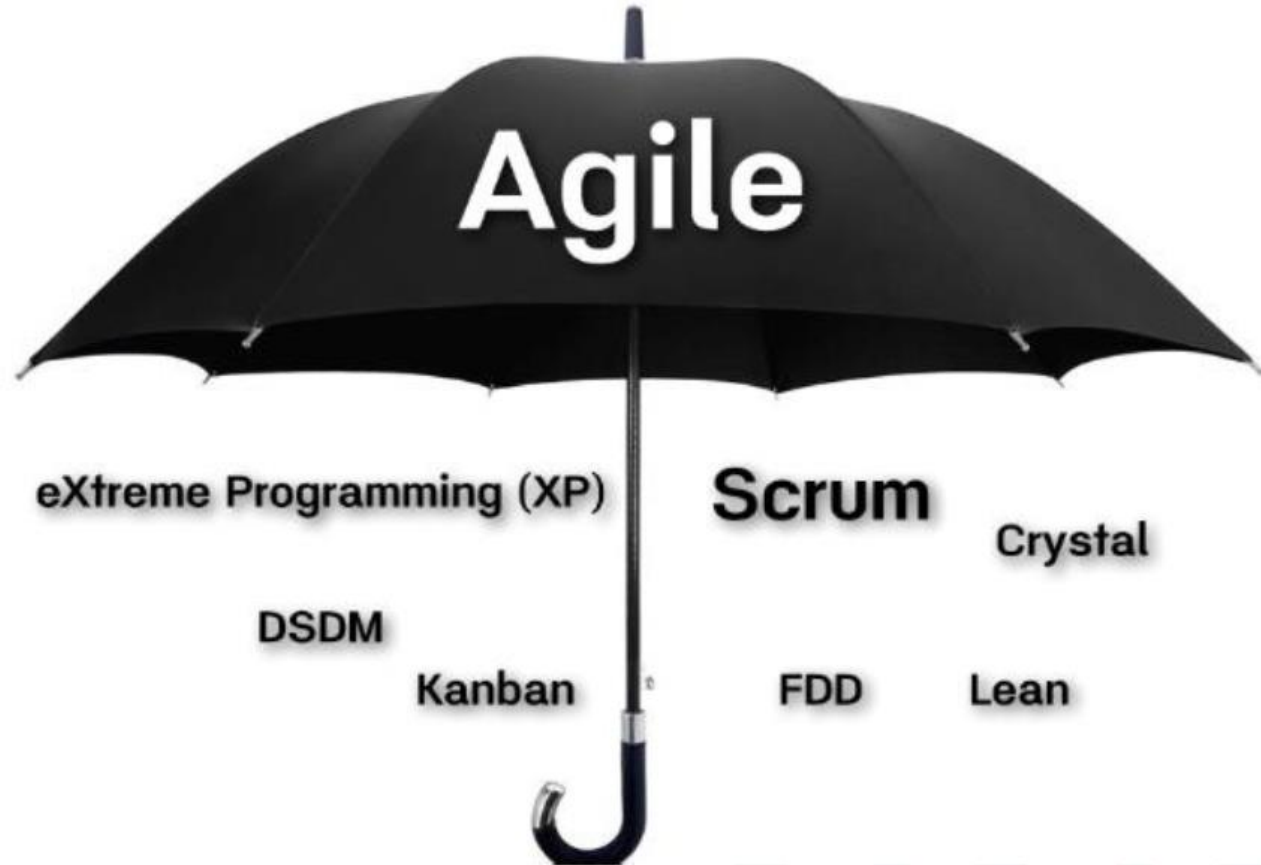
<http://agilemanifesto.org/>

Spējās izstrādes (*Agile*) modelis

- Spējā izstrāde ir metodoloģija, kas satur organizatorisku un metodoloģisku pasākumu kopumu ar mērķi nodrošināt iespējami ātru, elastīgu un efektīvu programmatūras projektu izstrādi un ieviešanu.
- Spējas izstrādes principi balstās uz iteratīviem soļiem - izstrādes projektu visā tā dzīves ciklā realizējot salīdzinoši nelielu, secīgu darba daļu –sprintu veidā.
- Katra soļa sākumā, atkarībā no līdzšinējā progresā, tiek veikta jauna sprinta plānošana un šī posma izpildes gaitā to „no ārpuses” nevar ietekmēt.

Spējā izstrāde pašlaik no programmatūras izstrādes metodoloģijas ir kļuvusi par sava veida etalonu kā vajadzētu vadīt IT uzņēmumu.

Spējās metodoloģijas



Scrum metodoloģija

- *Scrum* ir viena no populārākajām spējās izstrādes metodoloģijām, kas balstās uz inkrementālu izstrādes pieeju, kas paredz izstrādi īsos posmos ar regulārām atskaitēm.
- *Scrum* paredz ļoti lielu caurskatāmību projektam, kas sniedz ātru atgriezenisko saiti vadībai, tādējādi laicīgi ļaujot novērot progresu, kā arī problēmas un attiecīgi rīkoties.
- Pēdējos 10 gados *Scrum* piedzīvojis strauju popularitātes kāpumu un pašlaik ir spējo izstrādes metodoloģiju līderis.

Scrum metodoloģija

- Izstrāde *Scrum* notiek noteikta garuma iterācijās, kas tiek sauktas par **sprintiem**. Katrs sprints ir kā atsevišķa projekta daļa, ar sākumu, un beigām, katrā ietilpst produkta – daļas no kopīgā projekta rezultāta, piegāde. Sprintam ir noteikta kapacitāte – maksimālā darbietilpība.
- Pirms sprintu uzsākšanas notiek uzdevumu sadale un novērtēšana. Novērtēšana palīdz izvēlēties **optimālu uzdevumu skaitu sprintos**, ko komanda būs spējīga piegādāt un to secību.
- Pasūtītāja prasības – uzdevumi, kas jāiekļauj sprintos, tiek saukti par **lietotāja stāstiem** (User story). Katra stāsta izpildes darbietilpību raksturo ar nosacītām vienībām – stāsta punktiem.

Studiju darbā risināmā problēma

Problēma

- Kā optimāli sadalīt veicamos darbus atsevišķos sprintos, lai atdeve no projekta realizācijas (lietderība) būtu maksimāla.
- No optimizācijas viedokļa klasificējama kā **mugursomas problēma** (*knapsack problem*)

Mugursomas problēma (*knapsack problem*)

- Problēmas nosaukums radies pateicoties piemēram par tūristu.
- Pieņemsim, ka tūristam ir nepieciešams papildīt mugursomu ceļojumam. Ir piedāvāti n priekšmeti no kā izvēlēties, kā arī šiem priekšmetiem ir definēta vērtība jeb kaut kāda veida lietojamības novērtējums p_i . Tāpat arī katram no piedāvātajiem priekšmetiem ir noteikts svars w_i .
- Ierobežojums: visu izvēlēto priekšmetu kopējais svars nedrīkst būt lielāks par mugursomai paredzēto maksimālo pieļaujamo svaru - kapacitāti.
- Mērķis: izvēlēties priekšmetus tā, lai to kopējā noderība ceļojuma laikā būtu maksimāla, bet tajā pašā laikā, netiktu pārsniegta mugursomas kapacitāte.

Mugursomas problēma (*knapsack problem*)

- Lai modelētu lemšanas procesu izmanto bināru mainīgo x_i , kas katram izvēlētam priekšmetam pieņem vērtību $x_i = 1$, bet pretējā gadījumā $x_i = 0$.

Standarta mugursomas problēma (*The standard knapsack problem (SKP)*) :

$$\max \sum_{i=1}^n p_i x_i$$

Nosacījums:

$$\sum_{i=1}^n w_i x_i \leq C$$

$$x_i \in \{0, 1\} \quad \forall i$$

kur n = objektu skaits;

p_i = i-tā objekta lietderības raksturojums;

w_i = i-tā objekta svara novērtējums;

C = resursa (mugursomas) kapacitātes novērtējums;

x_i = i-tā objekta lemšanas mainīgais;

Mugursomas problēma (*knapsack problem*)

$$\max \sum_{i=1}^n p_i x_i$$

Nosacījums:

$$\sum_{i=1}^n w_i x_i \leq C$$

$$x_i \in \{0, 1\} \quad \forall i$$

kur n = objektu skaits;

p_i = i-tā objekta lietderības raksturojums;

w_i = i-tā objekta svara novērtējums;

C = resursa (mugursomas) kapacitātes novērtējums;

x_i = i-tā objekta lemšanas mainīgais;

Vairāku mugursomu problēma

- Vairāku mugursomu problēmas (*Multiple knapsack problem (MKP)*) pamatā ir pieņēmums, ka ir objektu kopa ar n lietām, kā arī ir m skaits mugursomu jeb resursu.
- Katram resursam ir sava kapacitāte c_i . Problēmas mērķis aizpildīt visus resursus ar izvēlētajiem objektiem tā, lai summārais ieguvums tiek maksimizēts, kā arī netiek pārsniegta katra resursa kapacitāte.
- Šajā gadījumā binārais mainīgais tiek apzīmēts kā x_{ij} , kur i – atbilst konkrētais objekts, bet j – apzīmē konkrēto resursu. Mainīgais var pieņemt vērtību 0, ja i -tais objekts netiek iekļauts j -tajā resursā, kā arī 1 pretējā gadījumā.
- Šī problēma definējama arī kā vispārīgā norīkojumu problēma (**generalized assignment problem – GAP**)

Vairāku mugursomu problēma (arī GAP)

$$\sum_{i=1}^n \sum_{j=1}^m p_{ij} x_{ij} \rightarrow \max$$

Nosacījumi:

$$\sum_{j=1}^m w_{ij} x_{ij} \leq C_i \quad i = 1, \dots, n$$

$$\sum_{i=1}^n x_{ij} \leq 1 \quad j = 1, \dots, m$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, n \quad j = 1, \dots, m$$

kur n = objektu skaits;

Studiju darba optimizācijas modelis

- **Plāns** – sprintu secība (katrā sprintā izpildāmi noteikti uzdevumi)
- **Sprints** – laikā ierobežota iterācijas vienība, tipiski vienas līdz četru nedēļu garumā, atkarībā no projekta sarežģītības un riskantuma. Sprints ietver noteiktu uzdevumu - lietotāju stāstu (user story) kopumu, un parasti noslēdzas ar rezultāta piegādi. Maksimālo katra sprinta ilgumu (kapacitāti) nosaka projekta komanda.
- **Lietotāja stāsts (User story)** – relatīvi maza daļa no lietotāja prasībām funkcionalitātei. Tas satur diezgan vispārīgu specifikāciju, kas vēlāk var tikt detalizēta pateicoties nepārtrauktai saziņai ar lietotāju; tajā pašā laikā tas dod pietiekami skaidru prasību raksturojumu, lai novērtētu tā izstrādes sarežģītību.
- **Stāsta punkts (story point)** - lietotāja stāsta izstrādes sarežģītības mērvienība. Projekta komandas dalībnieki, balstoties uz savu pieredzi, zināšanām un projekta specifiku, piešķir stāsta punktus katram lietotāja stāstam. Stāsta punkti tiek lietoti kā veicamā darba laika/darbietilpības abstrakts novērtējums, tā izvairoties no subjektīviem un nesalīdzināmiem vērtējumiem. Tipiski lietotāja stāsta komplicētība svārstās starp 1 un 10 stāsta punktiem.
- **Lietderība (utility)** - lietotāja stāsta biznesa vērtība no lietotāja viedokļa. Tās vērtība tiek izteikta kvantitatīvi izmantojot noteiktu skaitlisku skalu (parasti svārstās no 10 līdz 100).

Studiju darba optimizācijas modelis

- **Risks** – ka projekts netiks pabeigts kā paredzēts. Riska rādītājs tiek piesaistīts lietotāja stāstam kā **ietekmes** uz projekta veiksmīgu iznākumu raksturojums. Tādējādi *kritisks* stāsts ir tas, kuram ir stipra ietekme uz citiem stāstiem tā, ka nepareiza vai pavirša tā izpilde var izraisīt projekta izgāšanos (piem., konceptuālais dizains). Tālāk izskatītajā modelī riska novērtējumam tiek izmantoti pozitīvi skaitļi; Tiek izdalītas četras riska klases: 1 (bez riska), 1.3 (zems risks), 1.7 (vidējs risks) un 2 (augsts risks). Risku var traktēt arī kā stāsta prioritātes klasi – jo augstāka prioritāte, jo agrāk tas izpildāms.
- **Līdzība, radniecība (affinity)** – korelācijas pakāpe starp atsevišķiem lietotāja stāstiem. Līdzīgu stāstu iekļaušana vienā sprintā dod augstāku lietderību, jo lietotāji pozitīvāk uztver pilnīgākas piegādātās funkcionalitātes biznesa vērtību. Piemēram, stāsts "papildu datu ieguve" pats par sevi var būt ar zemu lietderību, bet tā lietderība palielinās, ja to piegādā kopā ar stāstu "papildu datu ielāde". „Radniecības pakāpes” diapazons tiek pieņemts [0,0.5], kas nozīmē, ka stāstu lietderība var palielināt ne vairāk kā par 50%.
- **Atkarība (dependence)** – izstrādes nosacījums savstarpēji saistītiem lietotāja stāstiem, kas nozīmē, ka kāda stāsta izstrāde nevar tikt uzsākta, pirms otrs nav pabeigts. Izmantojot spējas programmizstrādes metodes, lai uzlabotu projekta elastību attiecībā uz lietotāju vēlmēm, dažas lietotāju stāstu atkarības izstrādē obligāti jā saglabā (piemēram, loģiskā modeļa izstrādē jāievēro konceptuālais modelis). Lietotāja stāsta atkarība - pirms tā jābūt izpildītiem visiem ar to saistītajiem „pirms-stāstiem”.
- **Izstrādes ātrums** – Novērtēts stāsts punktu skaits, ko komanda var piegādāt atsevišķa sprinta ietvaros. Tas izsaka sprinta ilgumu kā sprinta kapacitāti (maksimālais sprintā izpildāmais stāstu punktu skaits).

Studiju darba optimizācijas modelis

Modelī iekļaujamie nosacījumi

- Lietotāja apmierinātība – lietotāja stāstus ar augstāku lietderību vēlams izpildīt un piegādāt pirmos, tādējādi palielinot lietotāja izpratni un uzticību.
- Radniecības menedžments – radniecīgos stāstus vēlams iekļaut vienā sprintā, lai palielinātu to vērtību lietotājam.
- Riska vadība – kritisko stāstu, kuri var izraisīt vēlāku nevēlamu efektu, izstrādi veic uzmanīgāk, lai būtu pietiekams laiks korekciju veikšanai; to var panākt izvietojot riskantos stāstus agrākos sprintos vai sprintos ar lielāku kapacitāti.

Studiju darba optimizācijas modelis

Modeļa formalizācija

Ir dota m sprintu kopa S un n lietotāju stāstu kopa U .

Pieņemtie apzīmējumi:

- $x_{ij} = 1$, ja j -tais stāsts iekļauts i -tajā sprintā; 0, ja nav iekļauts;
- u_j - j -tā stāsta lietderības raksturojums;
- p_j – j -tā stāsta punktu skaits;
- p_i^{\max} – i -tā sprinta kapacitāte (izteikta stāsta punktos);
- r_j – j -tā stāsta riska, jeb ietekmes uz projekta iznākumu vērtējums;
- a_j – j -tā stāsta „radniecības pakāpe”;
- Y_j – j -tajam stāstam radniecīgo stāstu kopa;
- y_{ij} – palīgmainīgais, kas norāda to kopas Y_j stāstu skaitu, kas iekļauti sprintā i ;
- D_j – to stāstu kopa no kuriem j -tais stāsts ir atkarīgs;

Optimizācijas modelis

Mērķa funkcija

$$Q = \sum_{i=1}^m \sum_{j=1}^n u_j \left(r_j x_{ij} + a_j \frac{y_{ij}}{|Y_j|} \right) \rightarrow \max$$

$x_{ij} = 1$, ja j -tais stāsts iekļauts i -tajā sprintā; 0, ja nav iekļauts;

u_j - j -tā stāsta lietderības raksturojums;

r_j - j -tā stāsta ietekmes vērtējums;

a_j - j -tā stāsta „radniecības pakāpe”;

$|Y_j|$ - j -tajam stāstam radniecīgo stāstu kopas apjoms

Optimizācijas modelis

Nosacījumi

- Sprinta ietilpība

$$\sum_{j=1}^n p_j x_{ij} \leq p_i^{max} \quad \forall i \in S$$

- Stāsta iekļaušana

$$\sum_{i=1}^m x_{ij} = 1 \quad \forall j \in U$$

Optimizācijas modelis

Nosacījumi

- Atkarība – stāstu nedrīkst izpildīt pirms tiem, no kuriem tas ir atkarīgs

$$\sum_{k=1}^i \sum_{z \in D_j} x_{kz} \geq x_{ij} |D_j| \quad \forall i \in S$$

Optimizācijas modelis

Nosacījumi

- Līdzīgu stāstu iekļaušana vienā sprintā ir racionāla

$$y_{ij} \leq \sum_{k \in Y_j} x_{ik} \quad \forall i \in S, j \in U$$

$$y_{ij} \leq |Y_j| x_{ij} \quad \forall i \in S, j \in U$$

Uzdevums

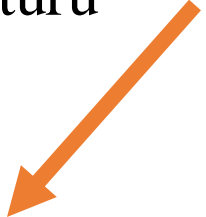
- **Izmantojot apskatītā modeļa principus izveidot optimālu sprintu plānu atbilstoši individuālā varianta nosacījumiem.**
- **(Modelī var tikt veiktas pamatotas izmaiņas)**

Varianta izvēle

- **Variants sakrīt ar studentu apliecības pēdējiem diviem cipariem, no 00 līdz 49. Ja skaitlis ir virs 49 – variants tiek izvēlēts kā dalījums pēc moduļa 50 (piem., 50 atbilst 00, 64 – 14, 92 – 42 u.t.t.).**

Variants

Jāaizpilda pēc savas izvēles,
ņemot vērā lietotāja stāstu
saturu



Sprinta Nr	1	2	3	4	5	6		
Kapacitāte p_i^{\max}	15	15	20	20	25	25		

User Story	Rentabilitāte U_j (10-100)	Story point P_j (1-10)	Ietekme r_j (1;1.3;1.7;2)	Korelācijas ietekme a_j (0-0.5)	Korelējošās US Y_j	Pirmsnorises D_j	
						AND	OR
R.02	85	8					
R.03	80	7					
R.12	70	3					
R.17	75	6					
R.28	35	4					
R.39	45	5					
...							

User story ID	Apraksts
R.01	Kā Datu pārvaldnieks, es vēlos pievienot, labot un dzēst informāciju par galvenajiem sadarbības partneriem, lai uzturētu aktuālu un kvalitatīvu partneru reģistru.
R.02	Kā Tehniskais arhitekts, es vēlos uzturēt informāciju par sistēmas arhitektūras moduļiem, lai pārējiem inženieriem būtu precīza
R.03	Kā Produktu menedžeris, es vēlos ievadīt un labot dizaina elementu datus (ikonas, maketi u.c.), lai komanda lietotu apstiprinātus un jaunākos materiālus.
R.04	Kā Drošības speciālists, es vēlos pārraudzīt lietotāju augšupielādētos failus un dzēst neatbilstošos, lai nodrošinātu drošību un konfidencialitāti.
R.05	Kā UI/UX komandas vadītājs, es vēlos uzturēt prototipu (wireframe, maketu) sarakstu vienotā reģistrā, lai dizaineri strādātu ar vienu oficiālo versiju.

- Detalizēts analītiskā modeļa izveides piemērs atrodams dokumentā
Piem_StudDarba_modelis
- Par datorizētu modeļa risināšanu – praktiskajā nodarbībā