TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Informaatika õppekava

Albert Luckas Wihler

Andmesõlmede toe lisamine eksperimentaalsele automaatsele BPMN paigutustööriistale

Bakalaureusetöö (9 EAP)

Juhendajad: Armin Daniel Kisand,
Heili Orav

Tartu 2025

Andmesõlmede toe lisamine eksperimentaalsele automaatsele BPMN paigutustööriistale

Lühikokkuvõte: Äriprotsessimudel ja -notatsioon (ingl *Business Process Model and Notation*, edaspidi BPMN) on kujunenud oluliseks standardiks äriprotsesside graafilisel modelleerimisel. BPMN-diagramme saab kasutada igal talitluse tasemel, tööalasest kasutajast protsessi teostajateni. Selle töö eesmärk oli täiendada eksperimentaalset BPMN-diagrammide automaatse paigutuse tööriista, lisades tuge andmeobjektide ja andmekogude paigutamiseks. Analüüsiti käsitsi loodud BPMN-diagramme, et tuvastada andmesõlmede paigutuspõhimõtteid, mille alusel töötati välja automaatse paigutuse reeglid. Arendustöö tulemusena lisati tööriistale andmesõlmede tugi, mis suurendab oluliselt selle rakendusvõimalusi. Töö pakub olulise aluse tööriista edasiseks arenduseks tervikliku ja kasutajasõbraliku BPMN-paigutuslahenduse suunas.
Võtmesõnad: BPMN, Sugiyama, paigutus, diagramm
CERCS: P175 Informaatika, süsteemiteooria

Adding support for data nodes to an experimental automatic BPMN layout tool

Abstract: Business Process Model and Notation (BPMN) has become an important standard for the graphical modeling of business processes. BPMN diagrams can be used at every operational level, from business users to process implementers. The aim of this thesis was to enhance an experimental BPMN diagram automatic layout tool by adding support for data objects and data stores. Manually created BPMN diagrams were analyzed to identify placement principles for data nodes, based on which automatic layout rules were developed. As a result of this thesis, support for data nodes was added to the tool, significantly increasing its potential applications. This thesis provides an important foundation for further development of the tool towards a comprehensive and user-friendly BPMN layout solution.
Keywords: BPMN, Sugiyama, layout, diagram
CERCS: P175 Informatics, systems theory

Sisukord

1 Teoreetiline taust **7**
1.1 Sarnased tööriistad . . . 7
1.2 Täisarv-lineaaroptimeerimine . . . 7
1.3 BPMN-diagrammi tutvustus . . . 9
1.4 Sugiyama raamistik . . . 14
1.5 BPMN-diagrammi automaatne paigutamine . . . 16
1.6 Programmi algseis . . . 16
1.6.1 Paigutamine . . . 18

2 BPMN andmesõlmede automaatne paigutamine **19**
2.1 Andmeobjektide ja andmekogude paigutamise põhimõtted . . . 20
2.2 Teostus . . . 22
2.2.1 Domeenispetsiifilise keele laiendused . . . 23
2.2.2 Andmesõlmede kihtidele paigutamine . . . 25
2.2.3 Ajutiste sõlmede leidmine . . . 26
2.2.4 Nooltevaheliste ristumiste vähendamine . . . 27
2.2.5 Y-koordinaatide määramine . . . 28
2.2.6 X-koordinaatide määramine . . . 32
2.2.7 Andmenoolte otseühenduste loomine . . . 32
2.2.8 Noolte marsruutimine . . . 33
2.2.9 Ajutiste sõlmede eemaldamine . . . 34

3 Jõudlustestid **35**

4 Edasine arendus **38**

5 Kokkuvõte **39**

A Sõlmede kihtide määramise ILP **41**

B Algne sõlmede paigutamise pseudokood **42**

C Algne noolte marsruutimise pseudokood **42**

D Andmesõlmede kihtide leidmise ILP **43**
D.1 Sõlmede ja noolte defineerimine . . . 44
D.2 ILP reeglid andmeobjektide ja andmekogude paigutamiseks . . . 44
D.2.1 Indeksid ja parameetrid . . . 45
D.2.2 Otsustusmuutujad . . . 45
D.2.3 Eesmärgifunktsioon . . . 45

D.2.4 Kitsendused 46

E Noolespetsiifiliste kaalude valik **46**

F Automaatselt paigutatud diagrammid **47**

G Valik analüüsitud diagrammidest **55**

H Litsents **58**

Terminid

BPMN (ingl *Business Process Model and Notation*) - graafiline tähistussüsteem äriprotsessiskeemide tarbeks[1].
Andmeobjekt (ingl *data object*) - programmide täitmiseks vajalik andmestruktuuri element, näiteks fail, massiiv või operand; võib olla konstant või muutuja[1].
Andmekogu (ingl *data store*) - organiseeritud ja püsiv otsinguvõimalusega andme- ja teabekogum[1].
Kiht (ingl *layer*) - sõlmedest koosnev vertikaalne kiht BPMN-diagrammis.
Täisarv-lineaaroptimeerimine (ILP) (ingl *integer linear programming*) - lineaaroptimeerimise alaliik, lisakitsendus on kõigi või osa muutujate täisarvulisus[1].
DSL (ingl *Domain-Specific Language*) - domeenispetsiifiline keel.
WASM (ingl *WebAssembly*) - portatiivne kompileerimise sihtkeel, mis võimaldab programme veebirakendustes kasutada.[2]

[1] `https://akit.cyber.ee/`
[2] `https://webassembly.org/`

Sissejuhatus

Äriprotsessimudel ja -notatsioon (ingl *Business Process Model and Notation*, edaspidi BPMN) on kujunenud oluliseks standardiks äriprotsesside graafilisel modelleerimisel. BPMN-diagramme saab kasutada igal talitluse tasemel, tööalasest kasutajast protsessi teostajateni. Traditsiooniliselt on BPMN-diagrammide koostamiseks kasutatud graafilisi kasutajaliideseid. Käesolevas töös käsitletav tööriist võimaldab aga defineerida BPMN-protsesse inimloetava tekstina, mille põhjal genereeritakse automaatselt paigutatud diagramm. Selline lähenemine kiirendab protsessidiagrammide loomist, vähendades vajadust sõlmi käsitsi paigutada - tegevus, mis on eriti ajamahukas suuremate diagrammide puhul või juhul, kui protsessi struktuuris tehakse muudatusi. Diagrammi loogilise struktuuri säilitamiseks tuleb tihti käsitsi ümber korraldada märkimisväärne osa elementidest. Automaatselt paigutatud diagrammid säilitavad ka ühtset paigutusloogikat erinevatel diagrammidel.

Selle töö eesmärk on täiendada olemasolevat prototüüpi, lisades sellele andmesõlmede toe ja täiustades paigutusloogikat. Prototüüp loodi neljases grupis 2024. aasta sügissemestri kursusel Tarkvaraprojekt (LTAT.05.005). Selle töö autor oli grupi liige ning vastutas diagrammide paigutamise eest. Töökäiku saab näha prototüübi originaalse hoidla *layout-datanode*[3] harus.

Eesmärgist lähtuvalt on sõnastatud ka järgmised uurimisküsimused:

Uurimisküsimus 1: Kas leidub kindlaid põhimõtteid kuidas andmesõlmed on paigutatud?

Uurimisküsimus 2: Kuidas peab modifitseerima Sugiyama raamistikku andmesõlmede toe lisamiseks?

Uurimisküsimus 3: Kuidas mõjutab andmesõlmede toe lisamine tööriista jõudlust?

Töö koosneb viiest peatükist. Esmalt tutvustatakse teoreetilist tausta ja tööriista. Teises peatükis kirjeldatakse töökäiku ning kolmandas peatükis jõudlustesте. Eelviimases peatükis loetletakse edasise arenduse võimalusi ning sellele järgneb viimane osa ehk kokkuvõte.[4]

[3] `https://github.com/jensjager/bpmn-parser/tree/layout-datanode`

[4] Selles töös on kasutatud OpenAI ChatGPT 4o mudelit, et parandada töö loetavust. Näiteks viimistleti sõnastust ja parandati kirjavigu.

1 Teoreetiline taust

Antud töö eesmärk on tööriistale andmesõlmede funktsionaalsuse lisamine. Selle saavutamiseks on vaja olemasolevaid lahendusi laiendada ja täiustada. Alljärgnevalt antakse ülevaade sarnastest tööriistadest ja tööriistas kasutusel olevatest tehnoloogiatest ning tööriista algseisust enne arendustööde alustamist.

1.1 Sarnased tööriistad

Sarnaseid tööriistu on varem loodud, mis võtavad teksti sisendi ja väljastavad BPMN diagrammi mingil kujul. Kaks peamist konkurenti, mis selles kontekstis esile tõusevad, on BPMN Sketch Miner[1] ja Farhath Ajmal jt peenhäälestatud GPT mudel[2]. Teised sarnased tööriistad ei toeta otseselt BPMN standardit, näiteks Mermaid.js[5].

BPMN Sketch Miner on tööriist, mille teostus on kõige sarnasem käesolevas töös arendatavale tööriistale. BPMN Sketch Mineriga on võimalik luua BPMN diagramme nende poolt määratletud kitsendatud loomulikus keeles, mille käigus luuakse reaalajas visuaalne diagramm. Tööriist toetab enamikku BPMN elementidest. Kuna tööriista lähtekood ei ole avalik ja paigutusprotsessi ei ole selgitatud, ei saa sellest vahendist palju õppida. Loodud diagrammid on piiratud paindlikkusega. Kui Sketch Mineris loodud diagramm imporditakse teise keskkonda, katkevad nooled, mis ühendavad erinevaid sõlmesid. See juhtub, kuna tööriista loodud sõlmed ei vasta BPMN standardi sõlmede struktuurile. Samuti nooled on üldiselt paigutatud otseühendustena mitte nagu käsitsi loodud diagrammides. Tööriist on saadaval vaid veebiplatvormil[6] ning valminud diagramme saab eksportida SVG, PNG ja BPMN (XML) formaatides.

Farhath Ajmal jt peenhäälestatud GPT mudel omab täiesti teistsugust lähenemisviisi. Selle mudeli puhul loob tehisintellekt vabas keeles kirjeldatud protsessist BPMN diagrammi ja väljastab selle pildina. Selle tööriista kasutamine võib aga viia olukorrani, kus loodud diagrammid ei ole järjekindlad ega vasta alati ootustele. Lisaks ei ole loodud diagrammi lihtne muuta, kuna see on esitatud pildifailina.

1.2 Täisarv-lineaaroptimeerimine

Täisarv-lineaaroptimeermine on lineaaroptimeerimise alaliik (edasipidi ILP), lisakitsendus on kõigi või osa muutujate täisarvulisus. Optimiseerimisel minimeeritakse või maksimiseeritakse sihifunktsiooni. Samuti saab defineerida kitsendusi. ILP mudelite lahendamine on NP-täielik.

[5] `https://mermaid.js.org`

[6] `https://www.bpmn-sketch-miner.ai`

Vaatleme järgmist näidet. Ettevõte toodab kahte erinevat toodet: A ja B. Toote A valmistamine võtab 3 tundi ja annab 4 eurot kasumit. Toote B valmistamine võtab 2 tundi ja annab 8 eurot kasumit. Ettevõttel on tootmiseks kokku maksimaalselt 10 tundi aega. Eesmärk on leida, mitu toodet A ja B tarvitseb toota, et kasum oleks maksimaalne.

Märgistame muutujad:

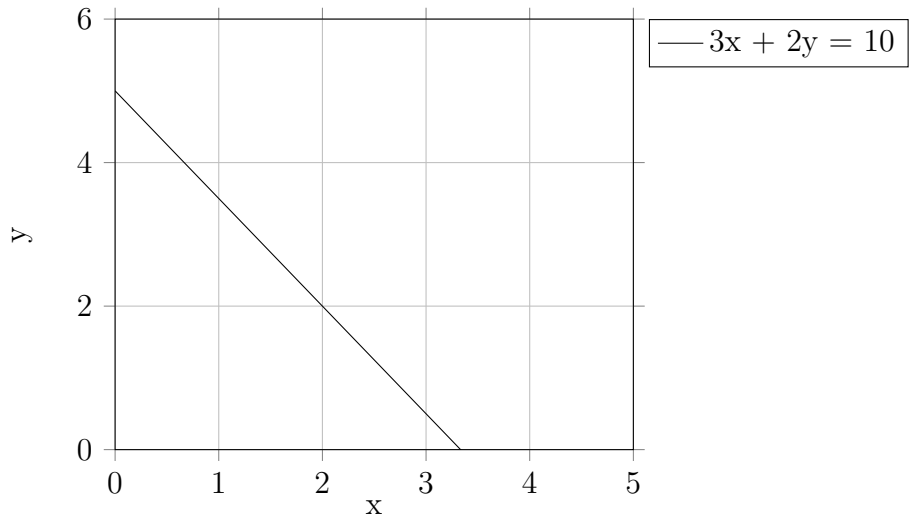
- x – toote A kogus,
- y – toote B kogus.

Formuleerime ülesande järgmise ILP mudelina:

$$\begin{array}{rl} \text{Maksimeeri} & z = 4x + 8y \\ \text{tingimustel} & 3x + 2y \leq 10, \\ & x, y \geq 0, \\ & x, y \in Z, \end{array}$$

kus:

- z tähistab kogukasumit eurodes,
- esimene tingimus tagab, et tootmiseks kuluv aeg ei ületaks 10 tundi,
- x ja y on mittenegatiivsed täisarvud.



Joonis 1: ILP graafiline esitus

Vaadates formuleeritud ILP graafi (vt joonis 1) saab leida optimaalse lahenduse. Leiame et $Z = 40$ kui $(x, y) = (0; 5)$. Antud ILP-d on võimalik lahendada geomeetriliste meetodite abil, kuid keerukamate probleemide korral tuleb kasutada spetsiaalseid lahendusalgooritme ja tööriistu.

Selles töös kasutatakse ILP probleemide lahendamiseks tööriista good_lp[7], mis võimaldab defineerida ILP mudeleid ja lahendada neid erinevate valitavate lahendajatega. Valitud lahendaja on microlp[8], mis ei ole kõige kiirem good_lp toetatud lahendajate seas, aga selle eelis on võimalus kompileerida *WebAssembly*[9] (WASM) formaati.

1.3 BPMN-diagrammi tutvustus

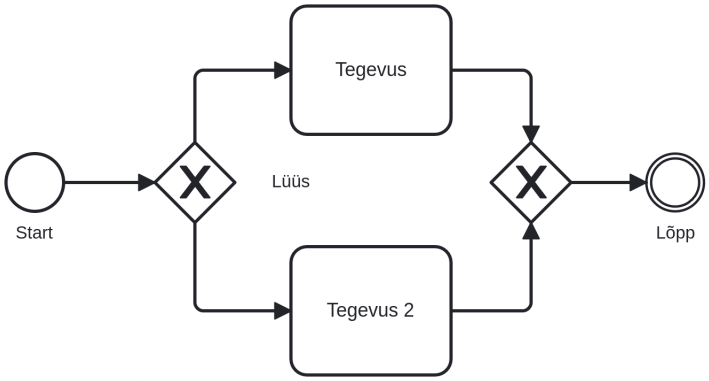
Business Process Model and Notation (BPMN), ISO 19510, diagramm on graafiline tähistussüsteem äriprotsessiskeemide tarbeks. Standardi lõi Object Management Group (OMG)[10], mittetulundusühing, mis tegeleb tehnoloogiastandardite loomise ja haldamisega. BPMN standardi loomisel oli eesmärgiks äriprotsessiskeemide üheselt mõistetavus iga osapoole jaoks. BPMN-diagrammid viivad kokku äriprotsessi osapooled ning kiirendavad disaini realiseerimist[3]. Edasi tutvustatakse BPMN-diagrammi elemente.

BPMN-diagrammi elemendid

BPMN-diagrammi elemendid jagunevad viide põhikategooriasse, mis aitavad selgelt ja tõhusalt esitada äriprotsesside loogikat[3].

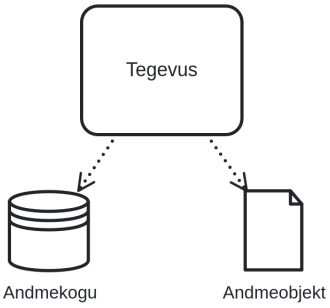
- **Vooelemendid (*Flow Objects*)**: Vooelemendid kirjeldavad protsessi põhikäitumist. Need on BPMN-diagrammi peamised elemendid (vt ka joonis 2):
  - **Sündmused (*Events*)** tähistavad protsessis olulisi hetki, nagu algus (protsess käivitub), vahe (protsessi vältel juhtub midagi) ja lõpp (protsess lõpeb).
  - **Tegevused (*Activities*)** esindavad konkreetseid ülesandeid või toiminguid, mida protsessis täidetakse, näiteks tööülesanne või automatiseeritud funktsioon.
  - **Lüüsid (*Gateways*)** võimaldavad teha otsuseid ja suunata protsessi erinevatesse harudesse vastavalt tingimustele.

[7] `https://github.com/rust-or/good_lp`
[8] `https://docs.rs/microlp/latest/microlp/`
[9] Portatiivne kompileerimise sihtkeel, mis võimaldab programme veebirakendustes kasutada.
[10] `https://www.omg.org/spec/BPMN`



Joonis 2: Voo sõlmed

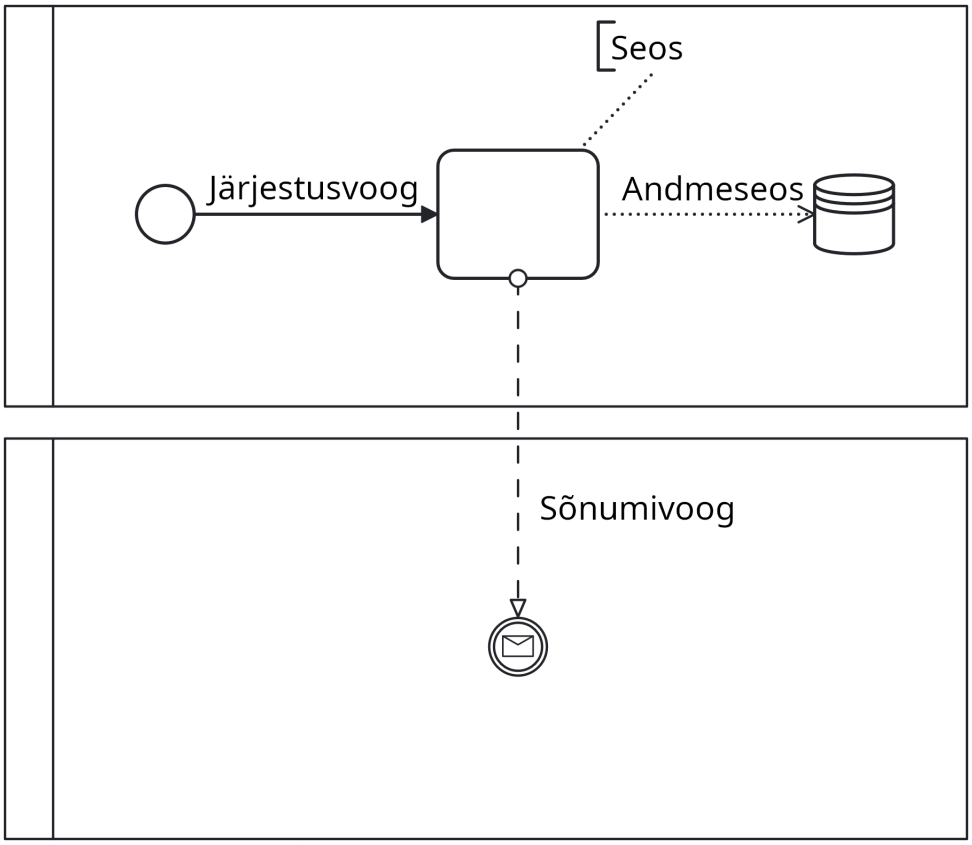
- **Andmed (*Data*)**: Andmed näitavad, millist teavet protsess kasutab ja kuidas see liigub (vt ka joonis 3):
  - **Andmeobjektid (*Data Objects*)** esindavad dokumente või muid andmeid, mida protsessis töödeldakse.
  - **Andmesisendid ja -väljundid (*Data Inputs and Outputs*)** näitavad, milliseid andmeid protsess vajab ja toodab.
  - **Andmekogud (*Data Stores*)** tähistavad pikaajalisi salvestuskohti, kus säilitatakse protsessi andmeid.



Joonis 3: Andmed

- **Ühenduselemendid (*Connecting Objects*)**: Ühenduselemendid seovad vooelemendid ja andmed, et näidata protsessi loogikat ja infovahetust (vt ka joonis 4):

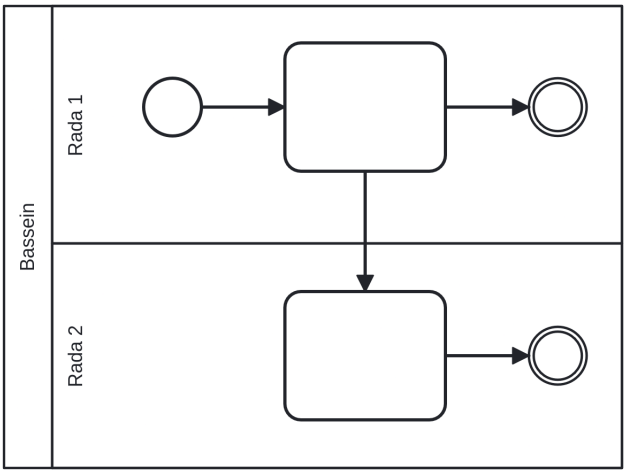
  - **Järgnevusvood (*Sequence Flows*)** ühendavad voo-objekte, näidates, millises järjekorras protsess liigub.
  - **Sõnumivoog (*Message Flows*)** näitab suhtlust protsessi osapoolte vahel.
  - **Seosed (*Associations*)** ühendavad voo-objekte täiendavate elementide, näiteks tekstimärkustega.
  - **Andmeseosed (*Data Associations*)** loovad sideme andmete ja tegevuste vahel.



Joonis 4: Ühenduselemendid

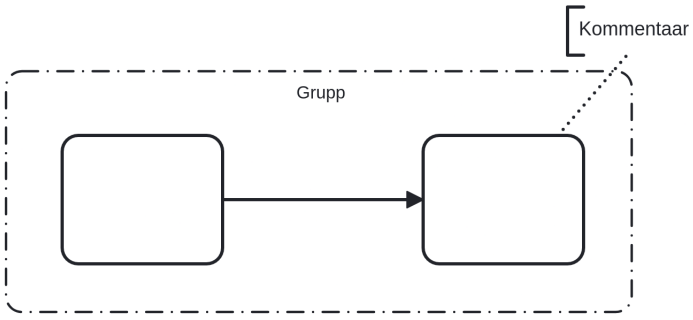
- **Basseinid ja rajad (*Swimlanes*)**: Basseinid ja rajad jagavad protsessi osadeks, et selgitada, kes või mis vastutab konkreetse osa eest (vt ka joonis 5):
  - **Basseinid (*Pools*)** tähistavad suuremaid osapooli või organisatsioone, näiteks ettevõtteid või ärivaldkondi.

– **Rajad (*Lanes*)** jagavad basseini alamvaldkondadeks, näiteks meeskondadeks või osakondadeks, selgitades vastutust ja rollijaotust.



Joonis 5: Basseinid ja rajad

- **Artefaktid (*Artifacts*)**: Artefaktid lisavad protsessi visualiseerimisele täiendavat teavet (vt ka joonis 6):
  - **Grupid (*Groups*)** võimaldavad voo-objekte loogiliselt grupeerida, ilma et see mõjutaks protsessi voolu.
  - **Kommentaarid (*Text Annotations*)** pakuvad selgitusi või täpsustusi, et parandada diagrammi arusaadavust.

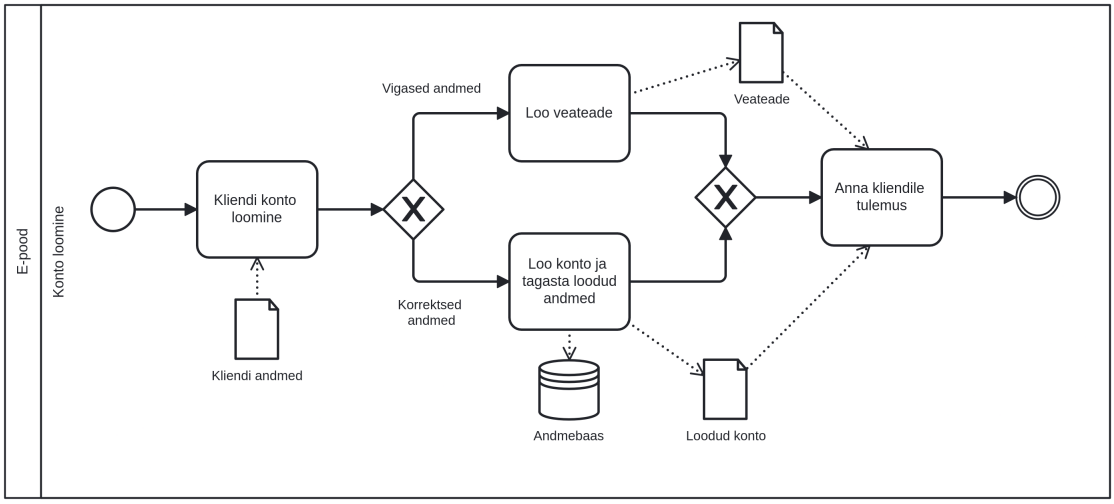


Joonis 6: Artefaktid

Selles töös on kasutatud peaaegu kõiki BPMN-diagrammi põhielementide kategooriaid. Välja on jäetud artefaktid ning sõnumid koos vastavate sõnumivoogudega. Nende kaasamine ei ole selle töö eesmärgist lähtuvalt ette nähtud, kuna see suurendaks oluliselt töömahtu. See töö keskendub eelkõige andmesõlmede funktsionaalsuse lisamisega tööriistale.

Joonis 7: Näidis BPMN-diagrammist

Joonisel 7 on kujutatud kõige lihtsamat BPMN-diagrammi, mis koosneb protsessi algussõlmest, ühest tegevusest ja protsessi lõpust. BPMN-diagrammide abil on võimalik modelleerida ka oluliselt keerukamaid protsesse, võimaldades visualiseerida keerulisi töövooge selgelt ja arusaadavalt kõigile protsessi osapooltele.



Joonis 8: Näidis BPMN-diagrammist kliendi konto loomisprotsessist e-poes

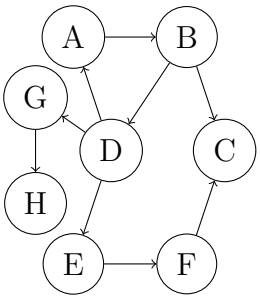
Joonis 8 illustreerib kliendi konto loomisprotsessi e-poes. Protsess algab kliendi andmete sisestamisega. Seejärel toimub lahknemine, mille alusel kontrollitakse sisestatud andmete täielikkust. Kui andmed on korrektsed, luuakse konto, salvestatakse andmed andmebaasi ning tagastatakse loodud konto andmed. Kui andmed on puudulikud, lõpetatakse protsess veateate edastamisega. Loodud diagramm näitab ka selles töös käsitlevate diagrammide keerukust. On võimalik luua keerulise-

maid diagramme koos mitme basseini, raja, sõnumivoogudega jne, aga need pole andmesõlmi käsitledes asjakohased.

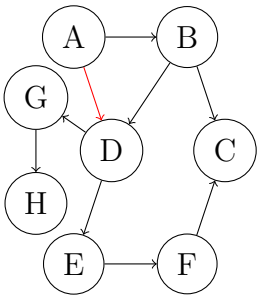
1.4 Sugiyama raamistik

Sugiyama raamistik on graafide paigutamise algoritm, mille lõi Kozo Sugiyama 1981. aastal. Raamistik loodi, et leida optimaalne paigutus graafidele, mis on suunatud ja tsüklivabad. Raamistik jagab graafi kihtideks, kus iga kiht on graafi sõlmede kogum. Iga kiht on paigutatud horisontaalselt ja sõlmed on paigutatud vertikaalselt. Sõlmede ja noolte paigutamiseks kasutatakse nii heuristilisi meetodeid kui ka ILP-d[4].

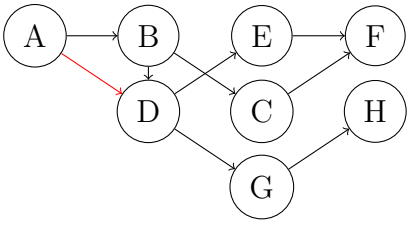
Alljärgnevalt on esitatud Sugiyama raamistiku algoritmi etapid vastavalt algoritmi definitsioonile (vt joonis 9).



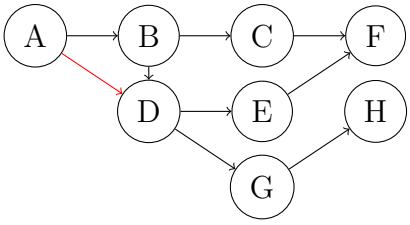
(a) Diagramm enne paigutamist, sisaldab tsükleid ja pole loetav



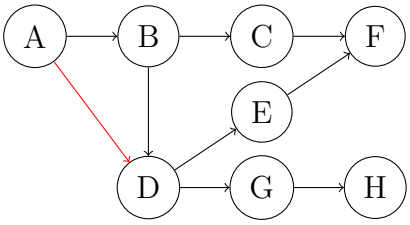
(b) Esimene samm, tsüklid eemaldatakse, et graaf oleks hierarhiline



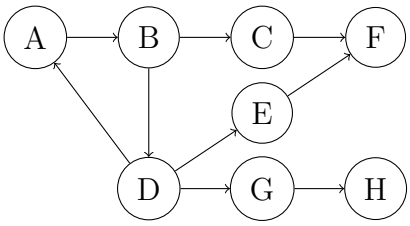
(c) Teine samm, sõlmed paigutatakse kihtidele hierarhiliselt



(d) Kolmas samm, noolte ristumised minimeeritakse



(e) Neljas samm, minimeeritakse diagonaalsed nooled



(f) Viies samm, tsüklid taastatakse

Joonis 9: Sugiyama raamistiku sammud

Tavaline Sugiyama raamistik ei ole aga otseselt kohandatud BPMN-diagrammide paigutamiseks. Selleks on vaja teha raamistikus muudatusi, mis tulevad välja töö teostuses. Alljärgnevalt on kirjeldatud kuidas traditsioonilises Sugiyama raamistikus sammud täidetakse.

Traditsiooniline Sugiyama raamistik

Samm 1: Tsüklite ja tagurpidi noolte eemaldamine Samm lahendatakse heuristiliselt, on NP-raske, ja seisneb tsüklite eemaldamises, et tekiks tsüklivaba suunatud graaf. Tsüklite eemaldamiseks leitakse minimaalne arv nooli, mis põhjustavad tsükli, ja need pööratakse järgmisteks sammudeks ümber. Ümberpööratud nooled taastatakse viimase sammuna.

Samm 2: Kihtidele paigutamine Kihte leitakse ILP-ga, minimeerides noolte pikkused, ehk on NP-täielik.

Samm 3: Nooltevaheliste ristumiste vähendamine Nooltevaheliste ristumiste vähendamine toimub raskuskeskme meetodil, mis on NP-raske, nagu on defineerinud Sugiyama. Ristumised minimeeritakse kahe järjestikuse kihi kaupa, kus leitakse sõlmede raskuskeskmed vastavalt nende ühendustele ja järjestatakse arvutatud raskuskeskmete järgi. Sugiyama pakub ka välja ILP, mis vähendab ristumisi sama efektiivselt.

Samm 4: XY-koordinaadi määramine Y-koordinaadid leitakse ILP-ga, minimeerides vertikaalset vahet kahe ühendatud sõlme vahel, et nooled oleksid ideaalis horisontaalsed.

Samm 5: Noolte marsruutimine Puudub traditsioonilises Sugiyama raamistikus, kus loodi kas otsenooled või kaared sõlmede vahel. Siinkohal toetutakse G. Sanderi[5] loodud heuristilisele Manhattani noolte marsruutimise põhimõtetele.

Kirjeldatud sammude täitmisel on võimalik luua korrektselt paigutatud ja kergesti loetav suunatud diagramm. Järgnevalt tutvustatakse lähemalt paigutamise protsessi ja tööriista prototüüpi, samuti tulevad edaspidi välja muudatused Sugiyama raamistikus.

1.5 BPMN-diagrammi automaatne paigutamine

BPMN-diagrammide loomisel on visuaalne selgus ja loogiline struktuur kriitilise tähtsusega, sest need aitavad paremini mõista protsessi kulgu ning tuvastada võimalikke kitsaskohti. Käsitsi paigutamine võib seejuures olla aeganõudev ja vigadele aldis, eriti kui protsesse tuleb sageli uuendada. Seetõttu on BPMN-i elementide automaatne paigutamine oluline tööriist, mis aitab luua ühtset ja loetavat protsessipilti. Suunatud graafide automaatne paigutus tugineb sageli Sugiyama raamistikule, mis võimaldab elemendid loogiliselt erinevatele kihtidele jaotada. Paraku tekib probleem, kui protsessi kaasatakse ka andmeobjekte ja andmekogusid. Need ei sobitu Sugiyama loodud struktuuri, sest andmesõlmed ei kuulu protsessivoogu. Domrös ja von Hanxleden [6] hoiatavad, et Sugiyama algoritmi pimesi rakendamine võib viia joonisteni, mis ei peegelda mudeli tegelikke vajadusi, mistõttu on oluline rakendada paigutuse üle kontrolli ja lisada protsessile sobivaid kohandusi. Samuti tõdevad Effinger jt [7], et BPMN-diagrammi efektiivne automaatne paigutus eeldab formaliseeritud esteetika ja loetavuskriteeriume, mis aitavad tagada visuaalse järjepidevuse ning vältida kasutajates segadust tekitavaid jooniseid. Nii on oluline integreerida esteetilised ja loogilised kaalutlused juba modelleerimisprotsessi algfaasis, et automaatne paigutus arvestaks nii protsessi kui ka andmeelementide eripärasid ning toetaks mudeli kasutajate tegelikke eesmärke.

1.6 Programmi algseis

Automaatse eksperimentaalse BPMN paigutustööriista arendus sai alguse 2024. aasta septembris kursuse Tarkvaraprojekt (LTAT.05.005) raames. Kursusel töötati neljases grupis välja prototüüp, mis loeb BPMN-diagrammi ning paigutab selle modifitseeritud Sugiyama raamistiku abil. Selle töö autor oli grupi liige ning vastutas diagrammide paigutamise eest. Sisendi koostamiseks loodi domeenispetsiifiline keel ning paigutamiseks kasutati Sugiyama raamistikku, mille rakendamisel on kasutatud nii täisarv-lineaaroptimeerimist kui ka heuristilisi meetodeid.

Programm suutis töödelda ainult vooelemente. Esmalt loeti sisse sisend, seejärel tuvastati ILP abil andmesõlmede kihid. Pärast seda määrati heuristikate abil sõlmede ligikaudsed asukohad ning noolte trajektoorid nende vahel. Lõpuks genereeriti BPMN standardile vastav .bpmn fail, mis vastas algsele sisendile. Väljastatud faili saab avada välises BPMN töötlusprogrammis, et kuvada visuaalne diagramm.

Alljärgnevalt on näha prototüübi põhifaili sisu:

```
...
solve_layer_assignment(&mut graph);
assign_xy_to_nodes(&mut graph);
assign_bend_points(&mut graph);
return generate_bpmn(&graph);
```

Järgnevalt kirjeldatakse kursuse jooksul loodud prototüübi peamisi komponente ning nende teostusviise.

Domeenispetsiifiline keel

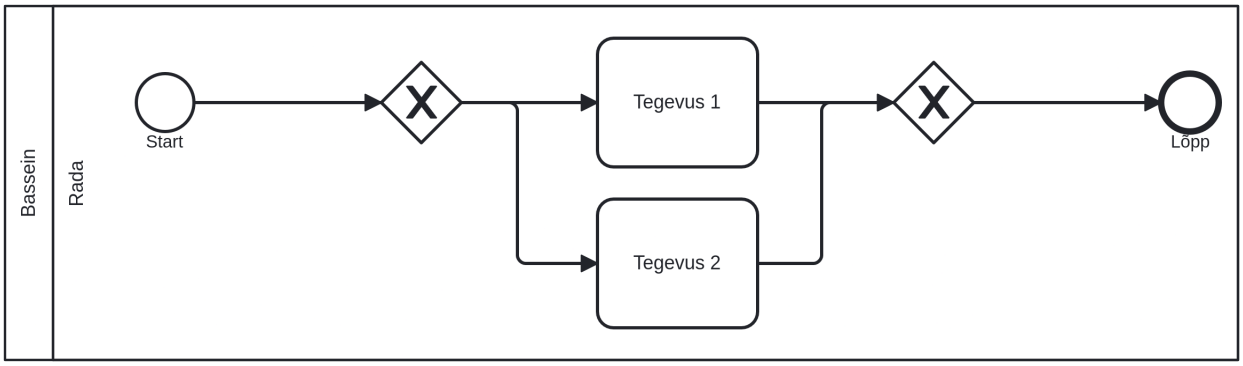
Domeenispetsiifiline keele (DSL) väljatöötamisel võeti eeskujuks PlantUML-i süntaks, mida on lihtne lugeda ja mis on hõlpsasti kohandatav BPMN-diagrammide kirjeldamiseks. Praeguseks toetab DSL enamikku olulisi elemente ja mitmesuguseid paigutusvõimalusi. Keele kasutamise kohta pakub parimat ülevaadet tööriista hoidlas asuv DSL-i dokumentatsioon[11]. Alljärgnevalt on toodud näide prototüübi DSL-i süntaksist ja sellest genereeritud BPMN-diagrammist:

```
// Kommentaare saab nii lisada
= Bassein
== Rada
# Start
X ->Lahknemine1 "Tingimus␣1" ->Lahknemine2 "Tingimus␣2
   ↪"

G <- Lahknemine1
- Tegevus 1
G ->Ühineminspunkt "Ühendan␣lahknemised"

G <- Lahknemine2
- Tegevus 2
G ->Ühineminspunkt "Ühendan␣lahknemised"

X<-Ühineminspunkt
. Lõpp
```

Rea alguses defineeritakse sõlme tüüp. Algussündmused defineeritakse sümboliga #. Pärast sõlme tüübi defineerimist saab defineerida sõlme kirjeldava teksti, mida ka sõlmes näidatakse. Mõnele sõlmetüübile nagu lahknemised saab lisada ka

[11] https://github.com/jensjager/bpmn-parser

spetsiaalsed märksõnad <- ja ->, mis vastavalt defineerivad, kas nool liigub sisse või välja.



Joonis 10: DSL-i süntaksi automaatselt koostatud diagrammi näide

Joonis 10 on loodud väljatöötatud tööriista abil. Tööriista väljundiks saadud XML-fail imporditi Camunda Modeler[12] keskkonda, mis genereeris vastava graafilise BPMN-diagrammi.

1.6.1 Paigutamine

Paigutamine põhineb modifitseeritud Sugiyama raamistikul. Kuigi raamistikku ei olnud veel tervikuna rakendatud, oli prototüübis kasutusel kolm peamist etappi: sõlmede kihtide määramine, sõlmede paigutamine ja noolte paigutamine. Sõlmede kihtide leidmiseks rakendatati täisarv-lineaaroptimeerimist ning sõlmede ja noolte paigutamiseks heuristilisi meetodeid. Järgnevalt on väljatoodud iga paigutamise samm ja selle seis prototüübis.

Samm 1: Tsüklite ja tagurpidi noolte eemaldamine Selles töös ei ole rakendatud tsüklite ega tagurpidi noolte tuvastamist ja eemaldamist. Seetõttu ei ole võimalik genereerida BPMN-diagramme, mis sisaldavad tsükleid. Paljusid protsesse on võimalik kirjeldada ka ilma tsükliteta. Seetõttu ei mõjuta tsüklite puudumine oluliselt töö tulemusi ja seega seda sammu siin ei käsitleta.

Samm 2: Kihtidele paigutamine Sõlmede kihtide määramine toimis ILP abil, mida saab näha lisas A. Mudel, mis põhineb Emden R. Gansneri jt[8] loodud mudelil, minimeeris kahe erineva noole algus- ja lõpp-punkti vahemaad.

[12] `https://modeler.camunda.io` (kasutaja loomine on kohustulik)

Samm 3: Nooltevaheliste ristumiste vähendamine Funktsioon, mis oli loodud nooltevaheliste ristumiste vähendamiseks, ei toiminud ootuspäraselt. Ajapuuduse tõttu otsustati seda prototüübis mitte kasutusele võtta.

Samm 4: XY-koordinaadi määramine Sõlmede paigutamisel, mida saab näha lisas B, kasutas programm heuristilist lähenemist, käies kihthaaval läbi kõik sõlmed. Iga sõlme puhul arvutati x-koordinaat vastavalt selle kuuluvusele konkreetsesse kihti. Y-koordinaadi arvutamisel lähtuti jooksvalt iga basseini raja esimese sõlme asukohast. Selle põhjal määrati kogu raja vertikaalne positsioon ning salvestati suurim kasutatud y-koordinaat, et arvutada järgmise raja või basseini asukoht. Sama y-koordinaati kasutati ka basseinide ja radade asukohtade ning kõrguste määramiseks.

Samm 5: Noolte marsruutimine Noolte paigutamiseks, mida saab näha lisas C, kasutati laiuti otsingu algoritmi, mis põhineb Manhattani kaugusel[13]. Otsing viidi läbi ainult horisontaalses ja vertikaalses suunas, et leida lühim võimalik tee kahe punkti vahel. Igal sammul kontrolliti, et nool ei läbiks diagrammil paiknevaid sõlmi ega ületaks diagrammi piire. Sobiva tee leidmisel minimeeriti nii teekonna pikkust kui ka suunamuutuste arvu. Sõlmedega ristumise vältimiseks kontrollitakse igal sammul, kas nool asub mõne sõlme sees.

Prototüüp sobis lihtsate diagrammide koostamiseks, kuid selle paigutus jäi siiski algeliseks. Järgnevalt on kirjeldatud, mida ja kuidas tehti tööriista laiendamisel.

2 BPMN andmesõlmede automaatne paigutamine

Selle töö praktilise osa eesmärk on täiendada eelnevalt loodud tööriista selliselt, et selle kasutusala ja rakendamisvõimalused oleksid võimalikult avarad. Algversioonis puudub tööriistal andmesõlmede tugi, mis piirab oluliselt selle kasutusväärtust. Seetõttu on vajalik lisada andmesõlmede, sealhulgas andmeobjektide ja andmekogude paigutamise toetus. Eesmärgi saavutamiseks tuleb analüüsida, kuidas paigutatakse andmeobjekte ja andmekogusid käsitsi loodud BPMN-diagrammides[14]. Analüüsi tulemusel on võimalik välja töötada reeglid, mis hõlbustavad andmeobjektide ja andmekogude automaatset paigutamist.

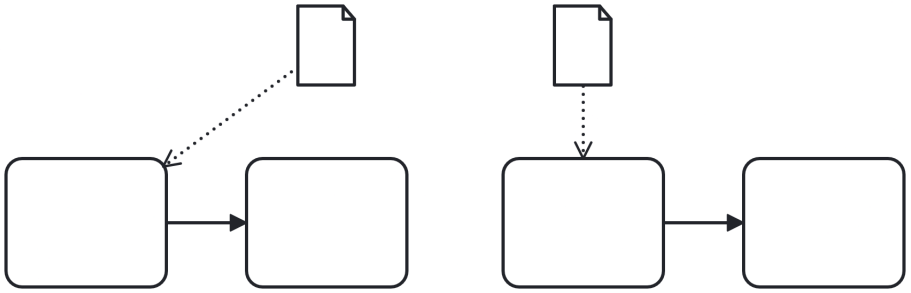
[13]Manhattani kaugus on kahe punkti vahemaa, kui liikuda saab ainult horisontaalselt ja vertikaalselt.

[14]Juhendaja Armin Daniel Kisandi sõnul on plaanis tulevikus laiendada tööriista PE-BPMN-i toetusega. PE-BPMN on BPMN-i laiendus, mis võimaldab analüüsida andmeobjektide nähtavust äriprotsessis. Seetõttu on oluline lisada tööriistale ka andmeobjektide paigaldamise tugi.

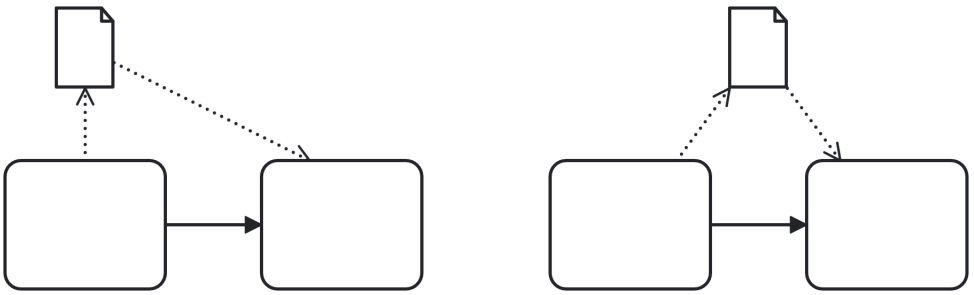
2.1 Andmeobjektide ja andmekogude paigutamise põhimõtted

Kui suunatud graafide, sealhulgas BPMN-diagrammi vooelementide (nt tegevuste ja lüüside) asukohti saab määrata Sugiyama raamistiku abil, siis andmeobjektide ja andmekogude paigutamine on keerukam. Visuaalse selguse tagamiseks peab sellele vaatamata järgima kindlaid põhimõtteid, mis aitavad paigutada andmeelemente võimalikult loogiliselt ja arusaadavalt. Käsitsi paigutatud diagrammide analüüsist leitud põhimõtted on järgmised[15] (valiku analüüsitud diagrammidest leiab lisas G):

1. Ühe kasutajaga[16] andmekogu või andmeobjekt asub samal kihil selle kasutajaga.



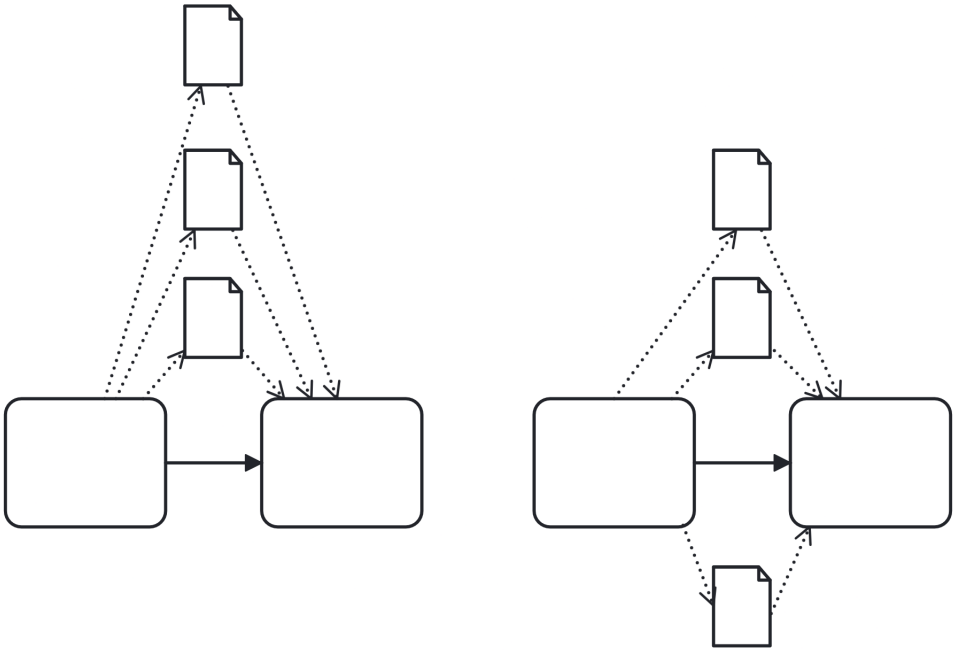
2. Andmesõlmed saavad paikneda ka "poolkihil". Poolkiht on kahe kihi vahel olev ala. Ühele poolkihile on võimalik asetada mitu andmekogu või -objekti.



[15] Iga reegli all on ka joonis, kus vasakul on "halb" paigutus ja paremal "optimaalne" paigutus. Joonised on käsitsi loodud Camunda Modeleri keskkonnas.

[16] Andmesõlme kasutaja on andmesõlmega ühenduses olev vooelement

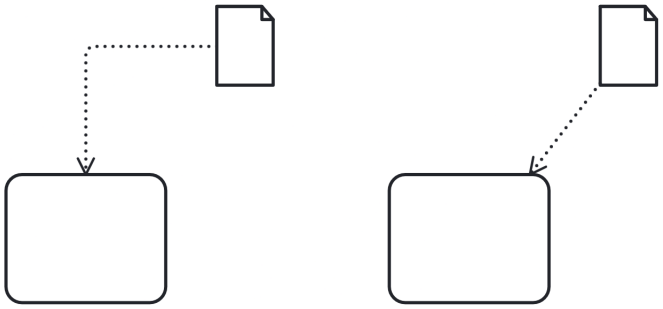
3. Võib juhtuda, et andmekogu või andmeobjekt ei mahu[17] samale kihile kasutajaga või tema määratud kihile. Sel juhul tuleb andmesõlmi hajutada.



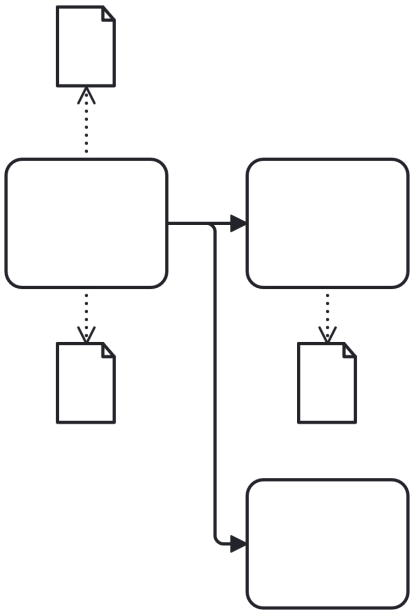
4. Andmekogu või andmeobjekti kirjeldus asetatakse andmesõlme alla.

5. Andmekogude ja andmeobjektide nooled on sirged, kui leidub otsetee kasutaja ja andmesõlme vahel.

[17] Ühel kihil on nii palju andmesõlmi, et see hakkab mõjutama diagrammi loetavust.



6. Andmekogud ja andmeobjektid saab paigutada samas kihis nii vooelemendi/vooelementide peale kui ka alla.



Eeltoodud põhimõtete järgimine aitab hoida BPMN-diagramme selge ja loogilisena ka keerukate protsesside modelleerimisel.

2.2 Teostus

Programmi arendamise käigus tekkis mitu uut funktsiooni ja muudeti olemasolevaid funktsioone(vt tabel 1). Selle käigus muutus ka olemasolevate funktsioonide täitmisjärjekord. Esimeseks jäi siiski kihtide määramine, millele järgneb andmeobjektide kihtide määramine. Seejärel leitakse ajutised sõlmed nii tavalistele andmeobjektide nooltele kui ka neile nooltele, mille pikkused on üle ühe kihi. Pärast seda

leitakse x- ja y-koordinaadid. Kui on võimalik luua otseühendus andmenoolega, siis leitakse need järgmisena. Eelviimasena marsruuditakse nooled. Lõpuks vahetatakse ajutised sõlmed välja ja asendadakse jälle algsete täispikkade nooltega. Alljärgnevalt on näha lõpliku põhifaili sisu:

```
...
solve_layer_assignment(&mut graph);
solve_data_layer_assignment(&mut graph);
generate_dummy_nodes(&mut graph);
reduce_all_crossings(&mut graph);
assign_xy_ilp(&mut graph);
find_straight_data_edges(&mut graph);
edge_routing(&mut graph);
replace_dummy_nodes(&mut graph);
```

Tabel 1: Tööriista alg- ja lõppseisu võrdlus.

Samm	**Enne**	**Pärast**
Tsüklite ja tagurpidi noolte eemaldamine	Ei olnud	Ei ole
Kihtidele paigutamine	Ainult vooelemendid	Vooelemendid ja andmesõlmed
Nooltevaheliste ristumiste vähendamine	Ainult vooelemendid	Vooelemendid ja andmesõlmed
XY-koordinaadi määramine	Naiivne vooelementide üksteise alla paigutus	ILP, mis arvestab vooelementide ja andmesõlmedega
Noolte marsruutimine	Manhattani algoritm	Otsenooled andmesõlmedele, poolik kuna tsükleid ja tagurpidi nooli pole eemaldatud

Alljärgnevalt on esitatud iga programmi sammu juures tehtud muudatused ja täiendused.

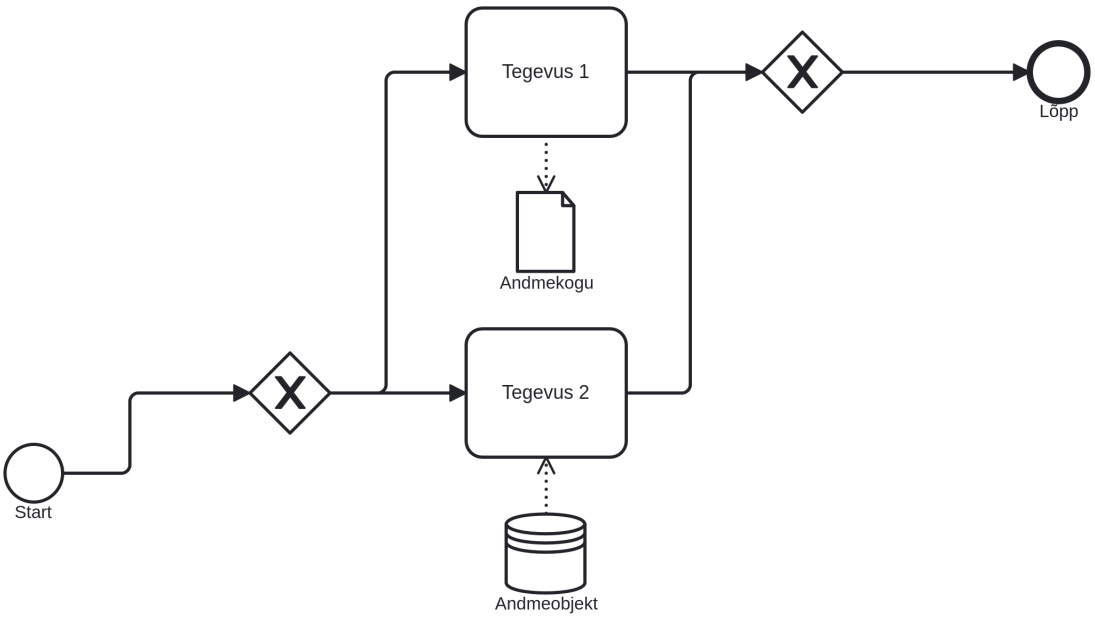
2.2.1 Domeenispetsiifilise keele laiendused

Andmesõlmede rakendamiseks on vaja olemasolevat DSL-i laiendada. Selleks on sisse toodud uued sümbolid OD ja SD (vastavalt andmeobjekt ja andmekogu)[18]. Vaadates allolevat näidet, on lihtne näha, kuidas andmesõlmi defineerida. Esmalt

[18]Sümbolid OD ja SD valiti juhendaja Armin Daniel Kisandi soovitusel, lähtudes tema plaanist täiendada DSL-i tulevikus.

on vaja anda tegevusele @ sümboliga tegevust identifiseeriv nimi. Andmesõlmed tuleb defineerida pärast ülejäänud diagrammi defineerimist. Noole suunda andmesõlme ja tegevuse vahel peab kirja panema kas <- või -> nooltega ning pärast seda tuleb sisestada tegevuse nimi. Alljärgnevalt on toodud näide laiendatud DSL-i süntaksist ja sellest genereeritud BPMN-diagrammist:

```
# Start
X ->Lahkenmine1 "Tingimus␣1" ->Lahkenmine2 "Tingimus␣2
   ↪"

G <- Lahkenmine1
- Tegevus 1 @tegevus1
G ->Ühineminspunkt "Ühendan␣lahknemised"

G <- Lahkenmine2
- Tegevus 2 @tegevus2
G ->Ühineminspunkt "Ühendan␣lahknemised"

X<-Ühineminspunkt
. Lõpp

OD <-tegevus1
SD ->tegevus2
```



Joonis 11: Laiendatud DSL-i süntaksi automaatselt koostatud diagrammi näide.

Joonisel 11 on näha, kuidas näeb välja ülal defineeritud diagramm. Vastavalt andmesõlme paigutuse reeglile 1 on mõlemad andmesõlmed paigutatud samale kihile oma kasutajaga.

2.2.2 Andmesõlmede kihtidele paigutamine

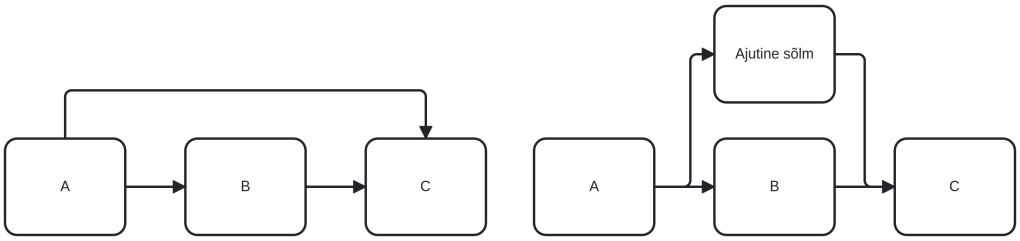
Esimesena prooviti andmeobjekte paigutada kihtidele ILP-ga. ILP-mudel realiseeriti ning testimisel oli see algselt funktsionaalne(vt lisa D). Testimise käigus ilmnesid aga teatud erandid. Näiteks ilmnesid olukorrad, mil ühele kihile paigutati liiga palju andmeobjekte. Liigne kuhjumine ühel kihil muudab diagrammi vertikaalselt liiga kõrgeks ning halvendab selle loetavust. Selliste juhtude vältimiseks oleks olnud vajalik täiendavate kitsenduste lisamine. Need täiendavad kitsendused oleksid aga oluliselt suurendanud muutujate hulka, mille tulemusel oleks ILP-mudeli jõudlus oluliselt vähenenud. Seega ei pakkunud kirjeldatud ILP-lahendus probleemile piisavalt efektiivset lahendust.

Kuna enamik andmeobjekte paigutub kihile, mis vastab kasutajate kauguste keskmisele väärtusele (mida ka eelnevalt kirjeldatud ILP ilma lisakitsendusteta tegi), on otstarbekam määrata iga andmeobjekti kiht heuristilise meetodiga. Selleks arvutatakse kasutajate kauguste põhjal objekti keskmise kiht. Pärast kihtide esialgset määramist on vajadusel võimalik andmeobjekte täiendava algoritmiga ümber paigutada, nagu näiteks eelnevalt väljatoodud kuhjumise puhul.

Praeguse meetodi puudujääk seisneb selles, et andmesõlmede paigutamisel ei arvestata reeglit 3, mille kohaselt tuleks vältida liiga tihedat sõlmede koondumist ühele kihile. Selle tulemusel võib tekkida olukord, kus sõlmede kuhjumine halvendab diagrammi loetavust ning neid ei hajutata automaatselt. Tegemist on siiski väikese puudujäägiga, kuna tavakasutuses sellised olukorrad tõenäoliselt ei esine. Piirjuhtudel on kasutajal võimalik pärast automaatset paigutust andmesõlmed käsitsi sobivamalt ümber paigutada.

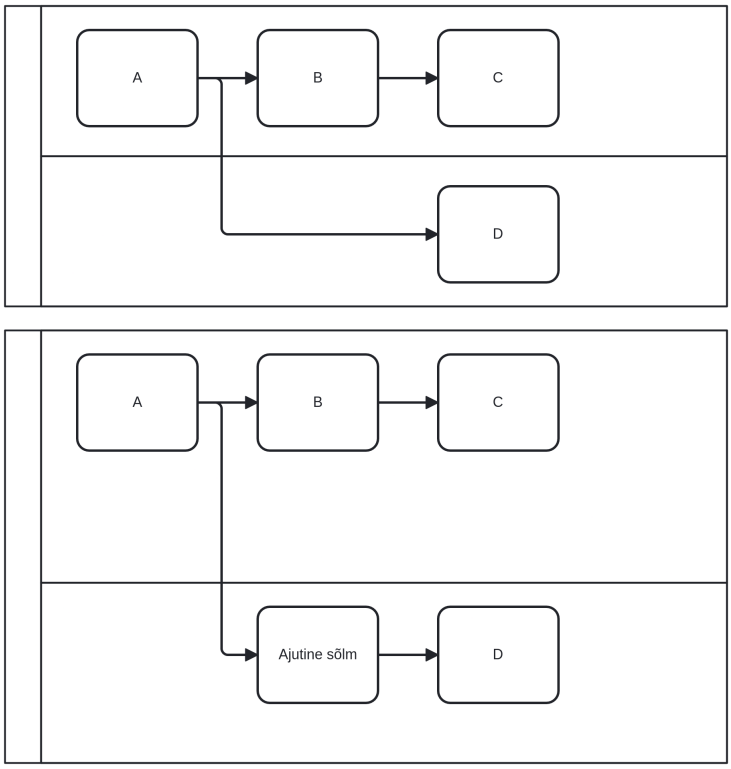
2.2.3 Ajutiste sõlmede leidmine

Ajutiste sõlmede loomine sujundab järgmiseid algoritme, kuna see võimaldab eeldada, et iga noole pikkus vastab ainult ühele kihile. Ajutised sõlmed tuleb lisada igale noolele, mille lähte- ja sihtkihi vaheline kaugus ületab ühe kihi (vt joonis 12). Selleks paigutatakse täiendav sõlm igale vahekihile, mis jääb noole lähte- ja sihtkihi vahele. Algne pikk nool asendatakse järjestikuste ühendustega, mis läbivad loodud ajutisi sõlmi. Selline lähenemine võimaldab saavutada selgema ja optimaalsema visualiseeringu järgmistel sammudel.



Joonis 12: Ajutiste sõlmede leidmise näide. Vasakul on kujutatud olukord, kus eksisteerib serv A–C, mille pikkus ületab ühe kihi. Paremal on lisatud ajutine sõlm, mille tulemusel on kõik servad diagrammis lühendatud pikkuseni 1.

Poolkihil paiknevaid andmesõlmi käsitletakse paigutusalgoritmis samaväärselt täiskihil asuvatega. Seetõttu ei ole andmenoolte töötlemisel vaja teha täiendavaid kohandusi.

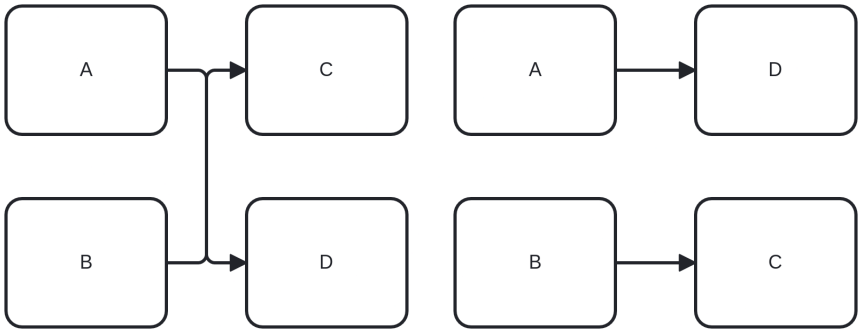


Joonis 13: Ajutiste sõlmede leidmise näide rajavahetuse korral.

Kui andmenool algab ühes rajast, kuid lõpeb teises, tuleb ajutiste sõlmede leidmisel arvestada rajavahetusega (vt joonis 13). Rajavahetus toimub täpselt noole keskpunktis. See tähendab, et nool liigub esmalt horisontaalselt kuni keskpunktini. Seejärel suundub nool vertikaalselt üles või alla sihtrajale. Pärast sihtrajale jõudmist jätkub liikumine taas horisontaalselt sihtsõlmeni. Rajavahetuse teostamiseks lisatakse sihtraja keskpunktis ajutine sõlm. Kui nool on jõudnud sihtrajale, jätkub ajutiste sõlmede lisamine samal viisil nagu ühe raja piires liikuvate noolte puhul.

2.2.4 Nooltevaheliste ristumiste vähendamine

Nooltevahelisi ristumisi vähendatakse raskuskeskme meetodil. Meetodi eesmärk on järjestada igas kihis olevad sõlmed nii, et need paikneksid võimalikult lähedal oma ühendatud naabrite keskmisele positsioonile (vt joonis 14). Selle saavutamiseks arvutatakse igas kihis iga sõlme jaoks raskuskese, tuginedes ainult väljaminevatele ühendustele. Seejärel järjestatakse sõlmed vastavalt arvutatud väärtustele ümber. Protsessi korratakse, kuni sõlmede järjestus stabiliseerub või hakkab korduma.



Joonis 14: Nooltevaheliste ristumiste vähendamise näide.

Andmesõlmede puhul tekib piirjuht, kui sõlmel on vaid üks kasutaja. Sellisel juhul paigutatakse andmesõlm otse oma ainsa kasutaja alla samas kihis. Ülejäänud sama kihi sõlmed nihutatakse seejärel ühe positsiooni võrra edasi.

Andmesõlmi, mis on poolkihil, käsitletakse nooltevahelisi ristumisi vähendades nii, nagu need asuksid täiskihil.

2.2.5 Y-koordinaatide määramine

Y-koordinaatide määramise meetodit on oluliselt edasi arendatud. Esialgne heuristiline lähenemine ei andnud suuremate diagrammide puhul visuaalselt kvaliteetseid tulemusi ning seetõttu asendati see ILP-ga. Uus meetod minimeerib noole algus- ja lõpp-punktide vaheliste kõrgusvahede summa. Meetodile lisati täiendavad kitsendused, mis säilitavad kihis paiknevate sõlmede vahel minimaalse vahemaa.

Ajutiste sõlmedega arvestamiseks tuli lisada eraldi konstandid. Need määravad minimaalse lubatud kauguse samas kihis paiknevate ajutiste sõlmede vahel ning üldise minimaalse vahemaa kihis olevate sõlmede vahel.

Järgmiselt esitatakse Y-koordinaatide määramiseks koostatud ILP. ILP põhineb Emden R. Gansneri jt[8] loodud mudelil.

Eesmärgifunktsioon:

$$\min \sum_{e \in E} \omega_e |y(w) - y(v)| \tag{1}$$

Muutujad:

- E: Järjestusvoogude ja andmeseoste hulk graafis.
- $e \in E$: Nool e, mis ühendab kahte tippu v_e ja w_e.

- $\omega_e \in R_{\geq 0}$: Noolespetsiifiline kaal:
  - $NOOL = 10.0$
  - $ANDMENOOL = 0.01$
  - $AJUTINE\_NOOL = 2.0$

  Erinevate väärtuste proovimisel osutusid need optimaalseteks kaaludeks. Optimaalse paigutuse jaoks peab prioriseerima seda, et järgnevusvood oleksid sirged, mistõttu on neil ka kõige suurem kaal (vt lisa E).
- $y(v) \in R_{\geq 0}$: Tipu v vertikaalne positsioon.
- $|y(w) - y(v)|$: Noole tippude w ja v vahe absoluutväärtus.

Kitsendused:

1. Tipu positsioon on positiivne reaalarv:

$$y(v) \in R_{\geq \sigma} \quad \forall v \in V$$

   Kus $\sigma = 100$ on minimaalne lubatud väärtus[19].

2. Selleks et absoluutväärtust $|y(w_e) - y(v_e)|$ defineerida, luuakse iga noole e jaoks abimuutuja d_e, mis modelleerib seda väärtust, ja rakendatakse järgmised kitsendused. See tehnika põhineb *Advanced Interactive Multidimensional Modeling System* loodud ILP modelleerimise raamatus esitatud võttel[9].

$$\begin{aligned} d_e &\geq y(w_e) - y(v_e) \\ d_e &\geq y(v_e) - y(w_e) \\ d_e &\in R_{\geq 0} \end{aligned}$$

   Seega $d_e = |y(w_e) - y(v_e)|$

3. Samas kihis olevate tippude omavaheline kaugus:

$$y(v_i) < y(v_j) \quad \text{kui } \pi(v_i) < \pi(v_j)$$

   Kus $\pi(v)$ tähistab tipu v positsiooni kihis.

[19] Väärtus on arbitraarne, kuid peab olema suurem kui suurima sõlme kõrgus. Vastasel juhul võivad järgnevad meetodid anda negatiivseid väärtusi. Antud väärtus ei mõjuta paigutust.

4. Kahe järjestikuse sõlme vahe sõltuvalt nende tüübist:

   Olgu igas kihis sõlmed järjestatud positsiooni alusel (st $v_1, v_2, \ldots, v_n$) ning $y(v)$ sõlme v vertikaalne koordinaat.

   Siis kehtivad iga kahe järjestikuse sõlme v_i ja v_{i-1} kohta järgmised kitsendused:

$$y(v_{i-1}) - y(v_i) \geq \begin{cases} \delta_{aa}, & \text{kui } v_i \text{ ja } v_{i-1} \text{ on mõlemad ajutised} \\ \delta_{ta}, & \text{kui üks neist on ajutine} \\ \delta_{tt}, & \text{kui } v_i \text{ ja } v_{i-1} \text{ on mõlemad tavalised} \end{cases}$$

   Kus:

   - δ_{aa} on minimaalne vahe kahe ajutise sõlme vahel,
   - δ_{ta} on minimaalne vahe ajutise ja tavalise sõlme vahel,
   - δ_{tt} on minimaalne vahe kahe tavalise sõlme vahel.

Meetodi rakendamisel ilmnes siiski probleem juhtudel, mil samas kihis paiknes lüüs koos teiste sõlmedega. Sellistes olukordades tekkisid soovimatud noolte ristumised lüüsi sisenevate nooltega. Selle lahendamiseks lisati meetodisse täiendav kitsendus. Kitsendus määrab lüüsiga samas kihis paiknevate sõlmede y-koordinaadid nii, et need asuvad kõrvalkihis paiknevatest sõlmedest kas rangelt kõrgemal või rangelt madalamal.

Lüüsi kihil olevate sõlmede paigutamise kitsendus: Kui lüüsiga asub samas kihis veel sõlmi, siis tagatakse, et:

- samas kihis olevad sõlmed, mis on enne lüüsi, jäävad kõrgemale eelmise kihi esimesest sõlmest:

$$y(h_{\text{esimene}}) - y(v) \geq \delta_b$$

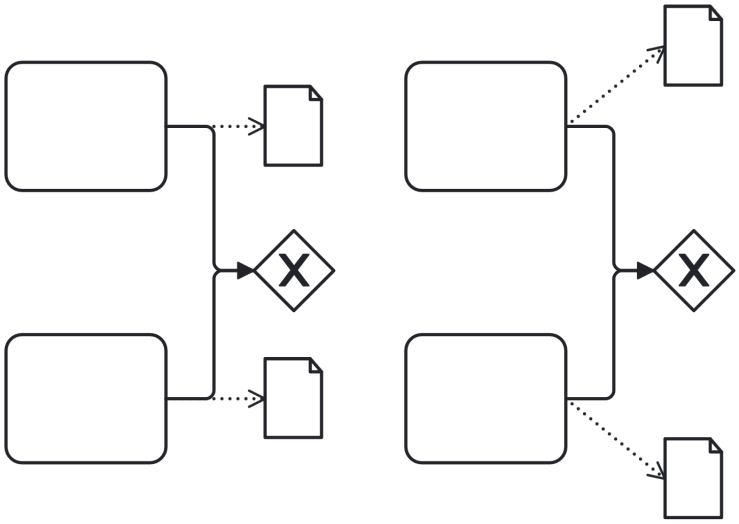
- samas kihis olevad sõlmed, mis on pärast lüüsi, jäävad madalamale eelmise kihi viimasest sõlmest:

$$y(v) - y(l_{\text{viimane}}) \geq \delta_b$$

Kus:

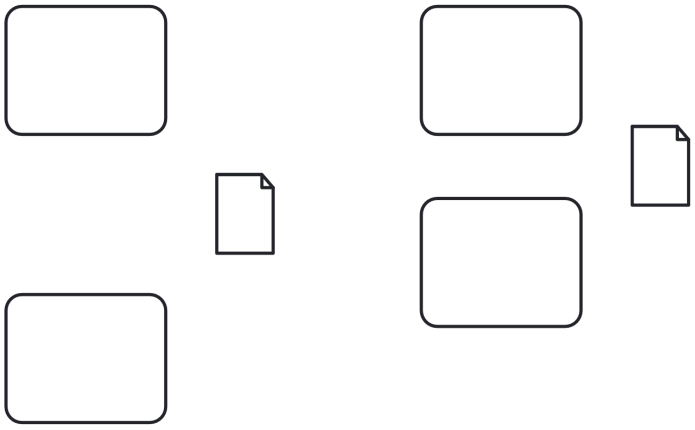
- $h_{\text{esimene}}, l_{\text{viimane}}$ — vastavalt eelmise kihi kõrgeim ja madalaim tipp.

- δ_b — minimaalne kõrgusevahe eelmise kõrgeima/madalaima sõlmega.



Joonis 15: Lüüsi kihil olevate sõlmede paigutamise kitsenduse näide.

Vaadates joonist 15, on vasakul näha ilma kitsenduseta juhtu. Andmesõlmed, mis on paigutatud lüüsiga samale kihile, segavad loetavust. Paremal on näha, kuidas üles ja alla liigutatud andmesõlmed tagavad selgema diagrammi.

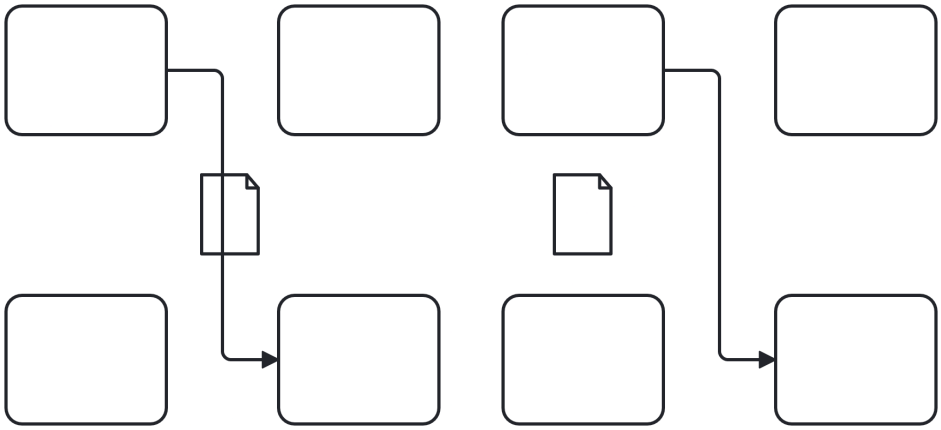


Joonis 16: Y-koordinaatide määramine poolkihtide korral. Vasakul on kujutatud tööriista praegune paigutus, mille puhul tekib sõlmede vahele vahe, mida ei ole alati vaja, paremal võimalik parendatud paigutus.

Andmesõlmi, mis asuvad poolkihil, käsitletakse y-koordinaatide määramisel nii, nagu need asuksid täiskihil. Selle tulemusel jääb praeguses lahenduses täiskihile kasutamata ruumi, kuna y-koordinaatide leidmisel ei ole veel teada, kas andmesõlme peab noolte ristumiste tõttu täiskihile liigutama. See ei sega oluliselt toimimist, ent edaspidi tuleks kaaluda lahendusi, mis võimaldaksid selle tühiku minimeerimist (vt joonis 16).

2.2.6 X-koordinaatide määramine

X-koordinaadid määratakse endiselt naiivselt. See meetod käib läbi kõik sõlmed ning määrab iga sõlme x-koordinaadi, korrutades sõlme kihi numbri kahe kihi vahelise standardse kaugusega. Sama protsessi käigus arvutatakse ka sõlmede vajalikud horisontaalsed ja vertikaalsed nihked.



Joonis 17: Y-koordinaatide määramine poolkihtide korral.

Siin funktsioonis käsitletakse ka piirjuhtu, kui andmesõlm jääb vertikaalse noolesegmendi sisse. Piirjuht esineb, kui andmesõlm on paigutatud poolkihile ja näiteks tema kohal olev tegevus on andmesõlmest all oleva tegevusega ühendatud. Meetod tuvastab selle ja liigutab andmesõlme täiskihile (vt joonis 17).

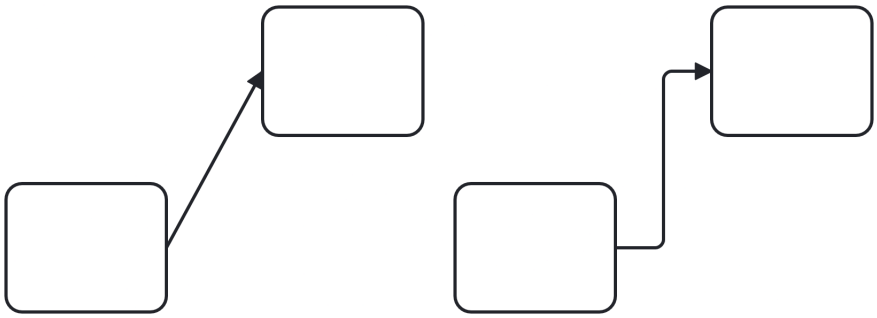
2.2.7 Andmenoolte otseühenduste loomine

Andmenoolte paigutamisel on üheks oluliseks põhimõtteks see, et nool peaks võimalusel olema sirgjooneline (vt reegel 5). Selleks kasutatakse osaliselt algset noolte marsruutimismeetodit (vt lisa C). Meetodis luuakse maatriks, mis kirjeldab iga sõlme asukohta graafis. Seejärel analüüsitakse kõiki andmenoolte ühendusi, mille jaoks on defineeritud kõik võimalikud algus- ja lõpp-punktid. Iga ühenduse puhul hinnatakse, kas on võimalik luua sirgjooneline otsetee, mis ei lõiku ühegi teise

sõlmega. Kõik sellised kehtivad otseteed koondatakse. Kui sobivaid otseteid leidub, valitakse neist kõige lühem ning selle koordinaate kasutatakse andmenoole marsruudi määramiseks.

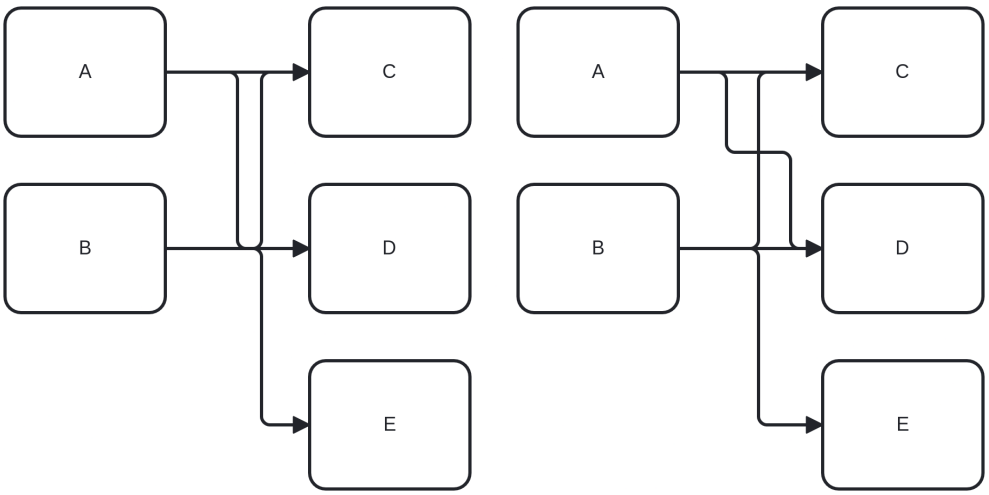
2.2.8 Noolte marsruutimine

Eelnevalt on tagatud, et iga noole pikkus ei ületa üht kihti. Seetõttu on võimalik rakendada kihipõhist noolemarsruutimist. Iga kihi nooled saab grupeerida ning igale noolele on võimalik määrata x-koordinaat, mille kaudu nool liigub kas üles- või allapoole. Siinkohal märgendatakse ka kõik ajutised andmenooled, millele on leitud otsetee. Kui alguse ja lõpu y-koordinaadid ühtivad, ei ole vaja nooli marsruutida.



Joonis 18: Noolte marsruutimise näide.

Joonisel 18 on selgelt näha, kuidas noolel on algselt vaid algus- ja lõppkoordinaadid. Marsruutimise tulemusena on määratud noolele punkt, kust see saab vertikaalselt sihtsõlme y-koordinaadile minna. Juhul kui ühel kihil on mitu noolt, siis nende vertikaalsed positsioonid on eraldatud nii, et need ei kattuks. Praeguses seisus on nooled, mis väljuvad samast sõlmest, grupeeritud ühele vertikaalsele joonele (vt joonis 19).

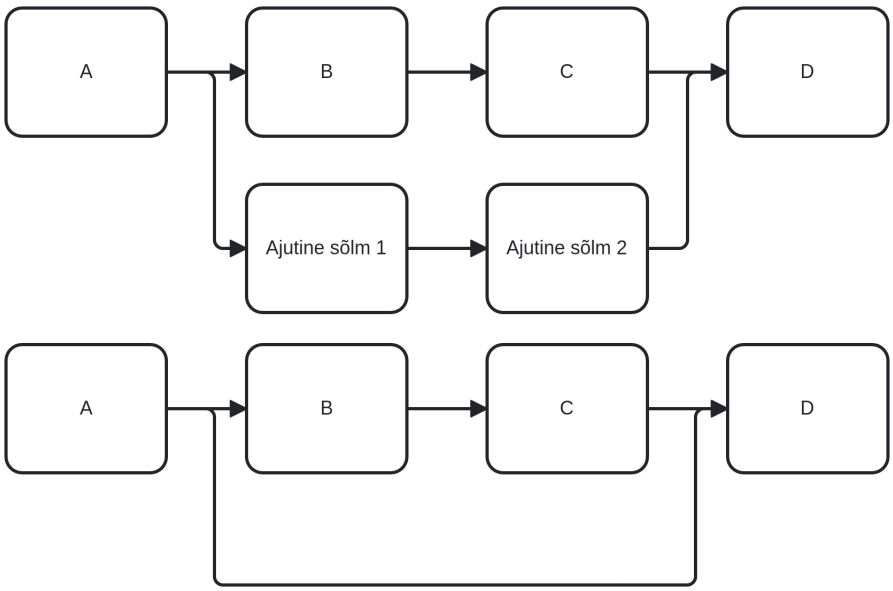


Joonis 19: Noolte marsruutimise puudujäägi näide mitme noolega. Vasakul on näha, kuidas praeguses seisus nooled marsruuditakse, ja paremal on optimaalne lahendus.

Vaadates joonist 19, tekib arusaamatus, kuidas ja kas on A-D tegevused ühendatud. Edasine töö, sealhulgas noolemarsruutimise lõplik valmimine, ei kuulu selle töö mahtu. Järgmisena on võimalik nooled sorteerida viisil, mis minimeerib nendevahelisi ristumisi ning parendab diagrammi loetavust.

2.2.9 Ajutiste sõlmede eemaldamine

Enne kui diagrammi saab väljastada XML-failis, tuleb ajutised sõlmed eemaldada, et rekonstrueerida pikad nooled (vt joonis 20). Selle protsessi käigus läbitakse iga tavalisest sõlmest (sh andmesõlmedest) väljuv ajutine nool. Igat ajutist noolt järgides liigutakse samm-sammult edasi, kuni jõutakse noole tegeliku sihtsõlmeni. Marsruudil kogutakse kõik ajutiste noolte vahepunktide koordinaadid. Lõpuks lisatakse need vastavale algsele pikale noolele.



Joonis 20: Ajutiste sõlmede eemaldamise näide.

Ajutiste sõlmede eemaldamisel jäetakse vahele kõik andmenooled, millele on andmenoolte otseühenduste loomise sammus otsetee juba leitud. Need nooled on vastavalt märgendatud noolte marsruutimisel.

3 Jõudlustestid

Jõudluse hindamiseks kompileeriti tööriist ning loodi testfailid. Testfailid sisaldasid eksponentsiaalselt suureneva arvu sõlmedega graafe, vahemikus 1 kuni 1000 sõlme (1, 5, 10, 25, 50, 100, 250, 500, 1000). Testid viidi läbi kaht tüüpi testfailidega, kus olid ainult vooelemendid ning võrdne arv vooelemente ja andmesõlmi. Jõudlustestide läbiviimiseks kasutati tööriista *hyperfine* [10], mis võimaldab täpset ja korduvat täitmisaegade mõõtmist.

Jõudlustestid teostati AMD Ryzen 5 7600X protsessoril, kasutades Linuxi tuuma versiooni 6.14.5. Lähtekood kompileeriti Rusti 1.86.0-nightly versiooniga.

All on näha testfaili sisu ühe vooelemendiga:

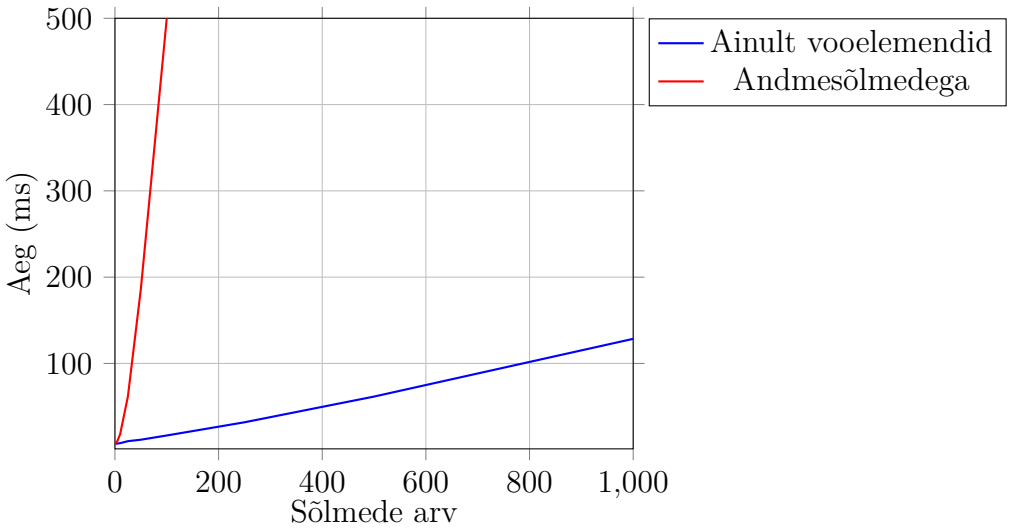
```
# Start
- Task @task1
. End
```

Teist tüüpi testfailis oli sama kogus andmesõlmi:

```
# Start
```

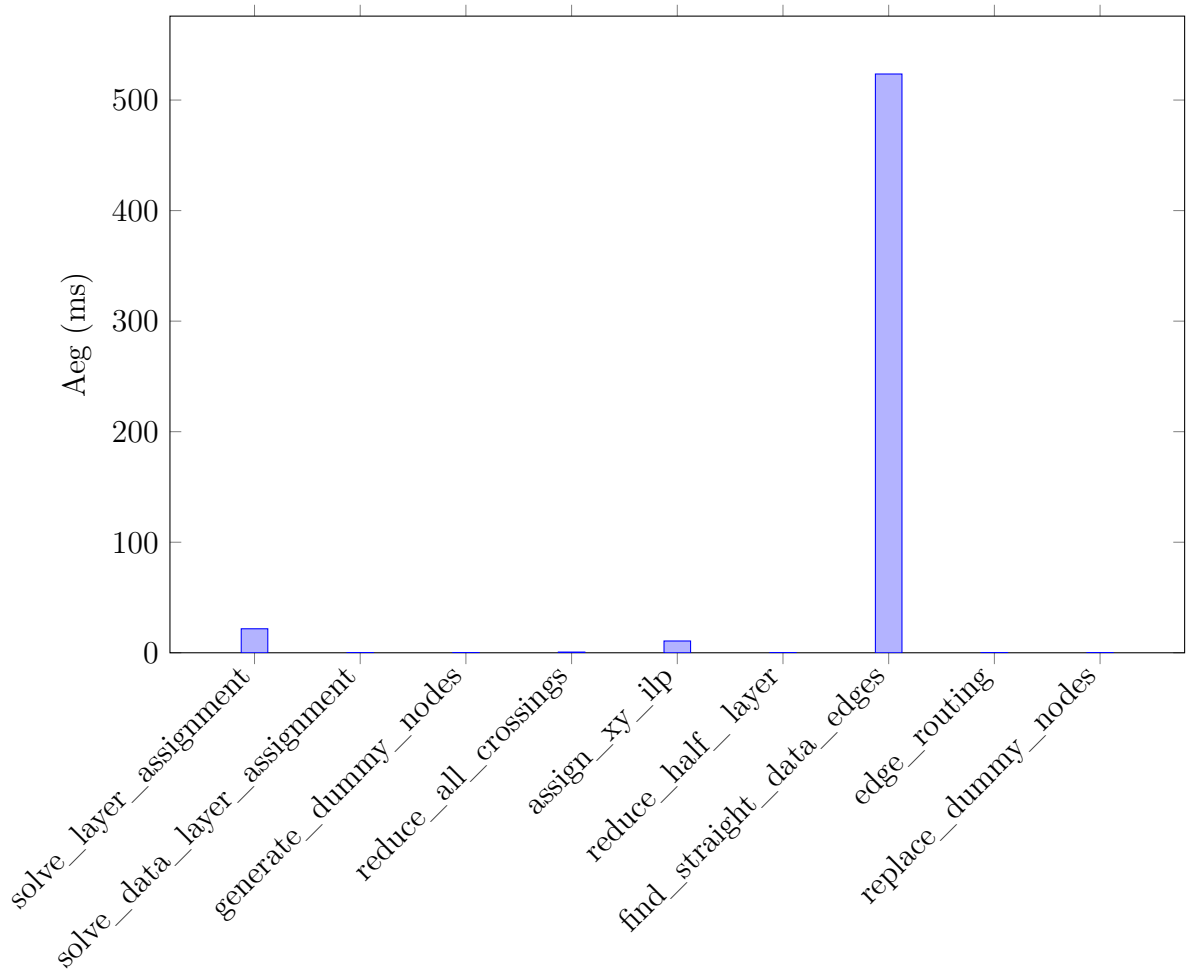
```
- Task @task1
SD <-task1
. End
```

Iga testfailiga leiti täitmisaeg 5 korda ja arvesse võeti keskmine täitmisaeg.



Joonis 21: Jõudlustesti tulemuste graaf

Jooniselt 21 on näha, et ainult vooelementide kasutamisel on tööriista ajaline keerukus lineaarne. Andmesõlmede lisamine muudab aga kasvukõvera eksponentsiaalseks. Juba 100 andmesõlme korral läheneb sooritusaeg ühele sekundile. Viimane test 1000 andmesõlmega kestis keskmiselt 68 sekundit.



Joonis 22: Iga sammu eraldi sooritusajad 250 vooelemendi ja 250 andmesõlmega.

Andmesõlmede suure ajalise keerukuse väljaselgitamiseks sooritati jõudlustest 250 voosõlme ja 250 andmesõlmega, jälgides iga sammu sooritusaega eraldi (vt joonis 22). Tulemused näitavad, et oluliselt pikemad täitmisajad ilmnevad andmesõlmedele võimalike otsenoolte otsimisel. Probleemi põhjustab see, et praegune meetod proovib naiivselt iga võimalikku defineeritud ühendust andmesõlme ja voosõlme vahel[20].

[20]Hetkel on defineeritud kokku 12×10 (12 algus-/lõpppunkti andmesõlmele ja 10 voosõlmele) võimalikku ühendust

4 Edasine arendus

Tööriista edasise arendamise käigus on oluline keskenduda noolte marsruutimise meetodi täiustamisele. Eelkõige tuleb parandada Y-paigutusmeetodit, nii et y-koordinaadid määrataks radade kaupa. Praegu võetakse kõik sõlmed samal ajal ette, mistõttu ei ole võimalik kasutada basseine ega radu. See piirab märkimisväärselt süsteemi funktsionaalsust ning takistab korrektse graafilise esituse loomist.

Praeguses lahenduses paigutatakse kirjeldused alati naiivselt andmesõlme alla, mistõttu on suur tõenäosus, et tekstid ristuvad noolte või teiste diagrammi elementidega. Soovitatav on rakendada meetodit, mis tuvastab võimalikud ristumised ning nihutab kirjelduse sobivamasse asukohta. Näiteks vasakule, paremale või üles. Eelistatud asukoht on siiski otse andmesõlme all, kuna see säilitab diagrammi loetavuse. Võimalusel tuleks kirjeldust nihutada minimaalselt, et vältida konflikte, või kui tekst on lühike, saab selle andmesõlme sisse panna.

DSL-i saab veel oluliselt täiendada. Üheks kasulikuks täienduseks oleks võimalus määratleda sõlmede suhtelisi positsioone teiste sõlmede suhtes. Näiteks saab andmeobjekti paigutada otse mõnele tegevusekohale. Selleks võiks DSL võimaldada määrata sõlme positsiooni teise sõlme suhtes. Kasutada saaks märksõnu nagu *above*, *below*, *left*, *right* koos viitega sõlme nimele (nt @andmesõlm above @tegevus1). Alljärgnevalt on toodud DSL-i näide pakutavate laiendustega:

```
# Start
X ->Lahkenmine1 "Tingimus␣1" ->Lahkenmine2 "Tingimus␣2
   ↪"

G <- Lahkenmine1
- Tegevus 1 @tegevus1
G ->Ühineminspunkt "Ühendan␣lahknemised"

G <- Lahkenmine2
- Tegevus 2 @tegevus2
G ->Ühineminspunkt "Ühendan␣lahknemised"

X<-Ühineminspunkt
. Lõpp

OD <-tegevus1 @andmesõlm
SD ->tegevus2

layout:
@andmesõlm above @tegevus1
```

Teine võimalus paigutuse täpsustamiseks oleks lubada andmesõlmede paigutuse fikseerimist nii, et need paikneksid ainult vertikaalsuunas teiste sõlmede suhtes (ülal või all). See piirang ei välistaks siiski erandjuhtumeid, mille puhul kasutatakse eelmainitud suhtelise positsioneerimise süntaksit. DSL-i süntaks võiks olla järgnev:

```
...
layout:
datanodes above
```

Lisaks tuleks DSL-i laiendada, et toetada teisi vooelemente. Kõige olulisem on sõnumisõlmede lisamine, mis eeldab olulisi muudatusi tööriista loogikas, arvestades nende eripärasid. Samuti on seni puudu artefaktide (gruppide ja kommentaaride) tugi, mille lisamine oleks vajalik tööriista funktsionaalsuse ja väljendusrikkuse suurendamiseks. Kuna kommentaarid ei ole osa protsessivoost, kuid neid paigutatakse diagrammil sarnaselt andmesõlmedega, on tõenäoliselt võimalik rakendada andmesõlmede paigutusloogikat ka kommentaaride paigutamiseks.

5 Kokkuvõte

Selle töö peamiseks eesmärgiks oli andmesõlmede toe lisamine automaatsele eksperimentaalsele BPMN paigutustööriistale, mis on suurel määral edukalt saavutatud. Andmeobjekte ja andmekogusi on võimalik defineerida ning neid paigutatakse automaatselt. Tööriista paigutusloogikat on mitmes kohas parendatud. Suurim areng üldises paigutuses tuli ILP rakendamisega y-koordinaatide määramiseks.

Siiski esineb tööriista arenduses mitmeid olulisi edasiarendamise suundi, mis on vajalikud tööriista terviklikkuse ja kasutatavuse tagamiseks. Ehkki andmesõlmede toe realiseerimine on suur areng, nõuab täisfunktsionaalse lahenduse saavutamine täiendavaid täiustusi. Esiteks vajab olulist tähelepanu graafilise paigutusloogika edasiarendus, eelkõige noolte marsruutimine. Samuti on vajalik sõnumsõlmede, artefaktide ja teiste vooelementide toe lisamine.

Järgnevalt on esitatud vastused uurimisküsimustele:

Uurimisküsimus 1: Andmesõlmede paigutus põhimõtted said defineeritud peatükis 2.1.

Uurimisküsimus 2: Andmesõlmede tugi sai väheste muudatustega Sugiyama raamistikkule tööriistale lisada. Andmesõlmed lisatakse kihtide leidmisel protsessivoogu ja mõjutavad voosõlmede paigutust vähe.

Uurimisküsimus 3: Jõudlustestide tulemused peatükis 3 viitavad kriitlisele jõudlusprobleemile andmesõlmedele otsenoolte leidmisel. See tuleks tulevikus lahendada asendades praeguse naiivse kontrolli efektiivsema algoritmiga.

Tööriista toimivuse illustreerimiseks on lisas F esitatud valik automaatselt paigutatud diagramme. Iga diagramm loodi OpenAI ChatGPT 4o mudeli abil, kasutades sisendina diagrammi definitsiooni jooniselt 11. Genereeritud diagrammid vajasid vähesel määral käsitsi kohendamist. Genereeritud diagrammid viitavad võimalusele laiendada tööriista kasutusvõimalusi, näiteks võimaldades tehisarul luua BPMN-diagramme dokumentatsiooni põhjal. Selletaolise kasutuse hõlbustamiseks tuleks kindlasti täiendada DSL-i dokumentatsiooni. Iga diagrammi eel on kirjeldatud selle definitsioon, millele järgneb vastav graafiline kujutis. Valminud tööriista saab näha prototüübi originaalse hoidla *layout-datanode*[21] harus.

Viited

[1] Ivanchikj A., Serbout S. ja Pautasso C. From text to visual BPMN process models: design and evaluation. Teoses: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. MODELS '20. Virtual Event, Canada: Association for Computing Machinery, 2020, lk. 229–239. https://dl.acm.org/doi/10.1145/3365438.3410990 (15.05.2025).

[2] Ajmal F., Wijekoon P., Dhanamina H., Ravishan Y., Nawinna D. ja Attanayaka B. Automated BPMN Diagram Generation. Teoses: *2024 6th International Conference on Advancements in Computing (ICAC)*. 2024, lk. 7–12. https://ieeexplore.ieee.org/abstract/document/10851120 (15.05.2025).

[3] Object Management Group. ISO/IEC 19510:2013 Information technology — Object Management Group Business Process Model and Notation. https://www.iso.org/standard/62652.html. (08.01.2025).

[4] Sugiyama K., Tagawa S. ja Toda M. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics* 11.2 (1981), lk. 109–125. https://ieeexplore.ieee.org/document/4308636 (15.05.2025).

[5] Sander G. A fast heuristic for hierarchical Manhattan layout. Teoses: *Graph Drawing*. Toim. Brandenburg F. J. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, lk. 447–458. https://link.springer.com/chapter/10.1007/BFb0021828 (15.05.2025).

[6] Domrös S. ja Hanxleden R. von. Diagram Control and Model Order for Sugiyama Layouts. 2024. arXiv: 2406.11393 [cs.DS]. https://arxiv.org/abs/2406.11393 (15.05.2025).

[21] https://github.com/jensjager/bpmn-parser/tree/layout-datanode

[7] Effinger P., Jogsch N. ja Seiz S. On a Study of Layout Aesthetics for Business Process Models Using BPMN. Teoses: *Business Process Modeling Notation*. Toim. Mendling J., Weidlich M. ja Weske M. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, lk. 31–45. https://link.springer.com/book/10.1007/978-3-642-16298-5 (15.05.2025).

[8] Gansner E., Koutsofios E., North S. ja Vo K.-P. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering* 19.3 (1993), lk. 214–230. https://ieeexplore.ieee.org/document/221135 (15.05.2025).

[9] Bisschop J. AIMMS optimization modeling. 2006. https://download.aimms.com/aimms/download/manuals/AIMMS3OM_LinearProgrammingTricks.pdf (15.05.2025).

[10] Peter D. hyperfine. Versioon 1.16.1. Märts 2023. https://github.com/sharkdp/hyperfine (15.05.2025).

A Sõlmede kihtide määramise ILP

Eesmärgifunktsioon:

$$\min \sum_{(v,w)\in E} (x(w) - x(v)) \tag{2}$$

Muutujad:

- V: Tippude hulk.
- $E \subseteq V \times V$: Suunatud noolte hulk, kus (v, w) tähendab noole tipust v tippu w.
- $x(v) \in Z_{\geq 0}$: Tipu v kiht.

Kitsendused:

1. Kõik tipud asuvad mittenegatiivsel täisarvulisel kihil:

$$x(v) \in Z_{\geq 0} \quad \forall v \in V$$

2. Iga noole $(v, w) \in E$ pikkus on vähemalt 1:

$$x(w) - x(v) \geq 1 \quad \forall (v, w) \in E$$

B Algne sõlmede paigutamise pseudokood

```
let y_coordinate = 0
let current_y_coordinate = 0
for pool in pools {
    for lane in pool.lanes {
        current_y_coordinate = y_coordinate
        for layer in lane {
            let x_position = layer * LAYER_WIDTH
            for node in layer.nodes {
                set_node_position()
            }
            current_y_coordinate += LAYER_HEIGHT
        }
        set_lane_dimensions()
    }
    set_pool_dimensions()
}
```

C Algne noolte marsruutimise pseudokood

```
let matrix

for node in nodes {
    matrix.add_node_position(node)
}

for edge in edges {
    let potential_start_points = find_start_points(
       ↪edge.from)
    let potential_end_points = find_end_points(edge.to
       ↪)

    let paths

    for start_point in potential_start_points {
        for end_point in potential_end_points {
            paths.push(find_path(start_point,
               ↪end_point))
```

```
            }
        }

        shortest_path = paths.
           ↪get_shortest_path_minimum_bends()
        edge.bendpoints = shortest_path
    }

    fn find_path(start_point, end_point) {
        let queue
        let paths
        while queue not empty {
            current_pos = queue.pop()

            if current_pos in end_pos {
                path = reconstruct_path()
                return path
            }

            for next_step in [current_pos.up,
            current_pos.down, current_pos.left,
               ↪current_pos.right] {
                if next_step not in_obstacle(matrix,
                next_step.position) {
                    if next_step.better_than_previous() {
                        queue.push(next_step)
                        paths.push(next_step)
                    }
                }
            }
        }

        return []
    }
```

D Andmesõlmede kihtide leidmise ILP

Reeglite loomiseks on vaja defineerida käideldavad andmehulgad.

D.1 Sõlmede ja noolte defineerimine

Olgu hulk sõlmi:

$$S = \{v_1, v_2, \ldots, v_n\},$$

ning noolte hulk:

$$E = \{e_1, e_2, \ldots, e_m\}.$$

Iga sõlm $v \in V$ kannab järgnevaid atribuute:

- id(v) – sõlme unikaalne ID;
- kiht(v) – kihi ID, kuhu sõlm kuulub;
- tüüp(v) – sõlme tüüp (nt andmeobjekt, andmekogu, sündmus, tegevus, …).

Edasi jaotatakse sõlmed kahte alamhulka. Olgu $S = \{v_1, v_2, \ldots, v_n\}$ kõigi sõlmede hulk.

- $S_A \subseteq S$: hulka S_A kuuluvad need sõlmed v, mis on kas andmeobjektid või andmekogud.
- $S_M \subseteq S$: hulka S_M kuuluvad kõik ülejäänud sõlmed (tegevused, lahknemised, sündmused jms).

Iga nool $e \in E$ kannab järgnevaid atribuute:

- algus(e) – noole algussõlme ID;
- lõpp(e) – noole sihtsõlme ID.

D.2 ILP reeglid andmeobjektide ja andmekogude paigutamiseks

Defineerime täisarv-lineaaroptimeerimise probleemi, et rakendada eelpool kirjeldatud andmeobjektide ja -kogude paigutamise põhimõtteid. Eesmärk on paigutada andmeobjektid ja andmekogud selliselt, et need järgivad eelnevalt defineeritud paigutamise põhimõtteid.

D.2.1 Indeksid ja parameetrid

S – kõikide sõlmede hulk: $S = \{v_1, v_2, \ldots, v_n\}$.

S_A – andmeelementide (andmeobjektide ja -kogude) alamhulk: $S_A \subseteq S$.

S_M – ülejäänud BPMN-sõlmede (tegevused, väravad, sündmused jms) alamhulk: $S_M \subseteq S$.

L – võimalike kihtide indeksite hulk: $L = \{1, 2, \ldots, L_{\max}\}$.

$K(v)$ – andmeelemendi $v \in S_A$ kasutajate hulk; st nende sõlmede $u \in S_M$ hulk, mis tarbivad või uuendavad antud andmeobjekti või andmekogu.

$D(v)$ – andmeelemendi $v \in S_A$ kaugus tema kasutajatest; st lühim kaugus kasutajatest kõige kaugemal asuvani.

D.2.2 Otsustusmuutujad

$x_{v,l} \in \{0, 1\}$ – binaarne muutuja, mis saab väärtuse 1, kui sõlm $v \in S$ paigutatakse kihile l, ning 0 vastasel korral.

$h_{v,l} \in \{0, 1\}$ – binaarne muutuja, mis saab väärtuse 1, kui sõlm $v \in S_A$ (andmeelement) paigutatakse poolkihile $l \to (l + 1)$, s.t kahe kihi vahele.

- Kui $h_{v,l} = 1$, siis tähendab see, et andmeelement v jääb poolkihile kihi l ja $l + 1$ vahele.
- Kui $h_{v,l} = 0$, siis see andmeelement v kas ei asu poolkihil või asub kihil määratud $x_{v,l}$ abil.

c_e – ristumiste loendamise abimuutuja nooltele.

D.2.3 Eesmärgifunktsioon

$$\min Z = \underbrace{\alpha_1 \sum_{v \in S_A} \sum_{u \in \text{Kasutajad}(v)} D_{v,u}}_{\substack{\text{(1) andmeelemendi ja tema kasutaja(de)} \\ \text{kihivahe või kauguse minimeerimine}}} + \underbrace{\alpha_2 \sum_{v \in S_A} \sum_{l \in L} h_{v,l}}_{\text{(2) poolkihtide kasutamise karistus}} \tag{3}$$

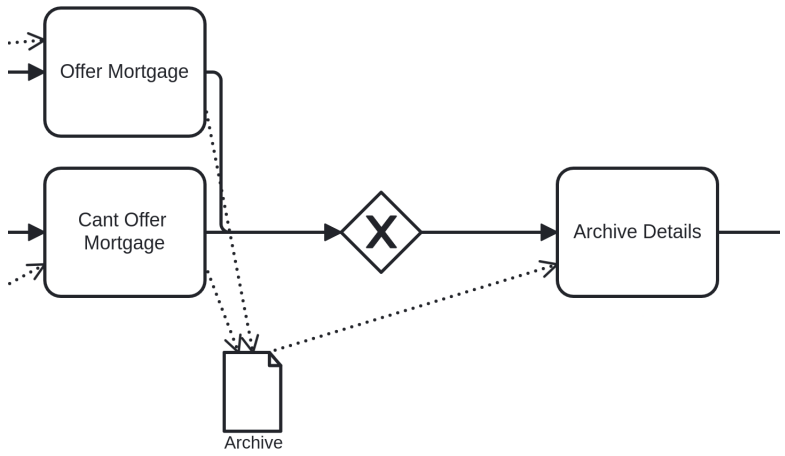
D.2.4 Kitsendused

1. Iga sõlm on täpselt ühel kihil või poolkihil. Andmeobjektide paigutamine eeldab, et need (a) paiknevad täielikult ühel kihil või (b) jäävad poolkihile kahe terve kihi vahele.

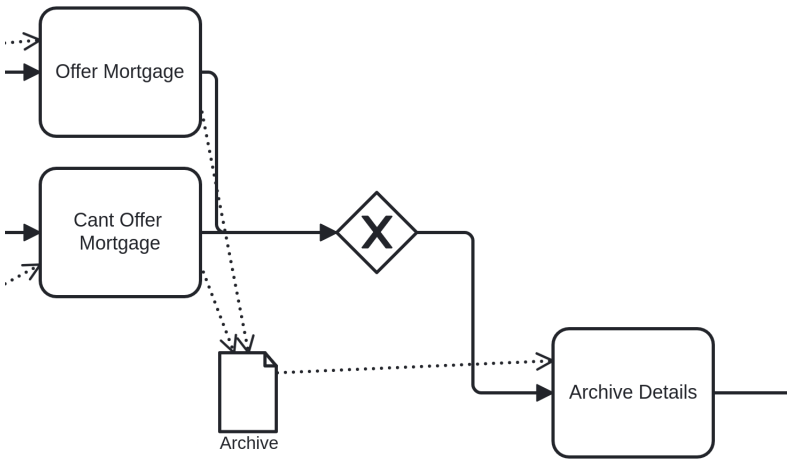
$$\sum_{l \in L} \left(x_{v,l} + \sum_{\ell \in \{l, l-1\}} p_{v,\ell} \right) = 1 \quad \forall v \in S_A. \tag{4}$$

Tingimus (4) ütleb, et andmeelement v paikneb kas ühel konkreetsel kihil ($x_{v,l} = 1$) või täpselt ühel poolkihil ($p_{v,\ell} = 1$), kus ℓ näitab, et see jääb kihi ℓ ja $\ell + 1$ vahele.

E Noolespetsiifiliste kaalude valik



Joonis 23: Noolespetsiifilised kaalud: $NOOL = 10.0$, $ANDMENOOL = 0.01$ ja $AJUTINE\_NOOL = 2.0$



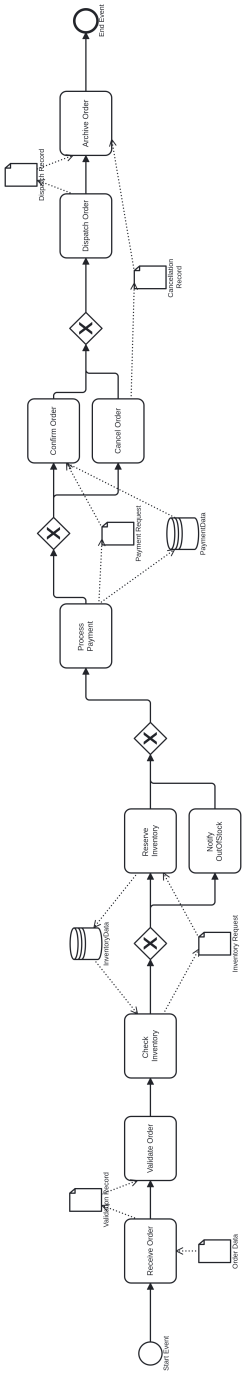
Joonis 24: Noolespetsiifilised kaalud: $NOOL = 1.0$, $ANDMENOOL = 3$ ja $AJUTINE\_NOOL = 2.0$

Vaadates joonist 23, on näha, kuidas järjestusvoo sirgust prioriseerides tekib lüüsi ja ”Archive Details” tegevuse vahel sirge nool. Kui nüüd järjestusvoo sirgust ei prioriseerita, siis vaadates joonis 24, on näha, kuidas y-koordinaadi määramise ILP toob eelnevalt mainitud tegevuse andmesõlme juurde alla.

F Automaatselt paigutatud diagrammid

```
# Start Event
- Receive Order @receiveOrder
- Validate Order @validateOrder
- Check Inventory @checkInventory
X -> InStock "In␣Stock" -> OutOfStock "Out␣of␣Stock"

G <- InStock
- Reserve Inventory @reserveInventory
G -> InventoryConfirmed "Inventory␣Confirmed"

G <- OutOfStock
- Notify OutOfStock @notifyOutOfStock
G -> InventoryConfirmed "Inventory␣Confirmed"

X<-InventoryConfirmed
```

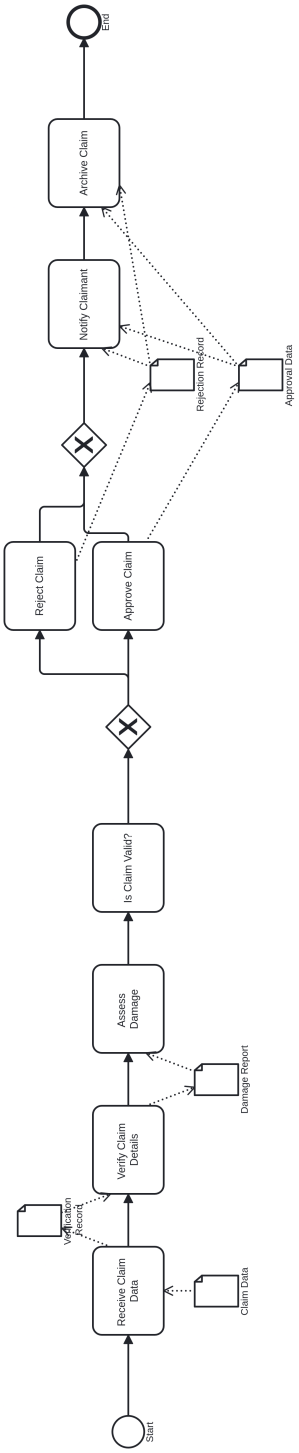
```
- Process Payment @processPayment
X -> PaymentApproved "Payment␣Approved" ->
   ↪PaymentDeclined "Payment␣Declined"

G <- PaymentApproved
- Confirm Order @confirmOrder
G -> OrderFinalized "Order␣Finalized"

G <- PaymentDeclined
- Cancel Order @cancelOrder
G -> OrderFinalized "Order␣Finalized"

X<-OrderFinalized

- Dispatch Order @dispatchOrder
- Archive Order @archiveOrder
. End Event

OD Order Data ->receiveOrder
OD Validation Record <-receiveOrder ->validateOrder
OD Inventory Request <-checkInventory ->
   ↪reserveInventory
OD Payment Request <-processPayment ->confirmOrder
OD Cancellation Record <-cancelOrder ->archiveOrder
OD Dispatch Record <-dispatchOrder ->archiveOrder

SD InventoryData <-reserveInventory ->checkInventory
SD PaymentData   <-processPayment ->confirmOrder
```



Joonis 25:

```
# Start
```

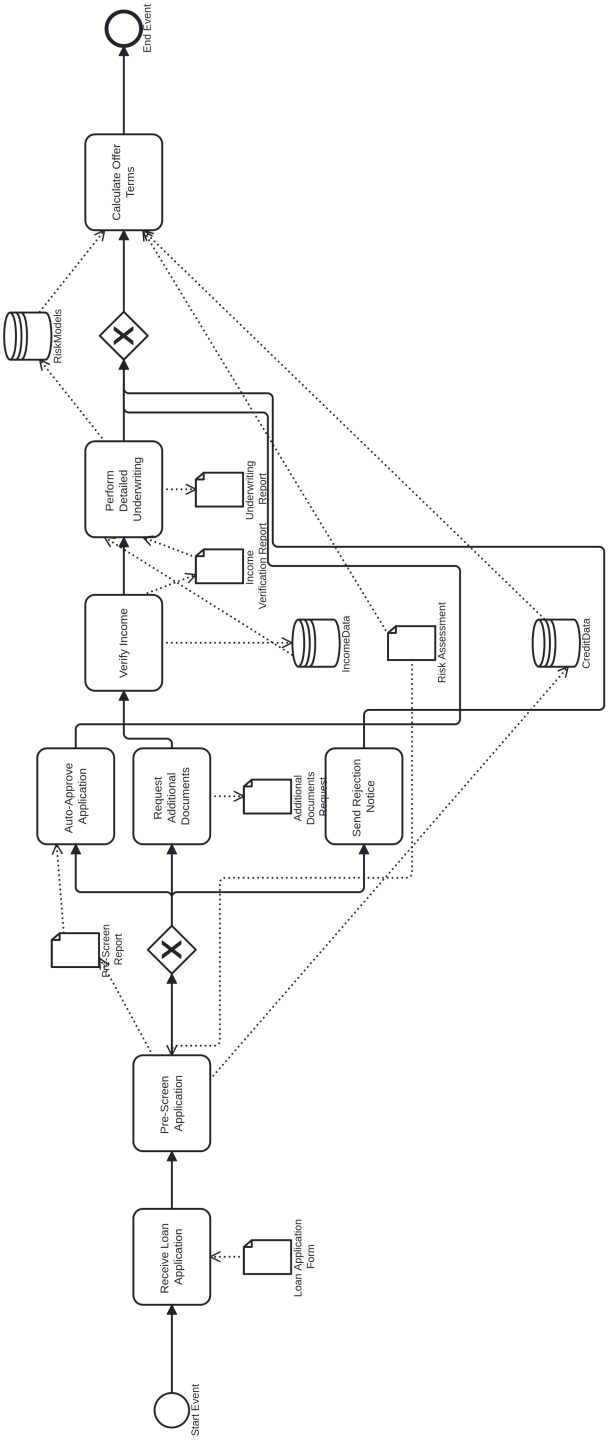
```
- Receive Claim Data @claimData
- Verify Claim Details @verifyClaim
- Assess Damage @damageAssessment
- Is Claim Valid? @claimValidity
X -> ValidBranch "Valid" -> InvalidBranch "Invalid"

G <- ValidBranch
- Approve Claim @approveClaim
G -> MergePoint

G <- InvalidBranch
- Reject Claim @rejectClaim
G -> MergePoint "MergePoint"

X<-MergePoint
- Notify Claimant @notifyClaim
- Archive Claim @archiveClaim
. End


OD Claim Data ->claimData
OD Verification Record <-claimData ->verifyClaim
OD Damage Report <-verifyClaim ->damageAssessment
OD Approval Data <-approveClaim ->notifyClaim ->
   ↪archiveClaim
OD Rejection Record <-rejectClaim ->notifyClaim ->
   ↪archiveClaim
```



Joonis 26: Automaatselt paigutatud BPMN-diagramm

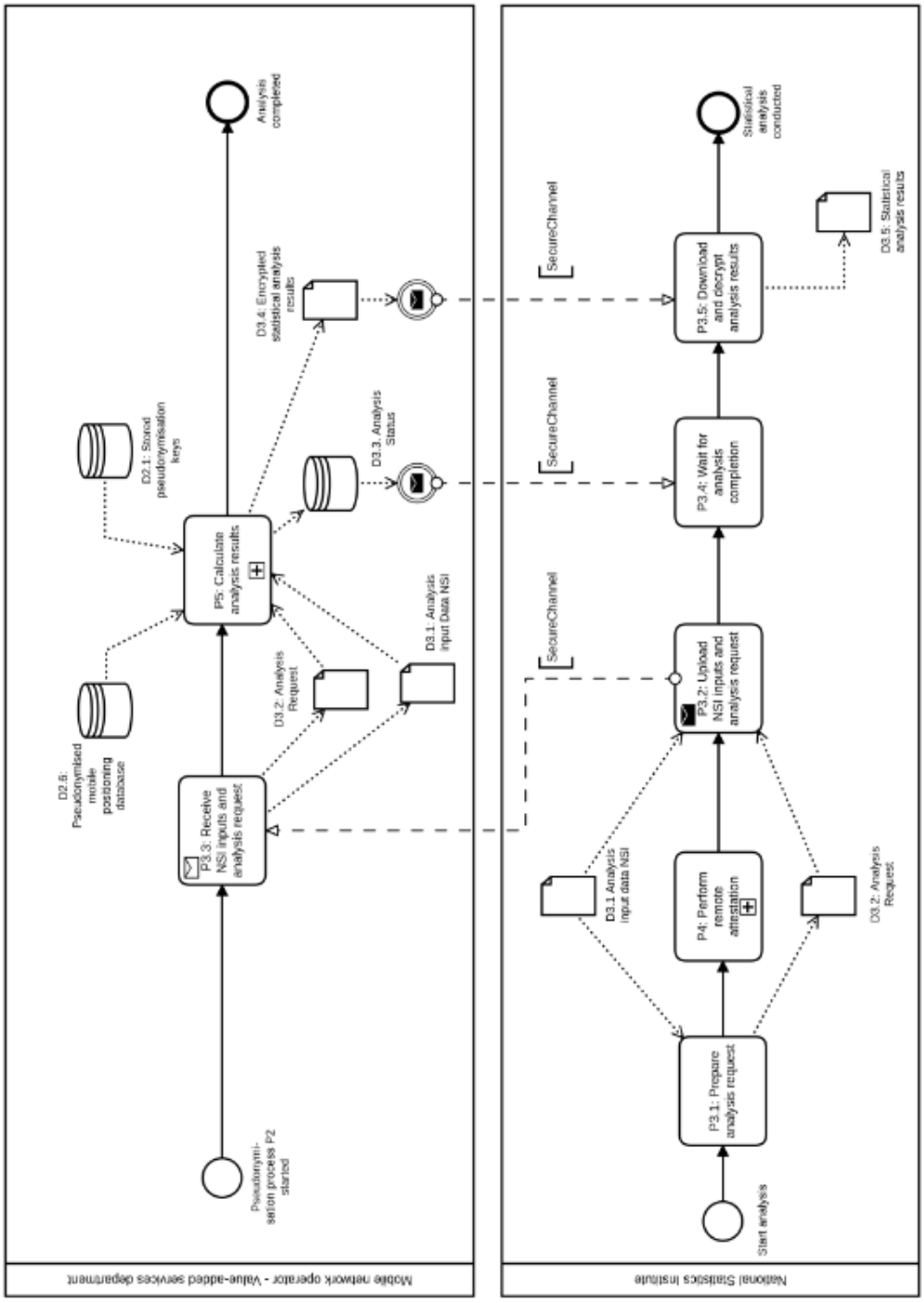
```
# Start Event
- Receive Loan Application              @receiveApp
- -PreScreen Application                 @preScreen

X -> AutoApprove  "Low␣Risk" -> ManualReview "Medium␣/
   ↪␣Incomplete␣Data" -> RejectDirect "High␣Risk␣/␣
   ↪Fraud␣Flag"

G <- AutoApprove
- -AutoApprove Application               @autoApprove
G -> AppEvaluated "Application␣Evaluated"

G <- ManualReview
- Request Additional Documents          @requestDocs
- Verify Income                         @verifyIncome
- Perform Detailed Underwriting         @underwrite
G -> AppEvaluated "Application␣Evaluated"

G <- RejectDirect
- Send Rejection Notice                 @directReject
G -> AppEvaluated "Application␣Evaluated"

X <- AppEvaluated

- Calculate Offer Terms                 @calcOffer

. End Event


OD Loan Application Form          ->receiveApp
OD -PreScreen Report              <-preScreen        ->
   ↪autoApprove
OD Additional Documents Request   <-requestDocs
OD Income Verification Report     <-verifyIncome     ->
   ↪underwrite
OD Underwriting Report            <-underwrite
OD Risk Assessment                <-preScreen        ->
   ↪calcOffer
```

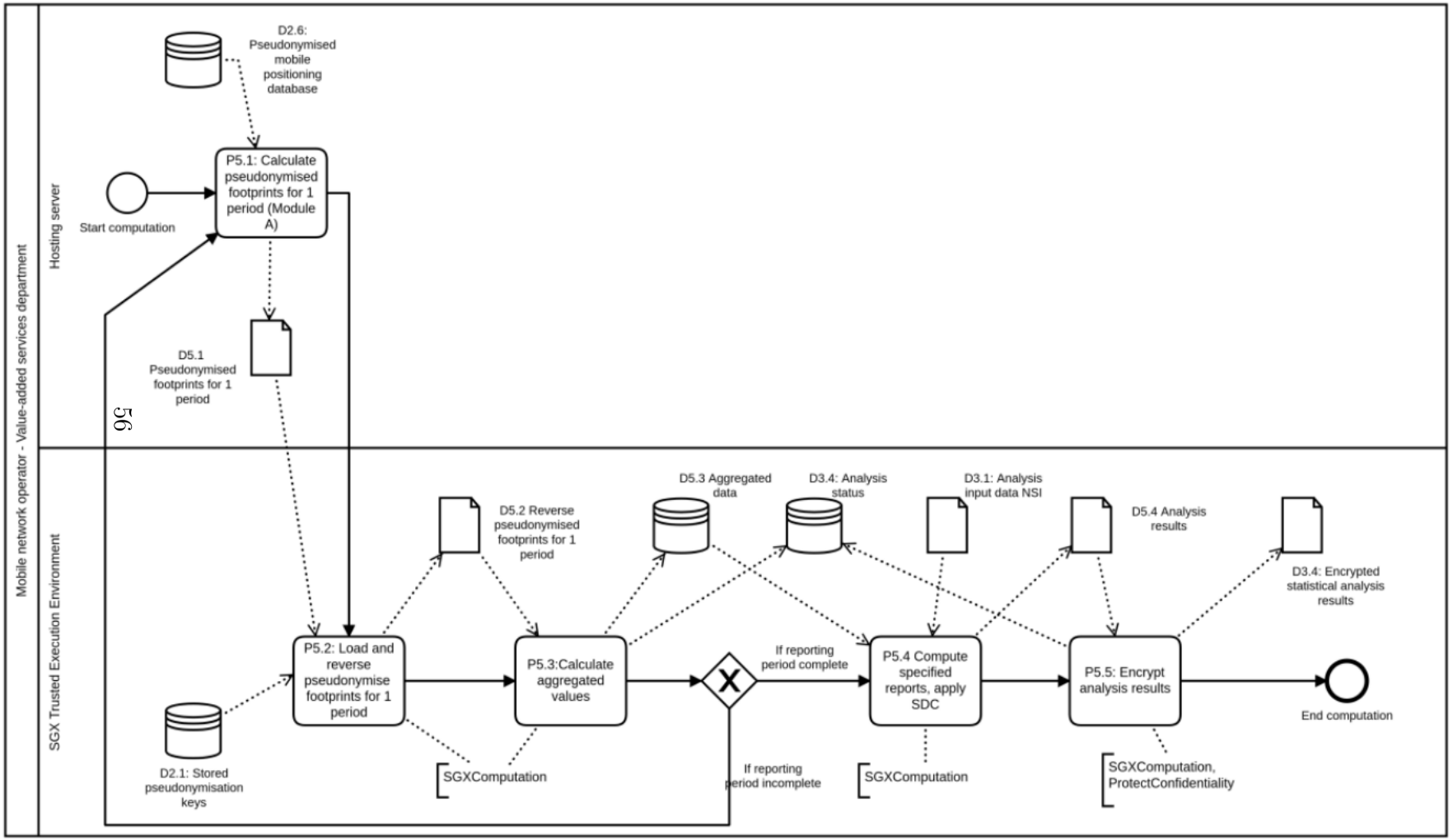
```
SD CreditData                    <-preScreen        ->
   ↪calcOffer
SD IncomeData                    <-verifyIncome     ->
   ↪underwrite
SD RiskModels                    <-underwrite       ->
   ↪calcOffer
```

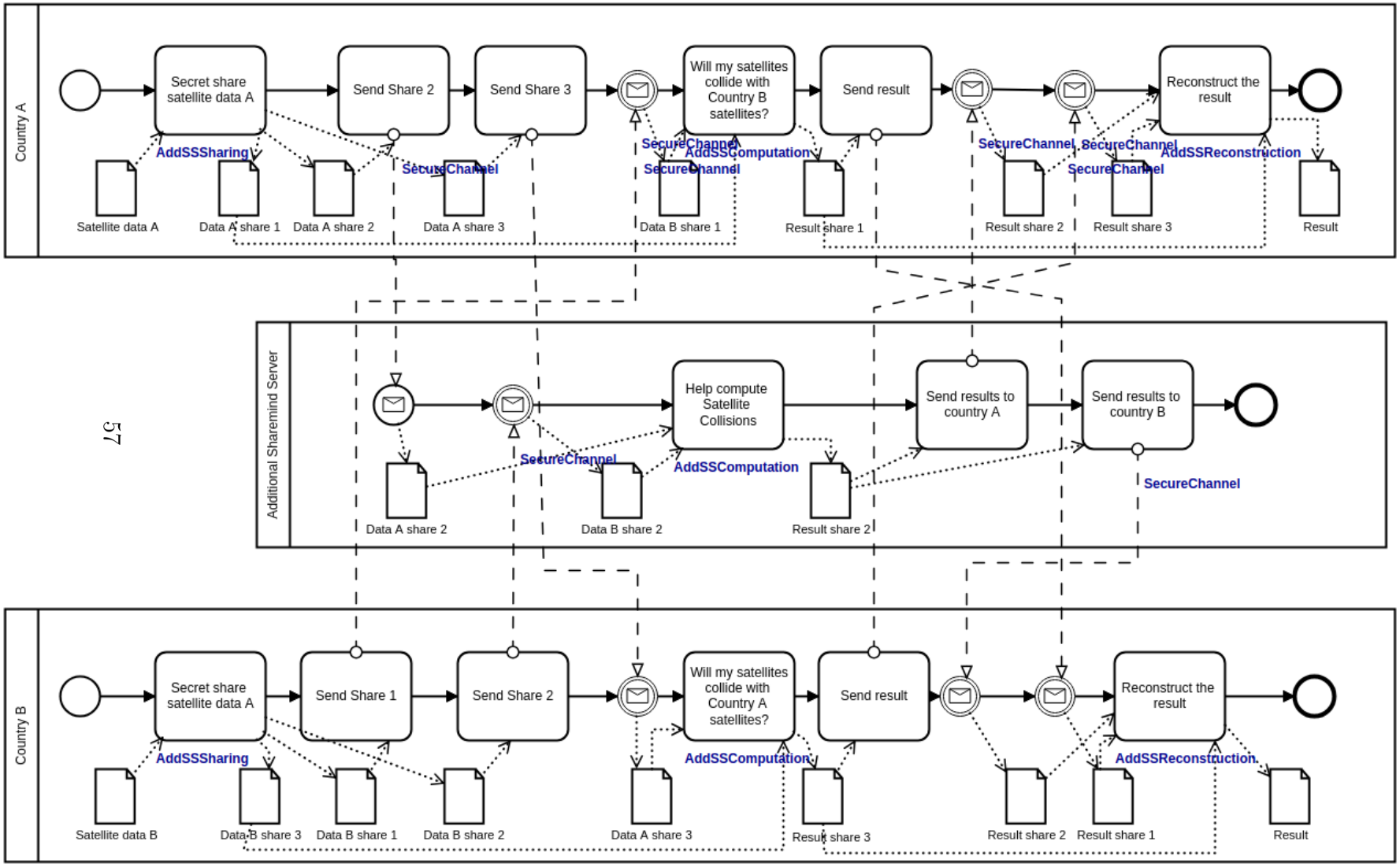


Joonis 27: Automaatselt paigutatud BPMN-diagramm

G Valik analüüsitud diagrammidest




Mobile network operator - Value-added services department
Hosting server
SGX Trusted Execution Environment
D2.6: Pseudonymised mobile positioning database
P5.1: Calculate pseudonymised footprints for 1 period (Module A)
Start computation
D5.1 Pseudonymised footprints for 1 period
D5.2 Reverse pseudonymised footprints for 1 period
D5.3 Aggregated data
D3.4: Analysis status
D3.1: Analysis input data NSI
D5.4 Analysis results
D3.4: Encrypted statistical analysis results
P5.2: Load and reverse pseudonymise footprints for 1 period
P5.3:Calculate aggregated values
If reporting period complete
P5.4 Compute specified reports, apply SDC
P5.5: Encrypt analysis results
End computation
D2.1: Stored pseudonymisation keys
SGXComputation
If reporting period incomplete
SGXComputation
SGXComputation, ProtectConfidentiality


Country A
Secret share satellite data A
Send Share 2
Send Share 3
Will my satellites collide with Country B satellites?
Send result
Reconstruct the result
AddSSSharing
SecureChannel
SecureChannel
AddSSComputation
SecureChannel
SecureChannel
SecureChannel
AddSSReconstruction
Satellite data A
Data A share 1
Data A share 2
Data A share 3
Data B share 1
Result share 1
Result share 2
Result share 3
Result
Additional Sharemind Server
Help compute Satellite Collisions
Send results to country A
Send results to country B
SecureChannel
AddSSComputation
SecureChannel
Data A share 2
Data B share 2
Result share 2
Country B
Secret share satellite data A
Send Share 1
Send Share 2
Will my satellites collide with Country A satellites?
Send result
Reconstruct the result
AddSSSharing
AddSSComputation
AddSSReconstruction
Satellite data B
Data B share 3
Data B share 1
Data B share 2
Data A share 3
Result share 3
Result share 2
Result share 1
Result

H Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks
Mina, Albert Luckas Wihler,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Automaatse eksperimentaalse BPMN paigutustööriistale andmesõlmede toe lisamine,

   mille juhendajad on Armin Daniel Kisand ja Heili Orav,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Albert Luckas Wihler
15.05.2025