LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

GRAFU PĀRLŪKOŠANA TĪMEKĻA PĀRLŪKĀ

BAKALAURA DARBS

Autors: **Aleksandrs Rilins**

Studenta apliecības Nr.: ar10073

Darba vadītājs: profesors Dr. dat. Guntis Arnicāns

RĪGA 2014

ANOTĀCIJA

Strauji attīstoties Informācijas Tehnoloģijām, pieaug arī iegūto datu apjoms, līdz ar to kļūst arvien sarežģītāka arī šo datu analizēšana un attēlošana. Viens no veidiem kā attēlot datus, ir izmantojot grafus, padarot vajadzīgo informāciju uzskatāmāku un ērtāku pārlūkošanai.

Lai atvieglotu grafu analizēšanu līdz šim tika izstrādāti vairāki risinājumi, toties pārsvarā tie ir darbvirsmas lietotnes, kuras liedz lietotājiem pilnvērtīgi piedalīties datu izvietošanā un modificēšanā.

Darba mērķis ir izpētīt grafu attēlošanas, pārlūkošanas un veidošanas iespējas tīmekļu pārlūkos, ar to saistītas problēmas un veidus kā nodrošināt sadarbību ar jau pieejamajiem un turpmāk veidotajiem rīkiem. Papildus tika apskatīti esošie risinājumi, grafu aprakstīšanas valodas, kā arī tika piedāvāts savs problēmu risinājums un izstrādāts risinājuma prototips.

Atslēgvārdi: *grafs, SVG, JavaScript, vizualizēšana, tīmekļa lietotnes*

ABSTRACT

Graph viewing in the internet browser

Every day, the IT industry has to deal with an ever-growing amount of data. The analysis and exploration of this data has become a non-trivial problem. To ease this task data can be visualized as a graph, making it more appealing for researchers.Most of the programms for graph exploration rely on specific computer platform and enforce their own work flow, providing only a static image of necessary data.

The main goal is to explore possibilities of visualizing and exploring graphs with web technologies, discuss related problems and graph description languages, review existing tools and provide a solution to the matter. Additionally, a prototype implementing these ideas is developed in the practical part of this paper.

Keywords: *graph, SVG, JavaScript, visualization, web applications*

Saturs

Apzīmējumu saraksts

Grafs – galīga objektu kopa, kuri var būt savienoti savā starpā. Abstraktus objektus sauc par grafa *virsotnēm*; saites - par *šķautnēm* (lokiem).

Hipergrafs – grafs vispārinājums, kurā ar katru šķautni var tikt savienotas ne tikai divas virsotnes, bet arī jebkādas virsotņu apakškopas.

Orientēts grafs –tāds grafs, kurā katrai šķautnei noteikts virziens un tā šķautnes veido galīgu sakārtotu pāru kopu, ko sauc par lokiem.

Virsotnes pakāpe – virsotnes kaimiņu skaits dotajā grafā. Parasti apzīmē ar $p_G(x)$, kur *G* ir grafs un *x* ir virsotne, kurai tiek aprēķināta pakāpe.

Klasterizācijas koeficients – mērvienība, kas nosaka apakšgrafa blīvumu, kur apakšgrafs sastāv no virsotnēm, kuras atrodas vienas šķautnes attāluma no izvēlētās virsotnes.

Ontoloģija – datorzinātnēs tas ir visaptverošas un detalizētas formalizācijas mēģinājums kādam zināšanu apgabalam ar konceptuālo shēmu palīdzību. Parasti tāda shēma sastāv no datu struktūrām, kuras satur visas saistītās objektu klases, to saites un likumus, kuri pieņemti šajā apgabalā.

JavaScript – skriptu valoda, kas balstīta uz prototipu koncepta un izpildās uz klienta datora. Pārsvarā tiek izmantota tīmekļu lietojumu izstrādei

JSON – JavaScript objektu aprakstīšanai domātā notācija (*angļ. val. - JavaScript Object Notation*).

XML – paplašināmā iezīmēšanas valoda, kas domāta visdažādākā veida datu aprakstīšanai globālajā tīmeklī. *(angļ. val. - eXtensible Markup Language)*

AJAX – ir savstarpēji saistīto tīmekļa izstrādes tehnoloģiju grupa, kas tiek izmantota lietotāja pusē asinhronu lietotņu veidošanai. (*angļ.val. - Asynchronous JavaScript and XML).*

SVG – vektora attēlu formāts, kas ir bāzēts uz XML struktūras. *(angļ. val. - Scalable Vector Graphic)*

OWL – zināšanu attēlošanai domāts valodu kopums. *(angļ. val. - Web Ontology Language).*

HTML5 – iezīmēšana valodas, kas domātas tīmekļa vietnes satura attēlošanai un tās strukturizēšanai, piektā versija.

Canvas – *HTML5* elements, kas ļauj veidot dinamiskus divdimensiju attēlus.

WebGL – lietojumprogrammu saskarne priekš *JavaScript* valodas, lai veidotu trīs dimensiju un divu dimensiju grafiku.

Flash – platforma priekš multivides un programmatūras izstrādes, kas tiek atbalstīta tīmekļa pārlūkos.

Silverlight – Microsoft kompānijas izstrādātais lietojumu ietvars.

Protege – atklāta pirmkoda ontoloģiju modificēšanai domāta lietotne.

ASCII – Amerikas informācijas apmaiņas standartkods. *(angļ. val. - American Standard Code for Information Interchange).*

Java – Objektorientēta programmēšanas valoda, kas atvasināta no valodas C++.

JVM – procesuālā virtuālā mašīna, kas izpilda Java bitu instrukciju kopu.

Android – operētājsistēmas uz *Linux* kodola, kas domāta ierīcēm ar skārienjutīgo displeju.

PHP – atklāta pirmkoda skriptu valoda, kas paredzēta servera puses lietojumos dinamiskai tīmekļa lapu ģenerēšanai.

PostgreSQL – objektu relāciju datu bāzes pārvaldības sistēma.

Ievads

Pateicoties straujai attīstībai sociālajos tīklos un sociālajos medijos, pieauga interese par semantiskiem tīkliem [1], kas pēc savas būtības ir orientēti grafi, kur katra grafa virsotne ir objekts no priekšmeta apgabala un grafa šķautne nosaka saistības starp objektiem. Datu attēlošana grafa veidā ir ļoti ērts attēlošanas veids tālākai datu analizēšanai, kas tiek izmantots visdažādākās zinātnes nozarēs: ķīmijā, socioloģijā, filoloģijā utt. [2].

No augstāk minētā var secināt, ka pieprasījums grafu vizualizēšanas rīkiem tikai pieaugs, bet līdz ar to pieaugs arī prasības. Jau šobrīd vairāki esošie risinājumi neatbilst visām pētnieku vajadzībām liekot tiem izmantot vairākas programmas vienlaicīgi, lai atrisinātu specifiskos apakšuzdevumus.

Darba autors izpētīja vairākus rīkus un bibliotēkas, kas domāti grafu attēlošanai un atklāja, ka gandrīz katram rīkam ir būtiski trūkumi un dažiem pat trūkst vienkāršas un acīmredzamas funkcionalitātes, kuru gaida lietotājs. Piemēram, tekošās pozīcijas noteikšana vai individuālo virsotņu vizuālā noformēšana.

Skatoties no tehniskās puses, Informācijas Tehnoloģijas nestāv uz vietas un pastāvīgi attīstās, tāpēc rodas aizvien vairāk jaunu iespēju, kā optimizēt vecus rīkus vai arī radīt radikāli jaunus risinājumus esošajām problēmām. Pagājušajā gadā, kursa darba ietvaros, veicot tīmekļa tehnoloģiju analīzi, autors nonāca pie secinājuma, ka mūsdienu tehnoloģijas dod iespēju dažas darbvirsmu lietojumprogrammas pārveidot par pilnvērtīgiem tīmeklī izvietojamiem rīkiem [3]. Darboties ar šiem rīkiem ir iespējams izmantojot populārākos tīmekļa pārlūkus, piemēram, *Chrome*, *Firefox*, *Internet Explorer* utt.. Pēc autora domām, grafu vizualizācijas rīki nav izņēmums un tos var implementēt pielietojot līdzīgu pieeju, pārnesot to tīmekļa vietnē un atvieglojot lokālas darbstacijas darbu.

Lai pamatotu savu ideju, darba ietvaros tika izstrādāta "*GrExp*" programmatūra, kas ir publiski pieejama globālajā tīmeklī. Testēšanas nolūkos prototipā tika ielādēts "ISTQB Glossary of Testing Terms" [4], kas dod lietotājiem iespēju apskatīt glosāriju grafa veidā, pāriet no viena saistīta termina pie cita, rediģēt vai arī papildināt to. "*GrExp*" ir piedāvātā risinājuma prototips un iekļauj dažas no autora idejām. Prototipa realizācijai par pamatu tika izmantota *JavaScript* valoda funkcionalitātes programmēšanai, bet grafiskajai daļai – tīmekļu pārlūkos atbalstīts vektor grafikas formāts *SVG*. Prototips sastāv no divām daļām: klienta puse – atbild par grafu attēlošanu lietotāja tīmekļa pārlūkā, servera puse – glabā testēšanas glosāriju datu bāzē un aprēķina tiešsaistē izvēlēto apakšgrafa virsotņu izvietošanu.

Darbs sastāv no 5 nodaļām. Pirmajā nodaļā tiek aprakstītas ar datu attēlošanu un grafu pārlūkošanu saistīto problēmu būtība. Darba otrajā nodaļā tiek apskatītas valodas grafu aprakstīšanai. Trešajā nodaļā tiek izvirzītas prasības grafu vizualizācijas rīkiem, ka arī tiek sniegta detalizēta analīze un novērtējms atbilstoši prasībām izmēģinātiem risinājumiem. Ceturtajā nodaļā, tiek aprakstīts autora konceptuālais risinājums, balstoties uz to tika izstrādāts prototips, kura realizācija un projektējums ir aprakstīti piektajā nodaļā. Šajā nodaļā tiek sīkāk pastāstīts par izstrādes procesu, izmantotajiem materiāliem un bibliotēkām, kā arī darbā gaitā radušos problēmu risinājumiem. Balstoties uz to, ka "*GrExp*" ir tikai sākotnējs prototips, apakšnodaļā "Tālākā attīstība" tiek apskatīta rīka iespējamā turpmākā pilnveidošana, ņemot vērā to pielietošanu citās nozarēs.

1. Problemātika

Mēdz teikt, ka viena bilde ir tūkstoš vārdu vērta un ir pilnīgi skaidrs, ka dažiem uzdevumiem (bet ne visiem) vizuāla datu attēlošana ir efektīvāka cilvēka uztverei nekā tās tekstuālais veids. Grafs kā datu attēlošanas veids ļauj apskatīt izvēlēto datu kopēju bildi un palīdz saskatīt likumsakarības starp datu vienībām.

Lielākā daļa rīku grafu un tīklu vizualizācijai piedāvā tikai statisku bildi un bieži vien tikai vienu pēc noklusējuma uzstādīto virsotņu izkārtojumu, parasti tas ir spēka-orientēts izkārtojums., kaut arī jau vairākkārt tika aprakstīti veidi kā attēlot uz plaknes dažādus grafa veidus, piemēraām orientētu grafu [5]. Lietotājam liegtā iepēja pašam pārkārtot grafa izkārtojumu vai vismaz izvēlēties citu izkārtojumu, traucē viņam atklāt šablonus grafā, balstoties uz specifiskas elementu izvietošanas.

Daudz nopietnāka problēma ir virsotņu attēloto atribūtu ierobežošana. Lai labāk saprastu šīs problēmas būtību, apskatīsim konkrētu piemēru. Mūsdienās ļoti aktuāla ir semantisko tīklu pētīšana, lai atvieglotu šo procesu, datus mēģina vizualizēt, un viena no pieejām ir attēlot tos kā grafu. Semantiskos tīklus pielieto, piemēram, lai aprakstītu terminu ontoloģijas, kur katram terminam vai jēdzienam var būt piekārtotas vairākas objektu klases. Savukārt, objektu klasēm piemīt vairāki atribūti. Vizualizējot tādus datus, grafa virsotne atbilst kādam ontoloģijas terminam vai jēdzienam, un to apzīmē grafā ar kādu vienkāršu ģeometrisko figūru – apli vai kvadrātu. Izmantojot esošos rīkus, lietotājs var nonākt situācijā, ka grafā atribūti, kas pieder konkrētai virsotnei pavisam nav redzami vai ir redzama tikai daļa no tiem. Tāda pieeja ir saprotama, jo mēģinājums attēlot virsotnēs visus ar to saistītos datus, var novest pie situācijas, kurā grafs paliks slikti izkārtots un vairākas virsotnes pārklāsies. Lai izvairītos no šīs situācijas, var savlaicīgi aprēķināt katras virsotnes izmērus, kopā ar visiem datiem, bet pat pienācīgi izkārtojot grafu, tas tik un tā paliks pārpildīts ar informāciju un būs lietotājam nelasāms.

Kā daļējais risinājums ir tekstuālā veidā attēlot viena atribūta vērtību atbilstošā virsotnē vai blakus tai. Toties lietotājiem jābūt iespējai vismaz izvēlēties kādus virsotnes datus attēlot, bet vēl labāk redzēt pēc iespējas vairāk informācijas, lai novērtētu situācijas īstos apmērus, tāpēc šāda pieeja neder.

Kā jau iepriekš bija minēts, ļoti iespējams, ka vajadzīgie dati iedalās grupās (piemēram, ontoloģijā tas ir objektu klases) un ir ļoti būtiski attēlot atbilstošas virsotnes piederību kādai no tām. Var notikt arī tā, ka rīks apstrādās lietotāja datus un attēlos virsotņu piederību rīkā definētām klasēm, nevis lietotāja norādītājām (piem., tā notiek vizualizācijas pievienojumprogrammās priekš *Protege* lietotnes), bet tas ir sliktas arhitektūras piemērs un tādu

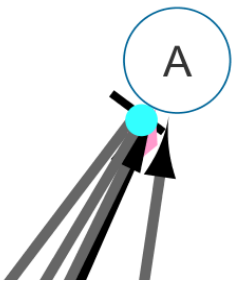
uzvedību var sastapt ļoti reti. Bieži vien virsotnes piederība attēlā tiek parādīta izmantojot virsotnes pildījuma krāsu un apmales vilkuma veidu, bet šāds risinājums nevar parādīt virsotnes piederību vairākām klasēm vienlaicīgi! Situācija, kad virsotnei atbilstoši dati pieder vairākām apakškopām, esošajiem rīkiem sagādā lielas problēmas un tie ne vienmēr spēj tikt ar to galā [6]. Tas apgrūtina lietotāju mēģinājumus izdalīt datus pēc piederības un var kļūt par nopietnu šķērsli darba veikšanai.

Izraisīt problēmas var arī gadījums, kad vajadzīgajā datu apgabalā ir datu objekta pilnīgs vai daļējais dublikāts, kad sakritība nav pilnīga, bet tikai pēc kāda konkrēta parametra. Piemēram, glosārijā tie varētu būt terminu sinonīmi, kas rakstās savādāk, bet pēc jēdziena definīcijas ir vienādi, vai arī, kad vienam un tam pašam terminam ir vairākas definīcijas. Pirmajā gadījumā visticamāk dublikātiem tiks piekārtotas liekas šķautnes, kas padarīs grafu pārpildītu, bet otrajā gadījumā tiks attēlotas vairākas virsotnes ar vienu un to pašu iezīmi, kas lietotājam apjukumu var radīt apjukumu.

Līdzīgas problēmas var konstatēt pievēršot savu uzmanību grafa šķautņu attēlošanai. Ja saitēm starp datu vienībām ir specifiski parametri, kā tos visus attēlot? Izmantojot esošos risinājumus, vairākos gadījumos tie netiks parādīti grafā.

Attēlojot liela apjoma grafu, par problēmu var kļūt šķautņu krustošanās, tātad jābūt iespējai izvēlēties planāro grafa izkārtojumu. Iespējamais risinājums varētu būt locīto līniju izmantošana taisno līniju vietā, bet tas ir ne tik daudzsološs uzlabojums.

Vēl vairāk, lietotājam var rasties vajadzība attēlot datu saistību ne tikai pēc virziena, bet arī pēc kāda cita kritērija, piemēram, mantošanas gadījumā, noderētu "*is-A*" saite. Tātad jāpatur prātā arī vairāku šķautņu galu attēlošanas veidus, lai attēlotu šķautnes lomu, kas savukārt var novest, pie gadījuma, ka vienā virsotnē ienāk pārāk daudz šķautnes, un lietotājiem būs grūti vizuāli noteikt kādam tipam pieder atsevišķa šķautne (Att.1.1).



1.1. att. **Šķautņu parklāšanās problēmas piemērs**

Vēl viens sarežģījums ir fakts, ka vairākums rīku ir instalējami, kas uzreiz atklāj divus trūkumus:

- lietotājs ir spiests pieturēties pie specifiskās operētājsistēmas.
- lietotāja darbstacija tiek noslogota, jo grafu izkārtojuma aprēķināšana pie lieliem datu apjomiem ir diezgan pret resursiem prasīgs process.

Analizējot grafu attēlošanas risinājumus, autors nonāca pie secinājuma, ka ir izplatīta prakse, piedāvāt lietotājiem saglabāt grafus dažādākos formātos, rīku specifiskos, lai viņi spētu turpināt iesākto darbu, un, protams, attēlu formātos, piemēram: *PNG*, *SVG*, utt.. Autoram radās jautājums, kādas ir lietotāja iespējas pārnest grafu no vienas lietotnes uz citu un kādi grafu aprakstīšanai domātie formāti ir pieejami šī uzdevuma atrisināšanai. Izrādās, ka formāti ir vairāki un tie ir aprakstīti darba 2. nodaļā. Eksistējošiem lietojumiem atbalstīto formātu klāsts ir atšķirīgs, tāpēc būtu aktuāli piedāvāt lietotājiem ērtu risinājumu, kas prot veidot jaunus grafus, modificēt esošos, importēt un eksportēt grafus priekš esošajiem rīkiem un pamēģināt nodrošināt datu izmantošanu arī turpmāk izstrādātajos rīkos.

2. Grafus aprakstošas valodas

Pastāv vairākas grafu aprakstīšanai domātas valodas, bet šajā nodaļā tiek aprakstītas tikai visizplatītākās no tām. Autors izvēlējās tikai tās valodas, kuras tiek atbalstītas vismaz divos no tālāk apskatītajiem informācijas vizualizēšanas risinājumiem.

2.1. GML

Grafu modelēšanas formāts (*angļ.val. - Graph Modelling Language*) ir uz *ASCII* pamata veidots datņu formāts, lai definētu grafus kā abstraktus datus [8]. *GML* struktūra ir ļoti vienkārša un sastāv no atslēgvārdu-vērtību pāru virknes. Atslēgasvārdi ir simbolu virknes, kas sastāv no burtu vai ciparu kombinācijām, piemēram: "*virsotne*" (*angļ.val. - node*) vai "*šķautne*" (*angļ.val. - edge*). Savukārt, vērtību norādīšanai tiek pieejami vairāki datu tipi: peldoša komata skaitļi, simbolu virknes (ierobežotas ar pēdiņām no abām pusēm) vai arī atslēgvārdu-vērtību pāru saraksts, kas tiek ierobežots ar kvadrātiekavām.

Formātu ir iespējams paplašināt pievienojot jaunus atslēgvārdus, jo parasti visi atslēgvārdi, kas nav aprakstīti sākotnējā specifikācijā, tiek ignorēti, tomēr ir jāievēro kopējos sintakses noteikumus [8]. Pēc noklusējuma, grafu apraksta ar trim pamatatslēgām: "*grafs*" (*angļ.val. - graph*), "*virsotne*" un "*šķautne*", kur pēdējie divi ir "grafa" bērni. Virsotnēm obligāti jābūt "*id*" atslēgvārdam ar unikālu vērtību visā grafā, toties šķautnēm jābūt atslēgvārdam "*avots*" (*angļ.val. - source*) un "*mērķis*" (*angļ.val. - target*), kuru vērtības ir savienotu virsotņu identifikatori. Ja "*grafa*" elementam tiek definēts atslēgvārds "*orientēts*" (*angļ.val. - directed*), tad šķautņu virziens tiek noteikts pieņemot, ka šķautne iziet no "*avota*" virsotnes un ieiet "*mērķa*" virsotnē. No tā seko, ka vai pilnīgi visām šķautnēm ir virziens vai arī nav nevienai no tām.

```
graph [
       node [
              id 7
              label "5"   ]
       node [
              id 15
              label "13"  ]
       edge [
              label "24"
              source 7
              target 15
]]
```

2.1. att. **GML sintakses piemērs**

2.2. GraphML

GraphML ir uz *XML* bāzēts dokumenta formāts, kura pirmā rinda definē kādu struktūras standartu, dokuments ievēro arī kāds ir tā kodēšanas veids, kas ir tipisks visiem *XML* dokumentiem.

Valodas saknes elements ir "*graphml*", kas ietver sevī visbūtiskāko dokumenta informāciju. Grafa aprakstīšana sākas ar "*grafa*" elementu, kas sastāv no vairākiem "*virsotņu*" un "*škautņu*" elementiem patvaļīgā secībā. Salīdzinot ar *GML* valodu, *GraphML* atbalsta orientēto un parasto šķautņu maisījumu, specificējot grafa elementam virzienu pēc noklusējuma ar obligātu "*edgedefault*" atribūtu. Ja šķautnes elementam netiek definēts virziens, tad tai tiek piešķirta "*edgedefault*" vērtība, kura var būt viena no divām: "*orientēta*" un "*neorientēta*" (*angļ.val. - undirected*).

Visas virsotnes tiek aprakstītas ar "*virsotnes*" elementu, un obligāta prasība ir identifikatora atribūta "*id*" definēšana, kuram jābūt unikālam visā *GraphML* dokumentā.

Šķautnes tiek aprakstītas ar "*šķautnes*" elementu, kuram obligātie atribūti ir "*avots*" un "*mērķis*", lai noteiktu ar šķautni savienotas virsotnes. Definējot tikai vienu galapunktu, tiek uzskatīts, ka šķautne ieiet un iziet no vienas un tās pašas virsotnes, tādas šķautnes tiek sauktas par "*cilpām*" (*angļ.val. - loop*) [8]. Tāpat kā virsotnēm, šķautnēm var norādīt "*id*" atribūtu, lai atvieglotu turpmāko pieeju šai šķautnei.

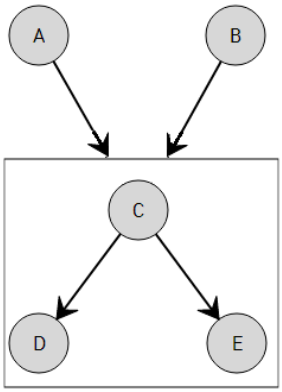
Papildinformācijas iekļaušanai tiek izmantots "*GraphML-Attributes*" paplašinājums, kas ļauj definēt jaunus atribūtus ar "*atslēgas*" (*angļ.val. - key*) elementa palīdzību, un kuram ir daži obligātie atribūti:

- "*id*" - identifikatora atribūts, kas tiek izmantots, lai noteikt atribūta nosaukumu vērtības piešķiršanas momentā. Tam jābūt unikālam visā dokumentā.
- "*for*" – nosaka kāda tipa elementam jaunizveidotais atribūts ir domāts. Iespējamas vērtības: virsotne, šķautne, grafs, visi.
- "*name*" – nosaukums, kas izskaidro atribūta nozīmi un mērķi. Tam jābūt unikālam visā dokumentā.
- "*type*" – atribūta vērtības tips, kas var būt viens no sekojošiem: Būla tips, vesels skaitlis, skaitlis ar peldošo punktu vai simbolu virkne.

Atribūta vērtība tiek piešķirta izmantojot "*data*" elementu. Tā "*atslēgas*" atribūta vērtība ir atbilstoša jaunizveidotajam atribūta identifikatorams, kas nosaka kāda tipa atribūtu tas reprezentē. "*Data*" elements tiek iekļauts atbilstošā tipa elementā, piemēram, virsotnē vai šķautnē.

Lai specificētu vērtību pēc noklusējuma, elementā "*atslēgas*" iekļauj "*noklusēto*" (*angļ.val. - default)* elementu. Gadījumā, ja jauns atribūts tiek definēts, bet netiek izmantots grafā, tad šī elementa jaunizveidotam atribūtam tiks piešķirta noklusētā vērtība [8]. Piemēru var redzēt 1.pielikumā.

Papildus *GraphML* atbalsta apakšgrafus ar ligzdošanas struktūras palīdzību. Katrs virsotnes elements var saturēt grafa elementu, kurš apraksta apakšgrafa virsotnes un šķautnes.



2.2. att. **GraphML Apakšgrafa piemērs**

2.3. XGMML

Dokumenta formāts, kas apvieno GML struktūru un XML 1.0 standarta specifikācijā aprakstītas vadlīnijas. Saknes elements ir "*grafs*", kurš var saturēt vairākas "*virsotnes*", "*šķautnes*" un "*att*" elementus. Pirmie divi no šiem elementiem tiek izmantoti atbilstošas grafa sastāvdaļas definēšanai, bet pēdējais tiek izmantots, lai norādītu papildinformāciju virsotnei vai šķautnei.

Korektam *XGMML* dokumentam "*att*" elementiem jābūt sākumā vai beigās, savukārt, virsotņu un šķautņu elementu savstarpējai secībai nav nozīmes. Grafa elements, kas ir iekļauts "*att*" elementā tiek uzskatīts par apakšgrafu. Katrai virsotnei jābūt unikāliem "*id*" un "*nosaukuma*" atribūtiem. Definējot šķautni, jāatceras, ka tā nevar norādīt uz virsotnēm, kuras nav minētas šajā dokumentā.

Grafa elementam iespējams norādīt vairākus atribūtus: ar Būla tipa atribūtu "*orientēts*" var norādīt grafa tipu; "*Saknes virsotne*" " (*angļ.val. - rootnode)* domāta, lai noteiktu programmu, ar kuru tika sastādīts dokuments; "*Izkārtojums*" " (*angļ.val. - layout)* pasaka, ar kādu algoritmu tika izkārtots dokumentā aprakstītais grafs; "*grafika*" (*angļ.val. - graphic)* - ir vēl viens Būla tipa atribūts, kurš nosaka vai dokuments satur datus par grafa vizuālo noformējumu.

Katram grafa elementam obligāti ir unikālo identifikatoru saturošs atribūts "*id*", nosaukumam domāts "*name*", iezīmes definēšanai izmantotais "*label*" un iezīmes pozīcijas noteikšanai atbilstoši elementam "*labelanchor*" atribūts.

Virsotnes elementam pastāv vēl divi specifiskie atribūti – "*edgeanchor*", kas apraksta šķautnes pozīciju atbilstoši virsotnes elementam un "*svars*" " (*angļ.val. - weight),* kas tiek izmantots dažos grafu izkārtošanas algoritmos (*piem.: atsperes modeļa izkārtošana).*

Korektai šķautnes definēšanai dokumentā jābūt vismaz divām virsotnēm, kuru identifikatori tiek norādīti šķautnes "*avota*" un "*mērķa*" atribūtos. Tapāt kā virsotnei, šķautnei ir "*svara*" atribūts priekš specifiskiem izkārtojuma algoritmiem.

Metadatu parametru glabāšanai tiek izmantots "*att*" elements, kuram ir tikai trīs atribūti:

- "*nosaukums*" – apraksta parametra nozīmi.
- "*vērtība*" – satur norādīto parametra vērtību .
- "*tips"* – norāda parametra datu tipu .

XGMML atbalsta "*att*" elementa ligzdstruktūru, lai aprakstītu sarežģītākus metadatu veidus.

Grafikas elements nodrošina iespēju aprakstīt virsotņu un šķautņu attēlojumu. Šis elements var būt iekļauts tikai virsotnes vai šķautnes elementā, savukārt tajā varbūt iekļauti "*att*", "*līnijas*" un "*centra*" elementi. Izmantojot grafikas elementa atribūtus var norādīt augstumu un platumu, teksta fontu, aizpildījuma un apmales krāsas. Līnijas elementa atribūti ļauj definēt divus punktus starp kuriem tiek novilkta līnija un līnijas attēlošanas veidu: biezumu, kā bultu utt.

```
<?xml version="1.0"?>
<!DOCTYPE graph PUBLIC "-//John Punin//DTD graph description//EN"
"http://www.cs.rpi.edu/~puninj/XGMML/xgmml.dtd">
    <graph directed="1" id="42" label="Test graph">
          <node id="1" label="A"></node>
          <node id="2" label="B"></node>
          <node id="3" label="C"></node>
          <edge source="1" target="2" label="A->B"></edge>
          <edge source="2" target="3" label="B->C"></edge>
    </graph>
```

2.3. att. **XGMML grafa piemērs**

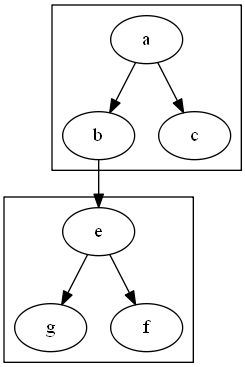
2.4. DOT

Gan cilvekiem, gam dartoriem vViegli lasāma un saprotama valoda grafu aprakstīšanai. Lai sāktu grafa sastādīšanu, jāizmanto "*graph*" atslēgvārds, kam seko figūriekavas, kurās ir uzskaitīts priekšrakstu saraksts. Katrs priekšraksts apraksta kādu noteiktu grafa sastāvdaļu,

piemēram, virsotnes definēšanai jānorada virsotnes iezīmi, kas kalpos tai, kā unikāls identifikators. Tālāk var sekot atribūtu saraksts, kurš apraksta virsotnes vizuālo noformēšanu un satur saistīto papildinformāciju. Šķautnes tiek definētas ar virsotnēm, kuras tā savieno. Vispirms tiek norādīts virsotnes identifikators pēc tam ar "-", "<" un ">" zīmēm aprakstīts šķautnes veids. Piemēram, "*A* -- *B*" atbilst neorientētai šķautnei, savukārt "*A* -> *B*" šķautnei, kas iziet no virsotnes "*A*" un ieiet virsotnē "*B*". Viena no priekšrocībām šai sintaksei ir fakts, ka virsotņu definēšana pirms šķautņu aprakstīšanas nav obligāta. Ja šķautnes priekšrakstā tiek izmantota virsotne, kas līdz šim netika minēta, tad to ir jāizveido, izmantojot grafa noklusētas vērtības.

Tapāt kā *GraphML* valodā - *DOT* atbalsta apakšgrafu aprakstīšanu, izmantojot "*apakšgrafa*" (*angļ.val. - subgraph)* atslēgvārdu, pēc kura tiek definēts apakšgrafa identifikators. Pēc tam, ierobežojot ar figūriekavām, tiek aprakstīts apakšgrafs ievērojot tos pašus noteikumus, ko izmanto parastam grafam.

Būtisks šīs valodas trūkums, ka apakšgrafs netiek uzskatīts par vecāka grafa virsotni. No tā seko, ka nevar būt šķautnes, kas savieno divus apakšgrafus. Var savienot tikai apakšgrafa virsotnes, šādu situāciju var redzēt attēlā 2.3.



2.4. att. **DOT apakšgrafa piemērs**

Vēl viens pielietojums "*apakšgrafa*" atslēgvārdam - jau esošo vecāka grafa virsotņu sadalīšana apakškopās. Šāda pieeja dod iespēju apvienot virsotnes grupā un piešķirt to atribūtiem kopīgo vērtību. Piemēram, var izdalīt virsotņu grupu ar vienu krāsu, apvienojot visas izvēlētās virsotnes apakšgrafā, un norādot apakšgrafam noklusēto virsotņu aizpildījuma krāsu.

3. Esošie risinājumi

Šajā nodaļā tiek detalizēti aprakstīta esošo rīku veiktā analīze un novērtējums, balstoties uz kuriem, darba 3. nodaļā autors piedāvā savu konceptuālo problemas risinājumu.

3.1. Iepriekšējā izpēte

Iepriekšējā gadā, kursa darba [3] ietvaros autors veica ieskatu tīmekļu tehnoloģijās, kurā tika apsvērtas iespējas attēlot divdimensiju grafiku tīmekļu pārlūkos. Papildus tika apskatīti konkrētie rīki un bibliotēkas, kas manāmi atvieglotu un veicinātu šo procesu. Starp apskatītajām tehnoloģijām bija: *HTML5, Flash, Silverlight* un citas pieejas. Toreiz tika secināts, ka divi optimālākie veidi priekš vizualizēšanas [3] ir:

- SVG - vektor grafikas formāts.
- HTML5 - iezīmēšanas valodu ar *canvas* elementa atbalstu.

Šis pieejas tika izvēlētas, jo, salīdzinot ar alternatīviem veidiem, tām ir raksturīgs:

- minimālas prasības lietotāja tīmekļu pārlūkiem,
- mazas noslodzes lietotāju datoriem atveidojot grafiskos attēlus,
- ērta piekļūve jau izveidotiem grafiskiem objektiem, izstrādājot vizualizācijas lietotnes.

Ņemot vērā kursa darbā iegūtos rezultātus [3], darba autors no visiem pieejamiem tīmekļa rīkiem, kas domāti datu attēlošanai, nolēma veikt analīzi tikai tiem, kuri par pamatu izmanto vienu no augstāk minētām tehnoloģijām.

3.2. Risinājumu obligātās prasības

Pirms veikt risinājumu salīdzināšanu un pētīšanu, tiks formulētas minimālās prasības lietojumu funkcionalitātei, kas kalpos par atskaites punktu vērtējot esošos rīkus.

Gandrīz visas lietotnes tiek radītas, lai atvieglotu lietotāju darbu, tāpēc jāsaprot kādas vajadzības ir cilvēkiem, kas strādā ar datu vizualizēšanu. Acīmredzamā esošo rīku problēma ir zems saskarnes interaktivitātes līmenis. Dažos rīkos trūkst pavisam vienkāršas funkcionalitātes, kura mūsdienu lietotājam liekas intuitīva, kuru viņš sagaida jebkurās lietotnēs, un ne tikai datu vizualizēšanas rīkos. Spilgts piemērs ir virsotņu vizuālā noformējuma modificēšanas iespējas trūkums *Protege* pievienojumprogrammās.

Par pamatu var kalpot uzdevumi, kuru izpildīšanu piedāvāja nodrošināt izstrādājot informācijas vizualizēšanas rīkus, Bens Šneidermans [2]. Savā publikācijā viņš uzsvēra, ka šāda tipa lietotnēm jābūt:

- Pārskata atbalstam – parādīt informācijas kopējo bildi.
- Tālummaiņai – iespējai izmainīt fokusa attālumu attēla apskatīšanai ekrānā no tālplāna uz tuvplānu vai pretēji.
- Attēlotu objektu filtrācijai, lai lietotājs redzētu tikai viņam vajadzīgo.
- Detaļu attēlošanai pēc pieprasījuma – lietotājs izvēlas kādas datu detaļas parādīt par objektiem visā attēlā vai par konkrētu izvēlēto objektu grupu.
- Attiecibām – parādīt saistības starp datu objektiem.
- Vēsturei – rīkam jāglabā veikto darbību hronoloģiskā secība, lai sniegtu iepriekšējo darbību vai izmaiņu atcelšanu pēc lietotāja pieprasījuma.
- Izvilkšanu – izvēlētās informācijas daļas vai pēc pieprasījuma ar parametriem atlasītas informācijas eksportēšana.

Uzreiz vajag atzīmēt, ka šie uzdevumi bija domāti datu vizualizēšanai ne tikai grafa veidā, tāpēc uzdevums par objektu saistību attēlošanu automātiski izpildās veidojot grafu, datu attiecības tiek attēlotas jebkurā gadījumā. Vairākums no šiem uzdevumiem mūsdienu rīkiem liekas pašsaprotami, bet, kā rāda prakse, ne vienmēr.

Sastādīto funkcionalitātes sarakstu nepieciešams papildināt ar sekojošiem uzdevumiem, kas ir specifiski šajā darbā apskatītai problēmai:

- Vairāku grafa izkārtojuma veidu atbalsts un iespēja izvēlēties starp tiem.
- Lietotājam jābūt iespējai mainīt atsevišķas virsotnes pozīciju, lai izveidotu individuālo grafa izkārtojumu.
- Ērta grafa aprakstīšanai domāta formāta atbalsts, gan importēšanai, gan eksportēšanai.
- Jāsniedz iespēju papildināt grafu ar jauniem datu objektiem.
- Jāsniedz iespēju modificēt attēlotu objektu datus.
- Grupēšanas atbalsts – iespēja apvienot vairākas virsotnes grupās.
- Risinājumam nevajadzētu būt lielām prasībām pret resursiem, lai to varētu darbināt uz veciem datoriem.
- Grafa virsotņu attēlojuma veida mainīšana – mainīt grafa virsotņu vizuālo formu, aizpildījumu krāsu, utt.

- Vairāku šķautņu attēlojuma veidu atbalsts – rīkam jāprot attēlot šķautņu virzienu, ja tas ir nepieciešams un jāprot attēlot dažādus saistību tipus. Papildus lietotājam jābūt iespējai mainīt šķautnes tipu jau attēlotajā grafā.
- Navigācijas atbalsts – jābūt iespējai pārvietoties starp grafa virsotnēm, parejot no vienas virsotnes pie otras.
- Meklēšanas atbalsts – funkcionalitāte, kas palīdz atrast lietotājam specifisko virsotni grafā. (Atšķirībā no filtrācijas, parāda meklētās virsotnes atrašanos sākotnējā grafā, nevis atlasa to atsevišķā apakšgrafā).
- Rīkam jābūt pieejamam globālajā tīmeklī un lietotājam jābūt iespējai dalīties ar izveidotiem grafiem.

Šīs saraksts nav pilnīgs un var tikt papildināts. Tas tika sastādīts tikai, lai kalpotu kā salīdzināšanas mērs, vērtējot esošo rīku atbilstību lietotāja vajadzībām.

3.3. GVedit

Grafu attēlošanas rīks, kas ietilpst *GraphViz* atvērta pirmkoda pakotnē, kura ir instalējama un atbalsta gandrīz visas izplatītākās operētājsistēmas, toties nav pieejama priekš mobilām ierīcēm.

Lietotne sagaida grafu kā *DOT* valodā rakstītu skriptu un spēj izkārtot to vairākos veidos, pateicoties pakotnē ietilpstošiem komandrindas lietojumiem:

- *Circo* – cirkulāra grafa izvietošana, piemērota grafiem ar vairākiem cikliem [9].
- *Neato* – atsperes modelis, balstoties uz virsotņu "*svara*" koeficientiem, mēģina samazināt globālas enerģijas funkciju [9].
- *Twopi* – radiāls izkārtojums, virsotnes tiek izvietotas uz koncentriskām riņķa līnijām, balstoties uz virsotņu attālumu no norādītas "*saknes*" virsotnes [9].
- *Dot* – hierarhiskais grafu izkārtojums priekš orientētiem grafiem, kas izvieto virsotnes pēc to šķautņu virzieniem, un tad mēģina reducēt šķautņu garumu un izvairīties no šķautņu krustošanās [9].

GVedit sniedz iespēju apskatīt izveidotu grafu tikai kā statisko bildi, nav nekādas iespējas interaktīvi pamainīt atsevišķa grafa elementa atribūtus. Visas izmaiņas tiek veiktas modificējot sākotnēju grafa skriptu. No tā var secināt, ka rīks neatbalsta lielāko daļu no augstāk izvirzītajām prasībām, un tā ir šī risinājuma lielākā problēma.

Neskatoties uz to, ka risinājumam ir absolūts interaktivitātes trūkums, tas atbalsta diezgan plašu elementu attēlošanas veidu klāstu. Norādot atbilstošus atribūtus grafa skriptā, var mainīt gan aizpildījuma krāsas virsotnēm un šķautnēm, gan apmales zīmēšanas stilu. Ir pieejami

vairāki šķautņu uzgaļi, lai attēlotu dažādākus saistību tipus. Papildus *GVedit* prot vizuāli izdalīt apakšgrafus, apvelkot tajos ietilpstošas virsotnes ar taisnstūri.

Izveidoto grafu var saglabāt vairākos datņu formātos, vairākums no kuriem ir attēlu formāts. Toties no grafu aprakstošiem formātiem tiek atbalstīti tikai daži, starp tiem šā darba ietvaros noderīgi ir tikai divi: "*DOT*" valodā rakstīts skripts ar "*.gv*" vai "*.dot*" datņu paplašinājumu un vienkāršā tekstā sastādīts grafa apraksts, kur katra rinda apraksta kādu atsevišķu grafa elementu, ar atstarpi atdalot elementa atribūtu vērtības.

Pievēršot uzmanību attēlojamajai informācijai, var redzēt, ka tiek attēlotas tikai virsotņu iezīmes, gadījumā, ja tām ir piekārtoti vairāki datu parametri, tie netiks parādīti un priekš lietotāja nebūs pieejami. Papildus iezīme tiek izmantota vienlaicīgi, gan kā identifikators virsotnei, gan kā attēlojamās informācijas avots. Situācijā, kad informācijas apjoms ir lielāks par vienu vārdu, šķautņu apraksts, kuras savieno šo virsotni ar citām, kļūst grūti uztverams.

Kaut gan *GVedit* nesniedz nekādu funkcionaliti vairāku lietotāju vajadzību īstenošanai un nebūt nav no labākajiem risinājumiem, tas uzskatāmi demonstrē "*DOT*" valodas spējas aprakstīt grafa struktūru un atsevišķu elementu vizuālo noformējumu. Vēl vairāk, tas parāda, ka vizualizēšanas rīks var sadarboties ar jau esošām grafa izvietošanas lietotnēm, atsakoties no izkārtošanas algoritmu implementācijas, kas būtiski samazina šāda risinājuma prasības pret lietotāja darbstacijas resursiem.

3.4. Cytoscape

Sākotnēji, *Java* valodā rakstīts risinājums, tika domāts bioloģiskiem pētījumiem, bet attīstoties kļuva par vispārīgas informācijas attēlošanas rīku.

Tādēl, ka tas izmanto *Java virtuālo mašīnu,* tas ir pieejams vairākās datora operētājsistēmās. Vērts pieminēt, ka tiek izstrādāta *Cytoscape.js* bibliotēka, kas iekļauj lielāko daļu no risinājuma funkcionalitātes un domāta, kā atbilde tīmekļu tehnoloģijām, kas strauji attīstās. Pagaidām tā ir vēl nepabeigta, bet nākotnē tā var kalpot par pamatu vajadzīga rīka izstrādei.

Pateicoties iebūvētam "*NetworkAnalyzer*" rīkam, var veikt ļoti dziļu un detalizētu tīklu analīzi, aprēķinot katrai virsotnei vairākus koeficientus un parametrus, piemēram, virsotnes pakāpi vai virsotnes klasterizācijas koeficientu.

Rīks atbalsta būtiskākās interaktivitātes prasības: tālummaiņu, iepriekšējo darbību vēsturi, pārskata iespējas, datu saistību attēlošanu. Salīdzinājumā ar *GVedit* lietotājam ir daudz vairāk iespēju modificēt grafa attēlu, piemēram, *Cytoscape* piedāvā plašāku grafu izkārtojumu

klāstu, ietverot ne tikai jau minētos izkārtojumus, bet arī vairākus citus: režģa izkārtojums, organiskais izkārtojums, utt..

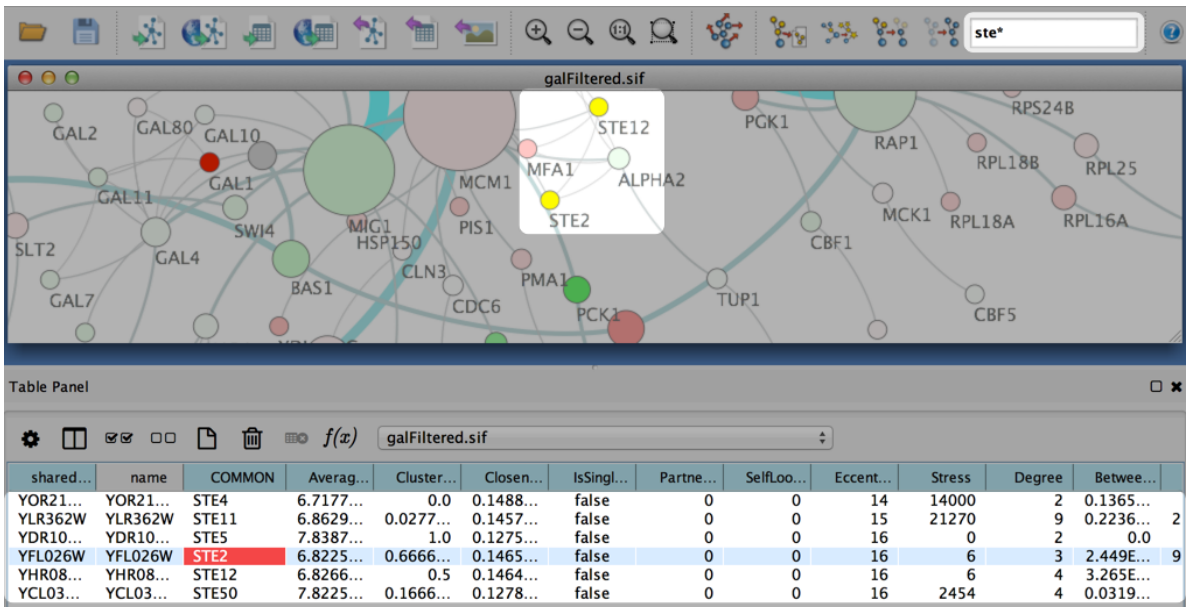
Katru virsotni un šķautni var patvaļīgi pārvietot vai izmainīt to vizuāla noformējuma atribūtus, ka arī abstraktos parametrus. Ir pieejami vairāki šķautņu tipi un lietotājs var mainīt gan loku līniju attēlojumu, gan šķautņu uzgaļu veidus. Izvēloties grafa elementu, tā atribūti tiek parādīti tam veltītā loga kreisajā pusē, kur var ērti norādīt ne tikai vajadzīgā atribūta vērtību konkrētam elementam, bet arī šī tipa elementiem globālo vērtību (*piem.: izvēloties virsotni var pamainīt ģeometrisko figūru tikai tai vai arī visām virsotnēm uzreiz*).

Vēl viena noderīga funkcionalitāte ir virsotņu grupēšana. Izvēlētās virsotnes var apvienot grupā, kurai jānorāda nosaukums. Divreiz uzklikšķinot uz jebkuras virsotnes, kas ietilpst grupā, paslēps visus grupas elementus, aizstājot tos ar vienu virsotni, kurai tiks attēlots grupas nosaukums. Uzklikšķinot divreiz uz "*grupas*" virsotnes, tās vietā atgriezīs iepriekš paslēptās virsotnes, saglabājot virsotņu iepriekšējo izvietojumu.

Loga apakšējā daļā grafa dati tiek attēloti kā trīs tabulas, sadalot tos šķautnēs, virsotnēs un grafos, pieņemot, ka grafā var eksistēt apakšgrafi *(Att. 3.1*). Katra tabulas kolona atbilst datu parametram, bet rinda – grafa elementam. Tabulai var pievienot jaunu parametru vai izmainīt jau esošos un gadījumā, ja tika pielietots "*NetworkAnalyzer*", tad tabulās virsotņu un šķautņu ierakstiem tiks izveidoti kolonas, kas glabās aprēķinātos koeficientus un parametrus.

Kaut arī tabulās var eksistēt vairāki elementa atribūti, uz attēla tiek parādīts tikai viens no tiem. Pēc noklusējuma tas ir elementa nosaukums, toties to var mainīt, izvēloties kādu citu no tabulās esošajiem elementu parametriem.

Izvēloties tabulā kādu ierakstu, tas tiek iezīmēts arī grafa attēlā, kas manāmi atvieglo vajadzīgā elementa meklēšanu. Papildus tam ir pieejams atsevišķs lauks, kurā var ierakstīt vajadzīgā elementa atribūta pilnu vai arī daļējo vērtību, izmantojot "*" simbolu. Elements tiks meklēts pēc ievadītas "*atslēgas*" starp tā atribūta vērtībām, kurš šobrīd ir izvēlēts attēlošanai grafā. *Attēlā 3.1* var redzēt, ka virsotnēm tika izvēlēts attēlot "*COMMON*" atribūtu un tiek meklētas visas virsotnes, kurām šī atribūta vērtība sākas ar "*ste*" virkni. Atbilstošas virsotnes tika iezīmētas attēlā ar dzelteno krāsu.



shared...	name	COMMON	Averag...	Cluster...	Closen...	IsSingl...	Partne...	SelfLoo...	Eccent...	Stress	Degree	Betwee...	
YOR21...	YOR21...	STE4	6.7177...	0.0	0.1488...	false	0	0	14	14000	2	0.1365...	
YLR362W	YLR362W	STE11	6.8629...	0.0277...	0.1457...	false	0	0	15	21270	9	0.2236...	2
YDR10...	YDR10...	STE5	7.8387...	1.0	0.1275...	false	0	0	16	0	2	0.0	
YFL026W	YFL026W	STE2	6.8225...	0.6666...	0.1465...	false	0	0	16	6	3	2.449E...	9
YHR08...	YHR08...	STE12	6.8266...	0.5	0.1464...	false	0	0	16	6	4	3.265E...	
YCL03...	YCL03...	STE50	7.8225...	0.1666...	0.1278...	false	0	0	16	2454	4	0.0319...	

3.1. att. **Cytoscape meklēšanas piemērs**

Gribētos atsevišķi atzīmēt *Cytoscape* filtrācijas funkcionalitāti, kas paradījās tikai lietotnes pēdējā versijā. Lietotājs var definēt savus patvaļīgus filtrus, atlasot virsotnes pēc trim kritēriju veidiem:

- tabulās esošām vērtībām.
- elementu topoloģiskiem parametriem (*piem.: kaimiņu skaits noteiktā attālumā*).
- virsotņu pakāpēm.

Šos veidus var apvienot ar loģiskam operācijām "*UN*" un "*VAI*" , kas ļauj sastādīt ļoti sarežģītus un specifiskus filtrācijas pieprasījumus. Atlasot kādus grafa elementus, tos var paslēpt kopējā grafā, pārnest atsevišķā logā kā jaunu grafu vai arī apvienot apakšgrafā un veikt izmaiņas, kas neietekmēs vecāko grafu (*piem.: izvēlēties citādāku apakšgrafu izkārtojumu*).

Datu pārnesamības ziņā šis risinājums piedāvā vairākas iespējas, nodrošinot importā atbalstu *GML*, *XGMML*, *GraphML* valodām, kā arī spēj ielasīt tabulām domātus datņu paplašinājumus, savukārt eksportēšanai, no visām grafu aprakstošām iespējam, piedāvā tikai *XGMML* vai *Cytoscape.js* bibliotēkai domātu *JSON* formātu.

Pastāv arī daži trūkumi, piemēram, apakšgrafi netiek attēloti kopējā bildē, virsotne tiek atzīmēta ar speciālo simbolu, norādot, ka tā satur apakšgrafu. Atverot apakšgrafu, tas tiek parādīts atsevišķā logā. Lietotājam noderētu apakšgrafa vai grupas attēlojums kopā ar vecāko grafu, lai rastu priekšstatu par datu globālo bildi, bet tā nav tik būtiska problēma. No izvirzītajām prasībām, rīks neatbilst tikai divām: trūkst navigācijas funkcionalitātes un pieejas globālajā tīmeklī, bet ņemot vērā *JavaScript* izstrādē esošo versiju, šis risinājums izskatās daudzsološi.

Kopumā *Cytoscape* ir ļoti labs piemērs, kā jāveido datu vizualizācijas rīku, jo īpaši ja uzmanība tiek koncentrēta tieši uz lietotāja pieredzes aspektu. Autors cer, ka tīmeklim tiks izstrādāts tikpat spēcīgs un ērts risinājums.

3.5. Gephi

Visjaunākais starp visiem apskatītajiem risinājumiem, kas vēl atrodas beta testēšanas stadijā, tāpēc iespēju ziņā nav tik bagāts kā *Cytoscape*. Tāpat kā iepriekš apskatītais rīks, *Gephi* izmanto *JVM*, tāpēc ir diezgan prasīgs pret datora resursiem, jo īpaši ja iet runa par vecajām darbstacijām. Pagaidām ir pieejams tikai *Windows, Linux* un *MacOSX* operētājsistēmu lietotājiem. Par mobilo ierīču atbalstu nekas netiek minēts, tāpat kā par rīka versiju priekš globālā tīmekļa.

Datu importēšanai tiek atbalstīti daudzi formāti, no iepriekš apskatītām valodām ir pieejami *DOT, GML* un *GraphML*, savukārt eksportējot grafu, no tiem tiek piedāvāti tikai pēdējie divi, kas liedz pārnest datus, piemēram uz to pašu *GVedit*.

Veidojot jaunus grafus vai modificējot esošos, paliek skaidrs, kāpēc risinājums vēl joprojām nav sasniedzis savu pirmo versiju. Esošo darbību uzvedība var dažreiz rast lietotāju apjukumu, jo tā nav vienmēr paredzama. Piemēram, atsevišķas virsotnes un vairāku virsotņu izvēlei jāizmanto dažādus rīkus. Kaut arī tiek atbalstītas vairākas no šajā darbā apskatītām vajadzībām, rīki to apmierināšanai nav intuitīvi vai nav līdz galam pabeigti. Piemēram, pašreizējā versijā nevar mainīt atsevišķas virsotnes formu, bet tikai tās krāsu. Savukārt šķautnes vispār nevar izvēlēties un līdz ar to mainīt to vizuālo noformējumu. Ir iespējama tālummaiņa un grafa bīdīšana, bet tā darbojas lietotājam neierastā veidā.

Pētot risinājumu tika konstatēti vairāki būtiskākie trūkumi:

- Netiek atbalstīta lietotāju darbību vēsture - visas veiktas izmaiņas ir neatgriezeniskas.
- Nav navigācijas funkcionalitātes.
- Virsotņu meklēšana nav pieejama.

Toties jau izveidoto grafu pārlūkošana un analizēšana ir izstadāta krietni labāk. *Gephi* atbalsta sarežģītu filtru veidošanu, piedāvājot pielietot datiem vairākas statistiskas funkcijas. Papildus, ir pieejams datu skats tabulu veidā, kur dati tiek sadalīti šķautnēs un virsotnēs līdzīgi ka *Cytoscape*. Neērtības sagādā fakts, ka datu skats ir vienīga vieta, kur tiek uzskaitīti grafa elementi un tas ir pieejams tikai atsevišķā cilnē - apskatīt to vienlaicīgi ar grafa attēlu nevar! Grafam var mainīt virsotņu izkārtojumu, bet to vidū nav vis biežāk pielietoto, piemēram, hierarhiskā vai cirkulārā.

Neskatoties uz to, ka dažus *Gephi* trūkumus var atrisināt, instalējot tām paredzētās pievienojumprogrammas (*piem., tādas, kas dod iespēju pievienot jaunus izkārtojumus, u.c.*), autors nerekomendē izmantot programmu, jo pagaidām tai trūkst dažu pamatdarbību, kas mūsdienās ir kļuvušas par pašsaprotamām lietām.

3.6. yEd

Bezmaksas plašlietojuma diagrammu zīmēšanas rīks, kurā var veidot un attēlot arī grafus. Visi pārējie produkti ir komerciāli un bezmaksas pieejamas tikai novērtēšanas pakotnes.

Risinājuma saskarne ir pārdomāta un ērta lietošanā, tā sastāv no vairākiem palīglogiem, kas izvietoti kreisajā un labajā malās, bet centrā atrodas plakne kurā tiek veidots datu attēlojums. Tekošas pozīcijas noteikšanai var izmantot "*Pārskata*" (*angļ.val. - overview)* palīglogu, kas parāda ekrāna pozīciju relatīvi visai bildei, tas ir ļoti noderīgi, ja tiek pārlūkots liela apjoma grafs, un lietotājs pietuvināja kādu no elementiem. Zem pārskata loga var redzēt dažāda veida apakšgrafus atkarībā no izvēlētās virsotnes. Katrs no šiem apakšgrafiem parāda virsotnes saistības pēc kāda noteikta principa: kaimiņus, priekštečus un pēctečus (*tikai priekš orientētiem grafiem)* vai arī apakšgrafu, ko satur pati virsotne.

Navigāciju nodrošina "*Struktūras skata*" (*angļ.val. - Structure view*) logs, kurā tiek parādīts saraksts ar visām grafa virsotnēm. Uzklikšķinot uz virsotnes nosaukuma, ekrāns tiks pārvietots, lai parādītu atbilstošas virsotnes atrašanas vietu attēlojumā. Pēc noklusējuma saraksts tiek veidots balstoties pēc virsotņu nosaukumiem, bet to var izveidot arī pēc apraksta vai pēc hipersaitēm. Gadījumā, ja virsotnei nav norādīta hipersaite vai apraksts, to var izdarīt tajā pašā logā. Cits veids kā to izdarīt, ir izvēlēties atbilstošu virsotni uz attēlojuma un ar labās pogas klikšķi izsaukt kontekstizvēlni, kurā jāizvēlas "*īpašības*". Atvērsies logs ar vairākām cilnēm, viena no kurām sauksies "*data".* Tajā var norādīt gan virsotnes aprakstu, gan hipersaiti, uz kuru jānorāda virsotnei, diemžēl tie ir vienīgie datu parametri, ko atbalsta *yEd*.

Meklēt virsotni pēc parametra vērtības, var izvēlēties tajā pašā palīglogā atbilstošu parametru, bet papildus norādot speciāli tam domātajā laukumā, meklēšanas kritēriju. Saraksts tiks sastādīts tikai no tām virsotnēm, kas atbilst ievadītajam nosacījumam.

Jaunas virsotnes veidošanu atvieglo labajā pusē esošais "*paletes*" logs, kurā tiek parādīts iespējamo figūru klāsts. Pievienot jaunu virsotni var vienkārši pārvelkot izvēlēto figūru uz ekrāna centrālo daļu, bet ja figūrai nav nozīmes, tad var divreiz uzklikšķināt vajadzīgajā vietā uz grafa attēlojuma un tur parādīsies kvadrāts. Grafa attēlojuma virsotnēm tiek radīts to nosaukums, lai to rediģētu, var uz tā divreiz uzklikšķināt, kas parādīs virsotnes tekstu kā ievadlauku, kurā varēs veikt nepieciešamas izmaiņas. Izvēlētai virsotnei ir iespējams mainīt

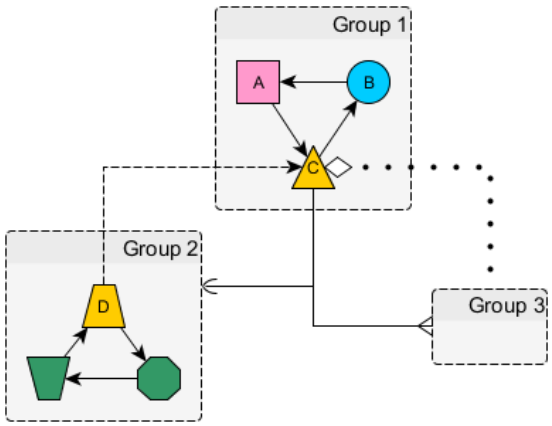
gandrīz jebkuru vizuālas noformēšanas atribūtu, pateicoties "*īpašību*" logam, kas atrodas labajā apakšējā stūrī. Starp īpašībām var atrast ne tikai elementa aizpildījuma krāsu vai apmales biezumu, bet arī novirzi relatīvi citiem attēla esošajiem elementiem vai arī iespējams virsotnē ievietot neliela izmēra bildes.

Nevienā no iepriekš apskatītiem rīkiem nebija tik viegli novilkt šķautni starp divām virsotnēm, ka *yEd* lietojumā. Lai savienotu divus elementus, pietiek nospiest peles labo pogu uz vienas virsotnes un neatlaižot to, novilkt līdz vēlamajam galamērķim. Brīdī, kad peles rādītājs atradīsies virs vēlamās virsotnes, risinājums piedāvās "*pieāķēt*" šķautni vienā no iepriekš definētiem punktiem uz virsotnes apmales. Pēc noklusējuma šķautnes ir orientētas, toties izvēloties šķautni , "*īpašību*" logā var izmainīt jebkuru no atribūtiem, tai skaitā arī šķautnes gala attēlošanas veidu (*bulta, aplis, utt*.).

Mainīt grafa izkārtojumu var gan bīdot atsevišķus elementus, gan pielietojot vienu no vairākiem piedāvātiem izvietošanas algoritmiem. Gribētos atzīmēt, ka tieši šī risinājuma algoritmu realizācijas tiek izmantotas vairākos datu vizualizēšanas rīkos, starp tiem ir arī *Cytoscape.*

Atšķirībā no *Gephi* tiek atbalstīta darbību vēsture, tāpēc lietotājam pieļaujot kļūdu, viņam nav jāsāk darbs no sākuma, jo visas izmaiņas var atcelt vai atgriezt. Toties filtrēt var ar daudz vienkāršākiem pieprasījumiem nekā *Cytoscape* vai *Gephi* atlasot rezultātu atsevišķā skatā, kā apakšgrafu.

Izvēloties vairākas virsotnes tās var apvienot grupā, kura vizuāli izskatīsies kā taisnstūris apkārt atbilstošam virsotnēm. Grupu ir iespējams minimizēt, tās vietā atstājot virsotni ar grupas nosaukumu iezīmes vietā. Ar šķautņu starpniecību vecāka grafa virsotnes var savienot ne tikai ar grupas elementu, bet arī ar tajā ietilpstošajiem elementiem (*att. 3.2*).



3.2. att. **yEd grupu piemērs**

Esošos grafu importēšanai un izveidoto grafu eksportēšanai var izmantot *GML* un *GraphML* formātus. Savukārt padalīties ar darbu caur globālo tīmekli pašlaik nevar, kaut gan

šīm rīkam eksistē vairākas tīmeklim domātas bibliotēkas, kas var kalpot kā sākums tīmekļa rīka izstrādei. Katra no bibliotēkām specializējas uz konkrētas tehnoloģijas grafikas vizualizēšanai. Starp tām ir arī risinājums kas saucas "*yFiles AJAX*", kas lietotāja pusē izmanto *SVG* vai *HTML5* atkarībā no lietotāja izvēles, bet visus aprēķinus veic servera pusē ar *Java* valodā rakstīto bibliotēku. Datu apmaiņa starp klienta pusi un serveri tiek nodrošināta ar *AJAX* tehnoloģijas palīdzību. Pats risinājums ir izstrādāts *Java* programmēšanas valodā un ir pieejams uz populārākajām datoru operētājsistēmām. Pašlaik tiek izstrādāta versija *Android* operētājsistēmai, kas nākotnē ļaus izmantot rīku arī daļai mobilo ierīču lietotājiem.

Apkopojot visus faktus, autors var droši pateikt, ka *yEd* atbilst lielākai daļai prasību, kas tika izvirzītas 3.2 apakšnodaļā, un dažu uzdevumu realizēšanā pārspēj visus citus risinājumus. Tomēr tam ir arī dažas nepilnības, piemēram, atšķirīgs virsotņu izkārtojums vienas grupas ietvaros netiek atbalstīts un grafa elementiem var būt tikai trīs stingri noteikti datu parametri, ar ko vairākos gadījumos nepietiek.

3.7. JavaScript bibliotēkas

Ideja, attēlot grafus izmantojot tīmekļa tehnoloģijas, ienāca prātāa jau daudziem cilvēkiem, kuri īstenoja to nevis kā rīku, bet gan bibliotēkas veidā. Tāpēc bez iepriekš minētajām bibliotēkām pastāv vēl dažas, kuras ir vērts pieminēt:

- *D3.js* – bibliotēka dinamiskai datu attēlošanai tīmekļu pārlūkos, kura prot veidot arī grafus. Grafu ielasa *JSON* formātā un piedāvā to izkārtot koka veidā (*ja tas ir iespējams),* hierarhiski, cirkulāri vai pielietot tam spēka vērstu izvietojumu. Vizualizācijai izmanto *HTML5* un *SVG* tehnoloģijas. Atbalsta teksta attēlošanu un atsevišķas virsotnes pārvietošanu.
- *Sigma.js* – atklāta pirmkoda bibliotēka, kas tika radīta priekš ātras un ērtas grafu attēlošanas tīmekļu vietnēs. Šobrīd atbalsta *JSON* un *GEXF* formātus priekš grafa importēšanas [10]. Pateicoties labai arhitektūrai, tā ir viegli paplašināma, kas jau ir pierādīts ar vairākām izstrādātām pievienojumprogrammām, piemēram, realizējot spēka vērstu izkārtojumu. Grafiskos elementus vizualizē ar *Canvas* un *WebGL* palīdzību.
- *KeyLines.js* – komerciāla bibliotēka, kas domāta tīmekļu rīku izstrādei, kura tiek pielietota liela apjoma datu attēlošanai vairākas nozarēs. Atbalsta populārākos tīmekļu pārlūkus un nodrošina korektu darbību arī mobilajās ierīcēs. Piedāvā

vairākus grafa izkārtojumus: planāro, hierarhisko, inkrementālo un radiālo. Satur optimizētas metodes ātrai datu filtrēšanai pēc vairāku parametru vērtībām, kā arī grafa analizēšanai, lai, piemēram, aprēķinātu klasterizācijas koeficientus vai noteiktu tīkla ietekmīgākas virsotnes.

Visas šis bibliotēkas var kalpot tikai kā atskaites punkts vajadzīga rīka izstrādei. Neskatoties uz to, ka tās paver iespējas veikt daļu no izvirzītajām prasībām, lai pilnībā īstenotu visu nepieciešamo funkcionalitāti, šīs bibliotēkas būs pamatīgi jāpapildina.

3.8. Secinājumi

Veicot esošo rīku izpēti tika konstatēts, ka neviens risinājums nespēj izvadīt uz grafa attēlojuma vairāk par vienu virsotnes atribūtu, vēl vairāk – tikai *Cytoscape* piedāvā opciju izvelēties, kuru no virsotnes datu atribūtiem rādīt. Savukārt, situācija, kad virsotnei jāattēlo piederība vairākām klasēm vai grupām, tiek atstāta lietotāja ziņā, piedāvājot problēmas risināšanai vizuālas noformēšanas rīkus un viņu pašu iztēli, kas pilnīgi nav pieņemams.

Runājot par dažāda tipa šķautņu pārklāšanas gadījumu, tika novērotas divas pieejas šķautņu pievienošanai virsotnei:

- Pievienot šķautni tuvākajā punktā uz virsotnes apmales;
- Pievienot šķautni tuvākajā pēc noklusējuma definētā punktā.

Katra no pieejām risina problēmu tikai specifiskās situācijās, atkarībā no virsotņu izvietošanas grafā, nesniedzot pieņemamus rezultātus problēmas novēršanai.

Kopumā vērtējot visus iepriekš minēto rīku stiprās un vājas puses, darba autors ar pārliecību var pateikt, ka ātrai nelielu grafu veidošanai un rediģēšanai, vislabāk izmantot *yEd* risinājumu, bet liela apjoma tīkliem un to detalizētai analizēšanai – *Cytoscape.*

4. Konceptuālais risinājums

Lēmums izmantot tīmekļa tehnoloģiju iespējas datu vizualizēšanai, tika pieņemts, balstoties uz faktu, ka šādai pieejai ir vairākas priekšrocības realizējot 3.2 apakšnodaļā izvirzītās prasības. Pirmkārt, tā atbrīvo lietotāju no nepieciešamības izmantot konkrētu operētājsistēmu, bet izvirza prasības pret lietotāja izmantojamajām tīmekļa pārlūkprogrammām. Kaut arī mūsdienās tīmekļu pārlūki pēc noklusējuma ir iekļauti vairākās populāro operētājsistēmu pakotnēs, to daudzveidība liek nodrošināt izplatītāko pārlūku atbalstu.

Svarīgi atzīmēt, ka pārnesot rīku no lokālas darbstacijas uz publiski pieejamo tīmekļa vietni, parādās iespēja daudz vieglāk realizēt vairāku lietotāju kopīgo darbu ar tiem pašiem datiem, kas var būtiski paātrināt un veicināt datu izpētes procesu.

Papildus – rīka izvietošana pasaules tīmeklī atvieglo datu apmaiņu starp vairākiem cilvēkiem. Būs iespējams dalīties ar sākotnēju informāciju vai vēl labāka perspektīva: ar darba gaitā sasniegtiem starprezultātiem. Nodrošinot publisku pieeju šiem datiem, tie var būt izmantoti izglītošanas un demonstrēšanas nolūkos. Tas rada nepieciesamību izveidot vairākas lietotāju lomas ar atbilstošām pieejas tiesībām, lai parūpētos par datu drošību.

Mazināt lietotāja datora noslodzi var sadalot risinājumu divos moduļos:

- Lietotāja modulis - atbildīgs par attēlošanu un lietotāja darbību apstrādi.
- Servera modulis - veic visus sarežģītus aprēķinus (grafa izkārtošana, analīzes veikšana, ceļu meklēšana, importējamo datu parsēšana) , nodrošina datu glabāšanu un izdošanu (izveidotus grafus, lietotāju kontus, piekļuves sesijas).

Izmantojot šādu arhitektūru, katrs no moduļiem būs neatkarīgs un vienas puses, pilnveidošana neietekmēs citu. Lai tas būtu iespējams, jāparūpējas par korektu sazināšanos starp moduļiem un apakšmoduļiem. Šīm mērķim tiks izveidots vienots datu formāts un, sekojot pieņemtām tīmekļa lietotņu izstrādē labām praksēm, par pamatu tika izvēlēts *JSON* formāts. Balstoties uz 2. nodaļā apskatīto valodu struktūrām, risinājuma iekšējam formātam obligāti jāsatur informācija par grafa elementu grafiskam īpašībām, kā arī datu objektus ar visiem vajadzīgajiem parametriem, bet vēl tam jāspēj aprakstīt katra objekta piederību kādai no lietotāja definētām klasēm un atsevišķas klases vizuālo noformējumu.

Servera modulis nodrošinās būtiskākos grafa izkārtojumus, izmantojot jau esošos atklātā pieejā esošos risinājumus un bibliotēkas, piemēram, jau apskatīto *Gephi*. Savukārt datu glabāšanai tiks izveidota datu bāze, kurā grafs un tā elementi tiks atveidoti kā atsevišķas entitātes, bet elementu savstarpējās saistības, kā entitāšu sakari.

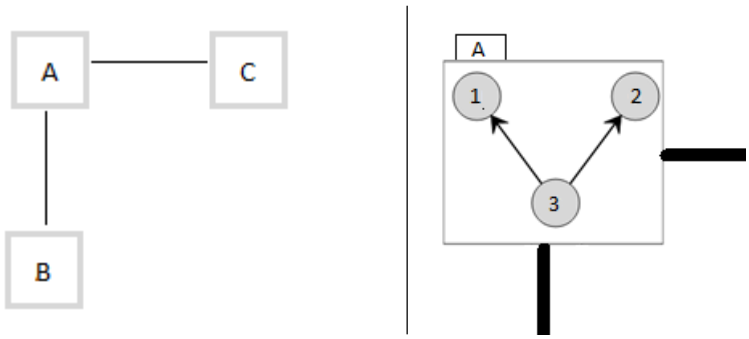
Lai būtu vieglāk realizēt lietotāju darbību vēsturi un veicināt saskarnes interaktivitāti, darba autors nolēma veidot lietotāja moduli izmantojot uz notikumiem balstītu arhitektūru. Tas ļaus sadalīt funkcionalitāti vairākos apakšmoduļos, ierobežojot to ietekmi lietotāja puses ietvaros un ļaus apakšmoduļiem darboties paralēli, kas palielinās risinājuma ātrumu.

Grafikas attēlošanai autors piedāvā izmantot *SVG* formātu uz *XML* bāzētās dokumenta struktūras, kas manāmi ietaupīs laiku katru reizi, kad būs nepieciešama piekļūve pie jau esošajiem attēlā elementiem. Tas tiks nodrošināts pateicoties tieši šim nolūkam paredzētam "*id*" atribūtam, kas ir obligāts katram šī formāta elementam un satur unikālo identifikatoru. Papildus katram *SVG* elementam ir personīga matrica, kurā koeficientu veidā tiek glabātas visas elementam veiktas izmaiņas (*bīdīšana, rotēšana, izmēru izmaiņas*). No tā var secināt, ka objektu pārvietošanas vai tālummaiņas realizacija nesagādās nekādas problēmas, jo īpaši tā fakta dēl, ka *SVG* ir vektor grafika, attēls nezaudēs kvalitāti, ja tiks mainīts fokusa attālums.

Ievērojot iegūtos rezultātus un pieredzi testējot esošos rīkus, autors uzskata, ka vismaz ir jārealizē sekojoša funkcionalitāte vizuālai noformēšanai jebkuram grafa elementam: apmales platums, krāsa un zīmēšanas veids, aizpildījuma un tekstuālas informācijas krāsa. Šķautnēm ir jābūt dažāda tipa galu attēlojumiem (*bultiņas, aplīši, četrstūri, utt.)* un līniju veidiem (*locītas, lauztas, taisnas, utt.)* , lai rīks atbalstītu dažāda veida saistību attēlošanu starp objektiem.

Datu objektu glabāšanai lietotāja pusē jāizmanto datu struktūra, kas sniegs ātrāku piekļuvi konkrētam objektam. Pēc autora domām var izmantot "*vārdnīcas*" struktūru, kurā katram objektam ir unikāla *atslēga* un, zinot to, var piekļūt objektam pa "*tiešo*" - bez meklēšanas algoritmu palīdzības. Optimizācijas nolūkos kā *atslēga* tiks izmantots atbilstoša *SVG* elementa "*id*" atribūta identifikators. Šī ideja krietni atvieglos datu objekta noteikšanu apstrādājot lietotāja darbības, piemēram, virsotnes izvēle uz attēlojuma. Nepieciešams vēl nedaudz uzlabot veiktspēju, sadalot visus objektus pēc grafa elementu tipiem – izveidojot atsevišķas datu struktūras grupu, šķautņu, virsotņu un apakšgrafu glabāšanai.

Virsotņu grupu atbalstu var nodrošināt glabājot to ligzdstruktūrā kā atsevišķu grafu. Vizuāli tā būs attēlota kā normāls grafs, kurš atrodas vecāka grafa virsotnē, vienkārši tā izskatīsies kā vairākas reizes samazināts grafs (*att. 4.1*).



4.1. att. **Grupas realizācijas piemērs**

Katrai virsotnei, šķautnei un virsotņu grupai tiks piekārtota iezīme, kas teksta veidā attēlos vienu atbilstoša datu objekta parametru, pēc noklusējuma tas būs elementa nosaukums. Attēlojamāas informācijas kontrolei, risinājums saturēs "*DataViewer*" apakšmoduli, kurš atbildēs par datu objektu parametru glabāšanu un pārvaldīšanu. Tas tiks pārstāvēts lietotnes saskarnē kā atsevišķs palīglogs, kurā būs iespējams norādīt attēloto iezīmju skaitu un katrai no iezīmēm būs pieejams saraksts ar visu datu objektu atlasīto parametru nosaukumiem. Izmantojot šo sarakstu, lietotājs spēs izvēlēties, kurus no parametriem, rādīt tikai izvēlētiem vai visiem elementiem grafā.

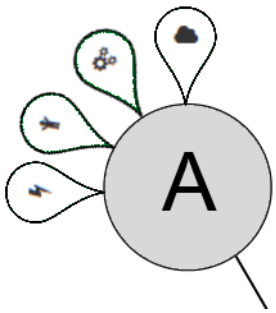
Meklēšanai lietotājam tiks piedāvāts atsevišķā ievadlaukā norādīt kritēriju un blakus būs jāizvēlas, pēc kura datu objekta parametra jāmeklē. Visi elementi, kas atbilst kritērijam tiks vizuāli izdalīti, pazeminot attēlošanas blīvumu visiem pārējiem elementiem. Filtrācijai, tiks piedāvāta līdzīga funkcionalitāte ar vienīgo atšķirību, ka lietotājam būs iespēja norādīt vairākus meklēšanas kritērijus un rezultāts tiks parādīts atsevišķā cilnē kā jauns grafs.

Pilnvērtīgai grafa pārlūkošanai tiks nodrošināts navigācijas atbalsts ar "*mini-kartes*" palīdzību, kas atradīsies atsevišķā palīglogā un atspoguļos tekošo ekrāna pozīciju un fokusa attālumu atbilstoši visai bildei. Papildus, lai lietotājs varētu pārvietoties pa grafu, tiek piedāvāts realizēt dubulto klikšķi uz grafa šķautnēm, kas novirzīs ekrānu tā, lai centrā parādītos tālākā no divām virsotnēm, kas ir saistīta ar šo šķautni.

Balstoties uz 3. nodaļā aprakstīto analīzi, lai nodrošinātu sadarbību ar esošiem risinājumiem, rīkam jāprot importēt un eksportēt grafus *GML, GraphML* un *DOT* formātos. Ielasot jaunus datnes šajos formātos, dati tiks konvertēti iepriekš minētā *savējā* formātā, lai pēc tam tos varētu izmantot visās lietotnes sastāvdaļās. Līdz ar to rodas vēl viens rīka pielietojums – to varēs izmantot formātu konvertēšanai. Piemēram, tas ļaus pārnest grafus starp *Cytoscape* un *GVedit* lietotnēm.

Protams, vissvarīgākā konceptuālā risinājuma īpašība ir 1. nodaļā aprakstīto problēmu atrisināšana. Pirmkārt, lai pārvaldītu lietotāju izveidoto objektu klases, autors piedāvā izmantot atsevišķu apakšmoduli, kas sniegs iespēju atsevišķā logā nodefinēt katras klases attēlošanas

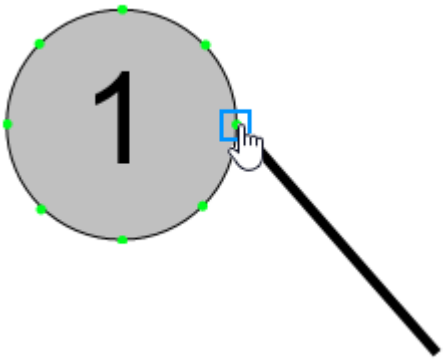
stilu, izmantojot aizpildījuma krāsas, apmales veidu un platumu, elementa formu. Tā ir standarta pieeja vienas piederības attēlošanai, bet lai atrisinātu problēmu ar elementa piederību vairākām klasēm, tiek ieviests vizuālais identifikators. Katrai klasei būs iespējams izvēlēties simbolu no esošajiem vai arī augšupielādēt lietotāja izvēlēto, kas grafā simbolizēs elementa piederību noteiktai klasei. Papildus, grafu elementiem tiks piešķirts "*galvenās klases*" atribūts, kas norādīs elementa augstākas prioritātes piederību. Grafā virsotnei tiks pielietots viņas *galvenās* klases stils, toties visas parējās klases tiks attēlotas ar atsevišķām grafiskām paskaidrēm (*angļ.val. - tooltip)* apkārt virsotnei (att. 4.2), kuru attēlošanu lietotājs varēs atslēgt. Novietojot peles radītāju virs vienas no paskaidrēm, tiks parādīts klases nosaukums un paskaidrojums, ja tie tika definēti.



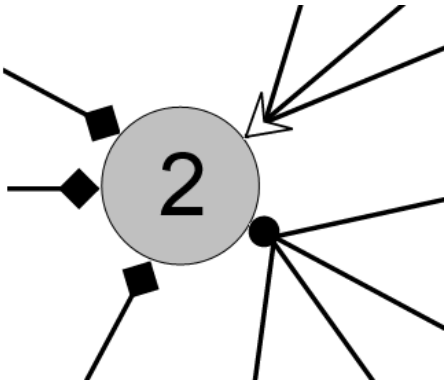
4.2. att. **Virsotne ar četrām klasēm.**

Virsotnei pievienoto šķautņu pārklājumu problēmas risināanai, autors nolēma piedāvāt lietotājam izvēlēties kādā veidā tiek noteikts savienošanas punkts tikai izvēlētai virsotnei vai visām grafā attēlotajām. Visvienkāršākais variants – aprēķināt tuvāko punktu uz virsotnes apmales. Otrais – katrai figūrai, kas ir pieejama rīkā virsotnes attēlošanai, tiks aprēķināti noteikti punkti uz tās perimetra un lietotājs pats varēs izvēlēties, pie kura no tiem pievienot šķautni (att. 4.2). Abi varianti ir plaši izmantojami un otrais nedaudz sakārto šķautnes apkārt konkrētai virsotnei, bet neatrisina problēmu ar vairākiem šķautnes galu attēlojumiem. Tāpēc autors izdomāja trešo variantu – piedāvāt lietotājam sagrupēt visas šķautnes apkārt virsotnei pēc attiecīga šķautnes gala tipa (att 4.3). Izvēloties virsotnei šādu šķautņu attēlošanas veidu, risinājums atstās tikai pa vienam pārstāvim no visiem šķautņu uzgaļu veidiem un aprēķinās tam optimālāko atrašanas punktu uz virsotnes perimetra, balstoties uz šķautņu otra gala pozīcijām. Protams, var būt gadījumi, kad atsevišķai šķautnei kopējais savienošanas punkts nebūs tuvākais, tāpēc, lai uzlabotu situāciju, lietotājam būs pieejamas dažas papilddarbības. Pirmkārt, pastāvīgi noteikt uzgaļa pozīciju, pārvietojot izvēlēto uzgali pa virsotnes perimetru. Otrkārt, peles kreisās pogas dubultklikšķis virs uzgaļa, sagrupēs visas šķautnes apkārt noteiktai virsotnei, kurām ir tāda paša tipa uzgalis. Atkārtojot šo darbību tam pašam uzgalim vēlreiz,

attēlos šī uzgaļa tipa šķautnes optimālākajos punktos uz virsotnes perimetra. Vairāku uzgaļu tipu gadījumā, šī funkcionalitāte ļauj norādīt, kura tipa šķautnes vajag grupēt.



4.3. att. **Noklusētie savienojuma punkti .**



4.4. att. **Šķautņu grupēšana pēc uzgaļa tipa**

5. Risinājuma realizācijas apraksts

Šajā nodaļā tiks aprakstīts prototipa izstrādes process, izmantotās tehnoloģijas un prototipa darbības pamatprincips.

5.1. Izmantotās tehnoloģijas

Prototipa saskarne tika izstrādāta ar *JavaScript* (*JS*) programmēšanas valodas palīdzību, izmantojot *jQuery* bibliotēkas 1.11 versiju, lai atvieglotu piekļuvi lietotnes dokumenta elementiem. *SVG* grafikas elementu ģenerēšanai no *JS* koda tika izmantota *Raphael.js* bibliotēka, kas sniedz daudz ērtākas metodes gan jaunu figūru zīmēšanai, gan esošo modificēšanai. Servera moduļa funkcionālā daļa galvenokārt tika realizēta *PHP* skriptu valodā, bet datu glabāšanai tika izmantota *PostgreSQL* datubāze. Laika taupīšanas nolūkos grafu izkārtošanai tika izmantotas *GraphViz* pakotnē esošās komandrindas lietotnes: *dot, circo* un *neato*. Prototipa mitināšanai globalajā tīmeklī autors izvēlējās *Apache2* tīmekļa servera lietotni, kas tika ieinstalēta datorā ar *Debian* operētājsistēmu. Tiek atbalstītas izplatītāko tīmekļa pārlūku versijas: *Chrome* sākot ar 25 versiju, *Firefox* sākot ar 18 versiju.

5.2. Arhitektūra un darbība

Kā jau tika piedāvāts 4. Nodaļā prototips ir realizēts divos moduļos – lietotāju modulis un servera modulis. Darba gaitā pavisam tika izstrādātas lietotāja moduļa trīs versijas, pirmā no tām tika domāta, lai izveidotu pamatmetodes *SVG* grafikas veidošanai un izmēģināt izvēlētās bibliotēkas funkcionalitāti. Otrā versija pēc būtības bija lietotnes nesaistes versija, tajā tika realizēti pamatrīki grafa elementu zīmēšanai un to vizuālā izskata noformēšanai. Pielietojot iegūto pieredzi veidojot pirmos divus prototipus, tika izveidota prototipa trešā versija, paralēli tika izveidots servera modulis un nodrošināta saziņa starp abām risinājuma daļām. Būtiska šīs versijas atšķirība ir pilnīgi pārtaisīta arhitektūra, kura tika izveidota cenšoties pieturēties divām objektorientētas programmēšanas koncepcijām: klasēm un pārmantošanai, kurus *JavaScript* valoda neatbalsta. Tas tika panākts pateicoties funkciju prototipēšanas pieejai, kas simulē un aizvieto abas koncepcijas. Izstrādes laikā tika izveidoti divi palīgrīki, viens no tiem sniedz iespēju norādīt grafu *DOT* formātā un saglabāt datubāzē, otrais - ļauj lietotājam izvēlēties saglabātā grafa virsotni un saistīto elementu attālumu, pēc šiem paramatriem izkārtojot atbilstošu apakšgrafu un izvadot to uz ekrāna *DOT* valodā.

Lietotāja modulis sastāv no vairākām klasēm, sadalot funkcionalitāti pa apakšmoduļiem, kas apstrādā notikumus un atbilstoši reaģē uz tiem. Moduļa centrā atrodas *GraphCore* klase, kas atbild par lietotāja darbību reģistrēšanu (*piem.: peles taustiņa klikšķis*), apakšmoduļu inicializāciju lietotnes palaišanas brīdī, tekošā grafa abstraktās daļas glabāšanu. Par vizualizāciju atbild *GraphCanvas* apakšmodulis, kas izveido plakni ar *Raphael.js* bibliotēkas palīdzību, nodrošina ekrāna pārvietošanos un fokusa maiņu. Papildus šī klase glabā divus objektus, kas satur norādes uz virsotņu un šķautņu grafiskajiem objektiem. Svarīgi atzīmēt, ka pateicoties *JavaScript* īpatnībai, objekta atribūti tiek glabāti kā asociatīvais masīvs. Pateicoties tam, katru elementu var glabāt kā objekta atribūtu, kuram nosaukuma vietā tiek izmantots atbilstoša *SVG elementa* identifikators, tādā veidā nodrošinot tiešo piekļuvi.

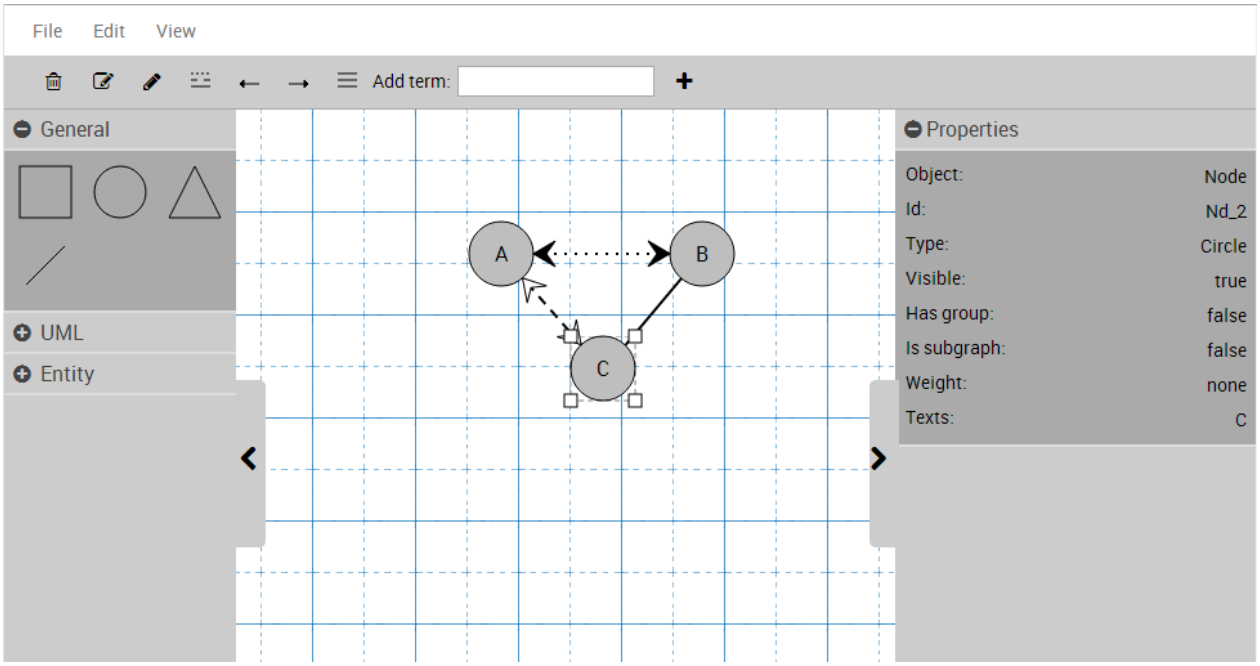
Klasēm, kas domāti grafiskajiem objektiem, pamatā ir *Shape* klase, kas realizē visām figūrām kopīgas metodes (*aizpildījuma krāsas maiņu, pārvietošanos, izmēru maiņu, utt.).* Uz šīs klases pamata tiek izveidotas

- *BaseNodeShape* klase – no kuras tiek mantotas visas virsotņu grafiskās klases. Tajā implementē virsotnēm kopīgas metodes un izveido tām specifiskos atribūtus.
- *BaseRelationShape klase* – abstraktas šķautnes aprakstīšanai ietver kopīgas metodes un atribūtus šķautnes attēlošanai un uzvedībai.
- *Handler* – apraksta grafisko elementu, kurš tiek izmantots, lai mainītu izvēlētās figūras izmērus vai šķautnes virzienu.
- *RelationHead* – atbild par šķautnes uzgaļu veidošanu un pārvaldīšanu.

Konkrētai figūrai, ar kuru tiks attēlota virsotne, tiek izveidota atsevišķa klase, kas tiek mantota no *BaseNodeShape* klases, bet specifisko šķautņu klases tiek pārmantotas no *BaseRelationShape* klases.

Grafa datu struktūru apraksta *Graph* klase, kas satur divus objektus virsotnēm un saistībām, tieši tāpat kā *GraphCanvas* klasē, katrs šo objektu atribūts ir virsotne vai šķautne. Atribūtu nosaukumi ir unikāli visā lietotnē un atbilst grafiskā elementa identifikatoram, nodrošinot vieglu sakaru starp grafa vizuālo un informatīvo daļu.

Visas saskarnes izvēlnes pārvalda *MenuManager* apakšmodulis, kas inicializē izvēlnes un rīkjoslas. Tas sastāv no *Toolbar* klases, augšējā daļā esošajām rīkjoslām (*att.5.1)* un *PropertyWindow* klases, kas atbild par izvēlētas virsotnes parametru attēlošanu prototipa labajā pusē (*att. 5.1*). Kreisajā pusē atrodas palete ar zīmēšanai pieejamām ģeometriskām figūrām (*att. 5.1).*



Att. 5.1 **Prototipa saskarnes ekrānuzņēmums.**

Prototipa tekošā versija piedāvā lietotājam veidot grafus izmantojot četru veidu virsotnes un taisnu līniju šķautnes, kā arī 8 šķautņu galu attēlojumus. Apmalēm un šķautņu līnijām var mainīt zīmēšanas veidu: punktētā, pārtrauktā, utt.. Visām virsotnēm un šķautnēm ir iespējams mainīt aizpildījumu krāsu, līniju platumu un krāsu. Katrai virsotnei ir piekārtota iezīme tekstuālo datu attēlošanai, kuras saturu var rediģēt divreiz uzklikšķinot uz tās ar peles kreiso pogu.

Izveidotus grafus ir iespējams eksportēt un arī importēt esošos grafus *JSON* balstītā formātā. Tas atdala grafa datus un vizuālās detaļas, divos asociatīvos masīvos: *graph* un *visuals*, formāta piemēru var apskatīt 2. pielikumā.

Testēšanas nolūkos, tika veltīta uzmanība programmatūras testēšanas glosārija [4] iekļaušanai. Izmantojot vienu no palīgrīkiem datubāzē tika saglabāts jēdzienu tīkls, kas bija iegūts no glosārīja. Prototips piedāvā izvēlēties no saraksta vienu terminu un apskatīt tam atbilstošu apakšgrafu, kura izmēri pēc noklusējuma ir ierobežoti divu izejošo un divu ienākošo saistību garumā. Izvēloties vārdu no saraksta, var norādīt grafa izkārtojumu - hierarhisko vai cirkulāro. Piemēru var redzēt 3.pielikumā.

Turot nospiestu "*ctrl*" taustiņu un klikšķinot peles kreiso pogu, kad peles radītājs atrodas tieši virs virsotnes, prototipam tiek dota komanda attēlot atbilstoša termina apakšgrafu. Gadījumā, ja lietotājs gribēs pievienot tekošajam grafam kādu terminu no jēdzienu tīkla, tad prototipa augšējā daļā, pie rīkjoslas, šīm nolūkam ievietots speciāls ievadlauks, kurā var norādīt

terminu. Ja attēlojumā tāda termina nav, tad tas tiks pievienots, bet ja ir - prototips novirzīs ekrānu tā, lai parādītu attiecīgās virsotnes atrašanas vietu.

Servera pusē glosārijs tiek glabāts datubāzē ar vairāku tabulu palīdzību. Terminiem tiek izmantota glabāta *glossary* tabula, kura satur to nosaukumu, aprakstu un valodu. Saistībam tika izveidota tabula *Relations*, kurā tiek glabāts saistības veids un virziens, kā arī savienotu terminu identifikātori. Virsotņu un šķautņu attēlojuma atribūtus glabā *term_attributes* un *relations_attributes* tabulās.

Momentā, kad lietotājs izvēlējās no saraksta kādu terminu un izkārtošanas veidu, serveris ar *AJAX* starpniecību saņem pieprasījumu un izpilda *graph_generator* skriptu, kas apstrādā pieprasījumus un sāk ģenerēt apakšgrafu šim terminam. Vispirms datubāzē tiek atrasti visi saistītie termini noteiktā attālumā no vajadzīga termina, pēc tam tiek meklētas visu iepriekš atrasto terminu savstarpējās saistības. Kad apakšgrafa abstrakta daļa ir izveidota, no datubāzes tiek atrasta katra termina vizuāla noformēšana un tiek veidota datne *DOT* formātā. Atkarībā no izvēlēta izkārtojuma, skripts darbina no komandrindas *neato* vai *circo* lietotnes. Saņemot no izkārtošanas lietotnes datni ar grafu elementu koordinātēm tekstuālajā veidā, dati tiek apstrādāti un pārveidoti prototipa iekšējā *JSON* formātā. Pievienojot šiem datiem, virsotnēm piekārtoto informāciju, tie tiek aizsūtīti lietotāja modulim attēlošanai.

5.3. Tālākā attīstība

Gribētos atzīmēt, ka tas ir tikai prototips, kurš nebūt nav pabeigts. Turpmāk tas tiks pilnveidots un tiks izstrādātas moduļu jaunas versijas. Protams, vispirms pakāpeniski tiks ieviestas pārejās konceptuālajā risinājumā minētās idejas, bet pēc tam rīku tik un tā var uzlabot un šajā apakšnodaļā autors grib aprakstīt rīka turpmākās attīstības iespējas.

Veidojot glosārija atbalstu, autoram nācās saskarties ar terminu ontoloģijām un lietotnēm, kas strādā ar tām. Tika noskaidrots kā tās tiek aprakstītas diezgan populārā *OWL* valodā, tāpēc viens no uzlabojumiem būtu šīs valodas atbalsts. Tas sniegtu iespēju importēt ontoloģijas bez grafu aprakstošas valodas starpniecības. Attīstot šo ideju, pakāpeniski var ieviest citām nozarēm specifiskus formātus, paplašinot rīka pielietojuma jomu.

Strādājot grafiem autoram bieži vien nācās atrast likumsakarības starp virsotnēm un aprēķināt dažādus koeficientus, tāpēc viena no idejām ir piedāvāt lietotājiem tīkla analizēšanas rīkus un sastādīt atbilstošas atskaites.

Būtiskais uzlabojums būtu palielināt piedāvāto izkārtojumu klāstu, piedāvājot, piemēram, spēka orientētu izkārtojumu vai koka veida izkārtojumu.

Vienā no darba nodaļām tika teikts, ka tīmeklī izvietotais rīks spēs nodrošināt dalīšanos ar starprezultātiem, tāpēc autors uzskata, ka nepieciešams ierobežot piekļuvi risinājuma privātai daļai, ieviešot lietotāju kontus un autentifikāciju. Piemēram, tieši glosārija pārvaldīšanai varētu ieviest lietotāju lomas, atļaujot tikai noteiktai cilvēku grupai papildināt un saglabāt oriģinālu, bet visiem pārējiem piedāvāt veidot savas glosārija versijas, ņemot oriģinālu par pamatu.

Pēc autora domām visinteresantāka šī rīka attīstība būtu pāriešana no divdimensiju uz trīsdimensiju grafiku, jo īpaši zinot, ka no tehnoloģiskās puses tas ir iespējams izmantojot *WebGL* tehnoloģiju [11]. Ņemot vērā, ka tai ir *JavaScript* rakstītas bibliotēkas [12] un pateicoties rīka modularitātei, šāda modernizācija ir ļoti iespējama.

Rezultāti un secinājumi

Darba gaitā tika veikta esošo grafu un datu tīklu vizualzēšanas lietojumprogrammatūru analīze, noskaidrojot nozares labās prakses saskarnes izstrādē, liela apjoma datu attēlošanas optimizācijas, kā arī tika noteiktas ar datu attēlošanu saistītās problēmas un pastāvošo rīku mēģinājumi tos atrisināt.

Papildus tika apskatītas izplatītākās valodas grafu aprakstīšanai, ar kuru palīdzību esošās lietotnes var nodrošināt datu pārnesamību, novērtējot to struktūru iespējas turpmākai paplašināšanai, lai iekļautu situācijām specifiskos datus.

Darba praktiskās daļas ietvaros tika piedāvāts savs problēmu konceptuālais risinājums un izstrādāts šī risinājuma daļējais prototips. Autors ieguva vērtīgu pieredzi informācijas attēlošanā ar tīmekļa tehnoloģiju palīdzību un nonāca pie secinājuma, ka notikumu bāzēta arhitektūra šāda veida tīmekļa lietotņu veidošanā ir optimālākā pieeja, padarot izstrādes procesu vieglāku, datu attēlojumu uzskatāmāku, bet lietotnes saskarni ērtāku izmantošanā.

Prototips tiks turpmāk pilnveidots, lai realizētu visas risinājuma aprakstā minētās idejas un izstrādātu jaunas tehnoloģijas specifikas optimizācijas.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] **D. Boyd and N. B. Ellison**, *Social network sites: Definition, history, and scholarship*. [tiešsaiste]. Pieejams: http://onlinelibrary.wiley.com/doi/10.1111/j.1083-6101.2007.00393.x/full

[2] **B.Shneiderman**, *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations.* [tiešsaiste]. Pieejams: https://www.cs.umd.edu/users/ben/papers/Shneiderman1996eyes.pdf

[3] **A.Rilins**, *Grafu pārlūkošana tīmekļa pārlūkā. (kursa darbs)*

[4] *Standard glossary of terms used in Software Testing*, [tiešsaiste]. Pieejams: http://science.df.lu.lv/kaab13/istqb_glossary_of_testing_terms_2.2.pdf

[5] **E.R. Gansner,** *A Technique for Drawing Directed Graphs* [tiešsaiste]. Pieejams: *http://www.graphviz.org/Documentation/TSE93.pdf*

[6] **Michael Jünger, Petra Mutzel,** *Graph Drawing Software.*

[7] *Graph Modelling Language.* [tiešsaiste]. Pieejams: http://en.wikipedia.org/wiki/Graph_Modelling_Language

[8] **Dr. Margaret Bernard,** *Graph file formats,* [tiešsaiste]. Pieejams: http://www2.sta.uwi.edu/~mbernard/research_files/fileformats.pdf

[9] *GraphViz* . [tiešsaiste]. Pieejams: http://edutechwiki.unige.ch/en/Graphviz

[10] *Sigma.js new - The amazing new online graph visualization tool,* [tiešsaiste]. Pieejams: http://noduslabs.com/radar/sigma-js-online-graph-visualization-tool/

[11] **G. Tavares,** *WebGL fundamentals.* [tiešsaiste]. Pieejams: http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/

[12] **J. M. Moore,** *WebGL for Scientific Visualization.* [tiešsaiste]. Pieejams: http://jaredmmoore.com/webgl-for-scientific-visualization/

PIELIKUMI

1. GraphML valodas piemērs

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
 http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
    <key id="d0" for="node" attr.name="color" attr.type="string">
        <default>yellow</default>
    </key>
    <key id="d1" for="edge" attr.name="weight" attr.type="double"/>
    <graph id="G" edgedefault="undirected">
        <node id="n0">
            <data key="d0">green</data>
        </node>
        <node id="n1"/>
        <node id="n2">
            <data key="d0">blue</data>
        </node>
        <node id="n3">
            <data key="d0">red</data>
        </node>
        <node id="n4"/>  <node id="n5">
            <data key="d0">turquoise</data>
        </node>
        <edge id="e0" source="n0" target="n2">
            <data key="d1">1.0</data>
        </edge>
        <edge id="e1" source="n0" target="n1">
            <data key="d1">1.0</data>
        </edge>
            <data key="d1">2.0</data>
        </edge>
        <edge id="e3" source="n3" target="n2"/>
        <edge id="e4" source="n2" target="n4"/>
        <edge id="e5" source="n3" target="n5"/>
        <edge id="e6" source="n5" target="n4">
            <data key="d1">1.1</data>
        </edge>
    </graph>
</graphml>
```

2. Prototipa grafa formāta piemērs

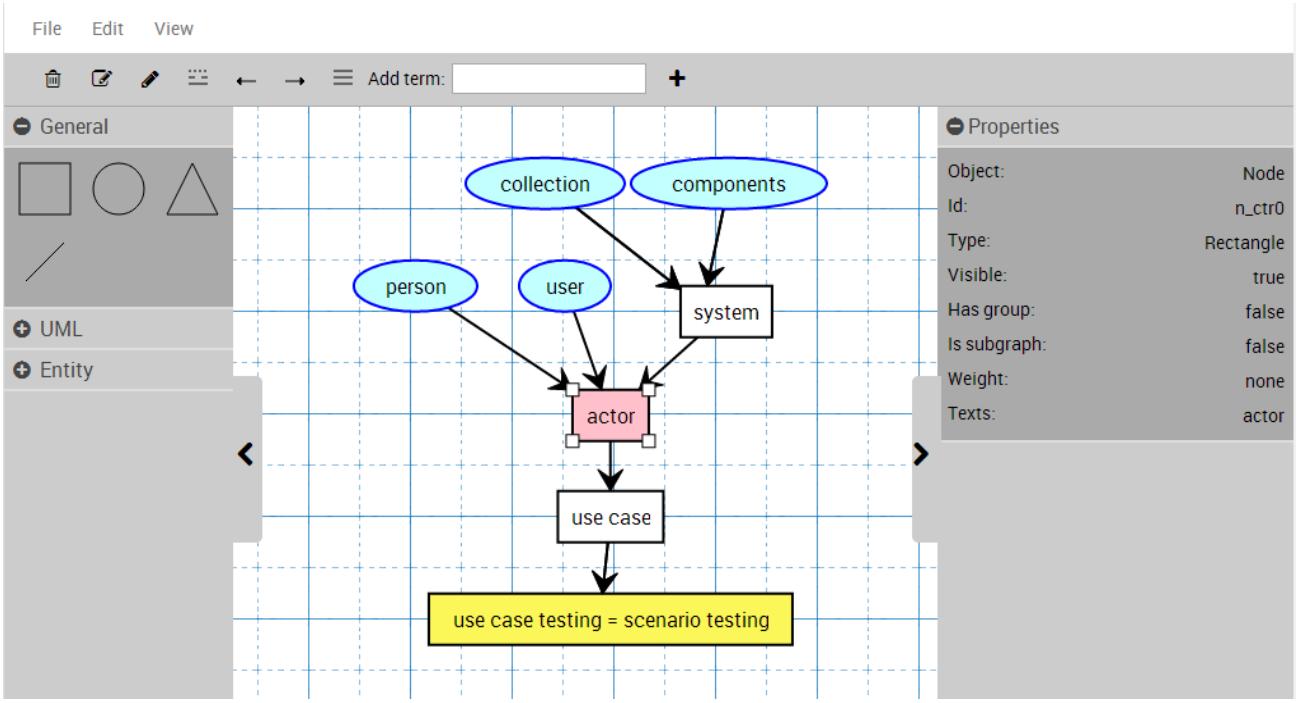
```
{ "graph": {
        "type": "basic",
        "nodes": {
            "Nd_0": {
                "data": {
                    "name": "Text"
                },
                "weight": false
            },
```

```
            "Nd_1": {
                "data": {
                    "name": "Text"
                },
                "weight": false
            },
            "Nd_2": {
                "data": {
                    "name": "Text"
                },
                "weight": false
            },
            "Nd_3": {
                "data": {
                    "name": "Text"
                },
                "weight": false
            }
        },
        "relations": {
            "Rltn_0": {
                "start": "Nd_0",
                "end": "Nd_3",
                "text": "",
                "weight": false,
                "direction": 0
            },
            "Rltn_1": {
                "start": "Nd_2",
                "end": "Nd_1",
                "text": "",
                "weight": false,
                "direction": 0
            },
            "Rltn_2": {
                "start": "Nd_0",
                "end": "Nd 2",
                "text": "",
                "weight": false,
                "direction": 0
            },
            "Rltn_3": {
                "start": "Nd_3",
                "end": "Nd_1",
                "text": "",
                "weight": false,
                "direction": 0
            }
        }
    },
    "visuals": {
        "nodes": {
            "Nd_0": {
                "shape": "Rectangle",
                "x": 408,
                "y": 71,
                "fill": "#fff",
                "stroke_color": "#000",
                "stroke_width": 1,
                "dash": ""
            },
            "Nd_1": {
                "shape": "Circle",
```

```
            "x": 508.96000000000004,
            "y": 245,
            "fill": "#2F2",
            "stroke_color": "#000",
            "stroke_width": 1,
            "dash": ""
        },
        "Nd_2": {
            "shape": "Triangle",
            "x": 286,
            "y": 231,
            "fill": "#F22",
            "stroke_color": "#000",
            "stroke_width": 1,
            "dash": ""
        },
        "Nd_3": {
            "shape": "Romb",
            "x": 649,
            "y": 26,
            "fill": "#22F",
            "stroke_color": "#000",
            "stroke_width": 1,
            "dash": ""
        }
    },
    "relations": {
        "Rltn_0": {
            "width": "2.00",
            "fill": "#000",
            "dash": "",
            "type": "Basic straight",
            "eX": 649,
            "eY": 106,
            "sX": 488,
            "sY": 93.990000000000001,
            "head": 0,
            "tail": 0
        },
        "Rltn_1": {
            "width": "2.00",
            "fill": "#000",
            "dash": "",
            "type": "Basic straight",
            "eX": 509,
            "eY": 271,
            "sX": 326,
            "sY": 271,
            "head": 0,
            "tail": 0
        },
        "Rltn_2": {
            "width": "2.00",
            "fill": "#000",
            "dash": "",
            "type": "Basic straight",
            "eX": 326,
            "eY": 271,
            "sX": 434.44000000000005,
            "sY": 111,
            "head": 0,
            "tail": 0
        },
```

```
            "Rltn_3": {
                "width": "2.00",
                "fill": "#000",
                "dash": "",
                "type": "Basic straight",
                "eX": 560.98,
                "eY": 257.5899999999999,
                "sX": 689,
                "sY": 186,
                "head": 0,
                "tail": 0
}}}}
```


3. Prototipa ekrānuzņēmums ar jēdziena tīkla piemēru.



DOKUMENTĀRĀ LAPA

Bakalaura darbs „Grafu pārlūkošana tīmekļa pārlūkā” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Aleksandrs Rilins _____

Rekomendēju / nerekomendēju darbu aizstāvēšanai

Vadītājs: profesors Dr. dat. Guntis Arnicāns _____

Recenzents: docents Dr.sc.comp Agris Šostaks.

Darbs iesniegts: 02.06.2014.

Dekāna pilnvarotā persona: metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē:

___.06.2014

Komisijas sekretārs: _____