Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Evidenčné číslo: FIIT-5212-5748

Lukáš Sekerák

Interaktívna vizualizácia informačnej siete

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Miesto vypracovania: Ústav informatiky a softvérového inžinierstva, FIIT STU Bratislava
Vedúci práce: RNDr. Michal Laclavík, PhD.

máj 2013

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika

Autor: Lukáš Sekerák
Bakalárska práca: Interaktívna vizualizácia informačnej siete
Vedúci bakalárskej práce: RNDr. Michal Laclavík, PhD.
máj, 2013

Táto práca je zameraná na riešenie teda zlepšenie interakcie pri vizualizácií informačných sieti. Informačná sieť je najčastejšie reprezentovaná vo forme grafu, matematického grafu. Táto forma dát je čoraz používanejšia ako prirodzená forma dát. Ich využitie je napríklad pri sociálnych sieťach, wikipédií, e-mailu a celkovo internetu. Vizualizácia informačnej siete ma teda veľké využitie. Existujúce riešenia vizualizácie sú však problémové, dokážu zobraziť len malé množstvo údajov a nie sú dostatočné prehľadné. Preto sa budeme snažiť vybrať čo najlepšie riešenie, zdokonaliť ho a prísť s ďalšími novými nápadmi. Opisujeme jednotlivé interaktívne nástroje, prvky zobrazenia, filtrovanie vrcholov a hrán. Zameriame sa aj na existujúce súborové formáty, trochu spomenieme fyzickú reprezentáciu dát. Všade spomenieme pozitíva a negatíva každého riešenia a v závere popíšeme riešenie ktoré je najlepšie a prečo sme si ho vybrali. Výsledkom práce bude funkčná aplikácia, ktorá vysvetli vzťah medzi vybranými vrcholmi grafu. Aplikácia by mala byť dostatočne interaktívna mala by zobraziť údaje, ktoré sú zadefinované v súbore. Nie je žiaduce, aby aplikácia načítala obrovské grafy.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatics

Author: Lukáš Sekerák
Bachelor Thesis: Interactive visualization of information network
Supervisor: RNDr. Michal Laclavík, PhD.
2013, May

This thesis is focused on solution and enhancement of interaction of information networks visualization. An information network is most frequently represented in form of a graph, a mathematical graph, which is increasingly used as a natural / basic form of data. Information networks visualization has wide range of use which can be found in f.e. social networks, Wikipedia, e-mail and internet in general. However, currently existing visualization solutions are problematic. They only display little volume of data in an insufficiently organized manner. In this thesis, we will strive for the best solution, We will try to perfect it and come up with another new ideas. We describe particular interactive Instruments, display elements, filtration of vertices and edges. As well as that, we will focus on existing format of files and briefly comprise physical representation of data. In each section we aim to state positive and negative aspects of every potential solution and the conclusion will include description of the solution which we consider to be the best, supported by explanation of my choice. My work is supposed to result in a functional application, which will reflect and explain the relation between selected vertices of the graph. The application should be interactive enough, display the data defined in file. User should avoid loading of overly complex graphs.

Obsah

1. Úvod

Cieľom práce je navrhnúť interaktívnu vizualizáciu informačnej siete. Túto informačnú sieť môžeme chápať ako graf. Preto sa v prvej časti zameriam najmä na vysvetlenie čo je to graf, aké vlastnosti grafu poznáme a do akých kategórii môžeme grafy deliť. Všetko to má veľký význam, pojmy je potrebne si najprv zadefinovať skôr ako ich používame. Najmä je potrebné si vysvetliť čo je to tá informačná sieť. Ako ju môžeme vizualizovať, ako interaktívne môžeme komunikovať s používateľom. To všetko vysvetlíme v prvej kapitole riešenia. Neskôr keď zistíme čo je to graf, uvedomíme si, že na grafe ma veľký význam usporiadanie a pozícia entít, najmä z vizuálneho hľadiska. Preto sme podrobnejšie popísali často používané pozičné algoritmy. Mnohé z nich sú komplikované, iné jednoduché, ale viac ako slova pozičný algoritmus popíše náhľad, preto pri každom jednom môžeme nájsť náhľad.

V rámci druhej kapitoly sa zameriame na analýzu problému, kde zistíme, že problém nie je až taký komplikovaný a problém rieši mnoho existujúcich riešení. Každé jedno riešenie ma rôzne plusy a mínusy, avšak ani jedno nie je dokonalé. Preto sme skúsili spísať všetky dôležité informácie o týchto riešení a vybrať to najlepšie.

V tretej kapitole sme sa sčasti pozreli na návrh riešenia, v úvode tejto kapitoly opisujeme, že riešenie sa pravdepodobne použije a otestuje na nejakej existujúcej sieti. Pričom je už potrebne sa zamyslieť nad logickou reprezentáciou dát, aké vizualizačné nástroje budeme chcieť implementovať. Dôležité je tiež pozrieť sa na nevýhody aktuálne zvoleného riešenia a skúsiť ich vyriešiť. Postupne v tejto kapitole prechádzam do podrobnosti, navrhnem fyzicky diagram tried a diagram balíkov. Stále táto kapitola je len návrhom možného riešenia a neskôr sa môže veľa veci zmeniť.

2. Informačná sieť

2.1 Graf ako informačná sieť

Každá informačná sieť alebo jej dáta sa dajú reprezentovať grafom. Graf je zase usporiadaná dvojica G=(V,E). V je konečná množina vrcholov a E je konečná množina hrán medzi vrcholmi. Každý údaj v informačnej sieti je reprezentovaný vrcholom (entitou) a každý vzťah (relácia) medzi vrcholmi hranou. Pohľadom na graf je teda možné pozorovať relácie medzi údajmi. [12] Vzťahy medzi vzdialenejšími vrcholmi, alebo podobné vrcholy sa dajú nájsť za pomoci hľadania najkratšej cesty. Na hľadanie najkratšej cesty v grafe sa často používa Dijkstrov algoritmus.[12]

Podľa množiny hrán delíme grafy na orientované a neorientované. Graf je orientovaný pravé vtedy, keď množinu hrán tvoria usporiadané dvojice vrcholov. Keď množinu hrán tvoria neusporiadané dvojice vrcholov, voláme graf neorientovaný.

Ďalej grafy delíme na grafy malého a veľkého sveta. Grafy malého sveta majú odlišné vlastnosti. Les alebo acyklicky graf je graf, v ktorom nie je kružnica (cyklus). Opakom je cyklicky graf. Hrany môžu byt ohodnotené, najčastejšie ohodnotené číslom alebo nemusia byť vôbec ohodnotené.

Grafy môžeme deliť do ďalších kategórii. Napríklad podľa pomeru počtu vrcholov a počtu hrán na grafy husté, riedke. Husté grafy majú zväčša niekoľko násobne viac hrán ako vrcholov.

2.2 Vizualizácia sieti

Pri vizualizácií ma veľký zmysel zobrazovať najmä podstatné informácie. Pri veľkých grafoch to je vážny problém, pretože tie obsahujú veľké množstvo elementov. Preto je potrebné tieto elementy – informácie prefiltrovať a zobraziť len to podstatné.

Jeden problém súvisí s ľudskou pamäťou a Millerovým zákonom[11] a druhý s vizuálnym neporiadkom v grafe, pričom Ellis, Dix a Aris uvádzajú ako hlavný dôvod problému skutočnosť, že sa zobrazuje príliš veľa dát na príliš malej časti displeja [6][20].

Cieľom je teda čo najlepšia filtrácia údajov, filtrovať informácie môžeme rôznymi metódami. Či už automaticky alebo manuálne. Filtrovanie údajov v našom prípade budú robiť nástroje, ktoré používateľ bude môcť ovládať.

2.3 Interakcia človeka s vizualizáciou

Interakcia alebo ovládanie vizualizácie je dnes možné viacerými spôsobmi. Najprv si treba ale zadefinovať aké vizualizácie poznáme. Môžeme ich rozdeliť na vizualizáciu prvkov v 2D, teda dvojrozmernom svete alebo 3D, trojrozmernom svete. Niektoré ovládanie je vhodné pre jeden svet iné pre druhý.

Ovládanie vizualizácie je možné cez rôzne zariadenia, najbežnejšie zariadenie je myš. Ďalšie ovládanie je možné pomocou dotykového zariadenia, napríklad tabletu. Bezdotykové ovládanie pomocou kinectu[1] či leapmotion-u[2], umožňuje ovládanie pomocou pohybov rúk (gestá). Ovládacie zariadenie je podstatné, ešte dôležitejšie je kvalitné užívateľské prostredie a v rámci neho použitie klávesových skratiek. Prostredie a klávesové skratky urýchlia a sprehľadnia kontrolu nad vizualizáciou.[14]

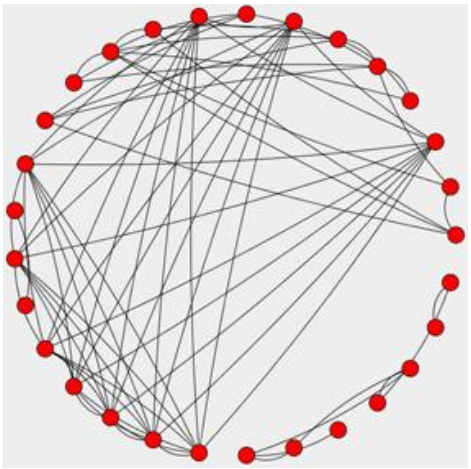
2.4 Algoritmy rozmiestnenia

V rámci vizualizácie ma veľký význam hovoriť aj o rozložení prvkov, konkrétne o rozložení vrcholov. S teórie grafov poznáme rôzne algoritmy, ktoré dokážu automaticky a kvalitne rozmiestniť vrcholy. Tieto algoritmy popisujem v ďalšej časti tohto dokumentu. Je dôležité nezabudnúť aby používateľ mal šancu aj manuálne presúvať elementy.

Ak informačnú sieť chápeme ako graf a chceme aby graf bol čo najčitateľnejší, potrebujeme vrcholy rozmiestniť čo najúčinnejšie. Hrany také ako samotne nepotrebujeme rozmiestniť, ich pozícia závisí aj tak od vrcholov. Pri jednoduchom grafe rozmiestnenie môžeme spraviť ručne, s narastajúcim počtom vrcholov to začína byt problém. Preto to treba zautomatizovať a to je možné vďaka existujúcim algoritmom. Každý algoritmus rozmiestňuje vrcholy podľa určitých kritérií.

2.4.1 Circle layout

Algoritmus circle layout je najjednoduchší a jeho názov hovorí za všetko. Algoritmus sa snaží usporiadať vrcholy do kruhu. Je najrýchlejší ale menej prehľadný pri veľkom počte vrcholov, pretože daný kruh bude obrovský.



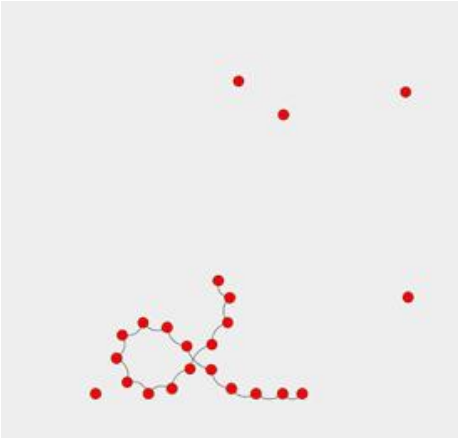
Obr. 1. : Použitie Circle layout za pomoci Jung-u.

[1] http://en.wikipedia.org/wiki/Kinect

[2] https://leapmotion.com/

2.4.2 Spring layout

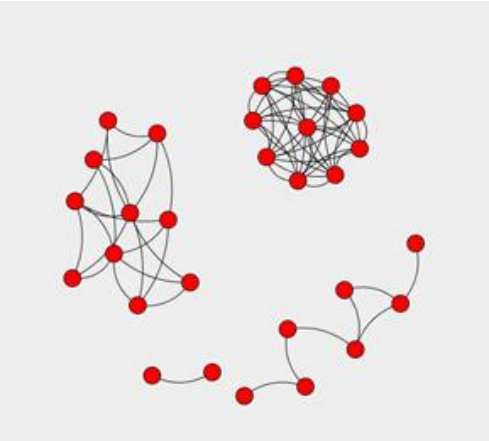
Rozmiestnenie pomocou spring layout nie je opäť komplikované. Pracuje na princípe takom, že vrcholy spojené hranou priblíž k sebe a vrcholy bez hrán skús oddialiť. Je vhodný pre všeobecné grafy, je ale už pomalší ako circle layout.[3]



Obr. 2. : Použitie Spring layout za pomoci Jung-u.

2.4.3 Kamada-Kawai layout

Kamada-Kawai algoritmus sa snaží rozložiť graf pomocou vzťahu druhá mocnina zo sumy medzi ideálnou a aktuálnou vzdialenosťou, medzi vrcholmi. Zjednodušene povedané, graf si treba predstaviť ako sústavu pružín, hranu ako energiu medzi pružinami a cieľom je minimalizovať celkovú energiu pružín.[2] Toto je teda silovo orientovaný rozmiestňovací algoritmus.[2]



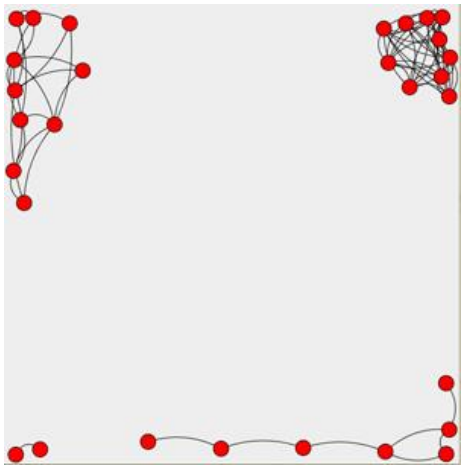
Obr.3. : Použitie Kamada-Kawai layout za pomoci Jung-u.

2.4.4 Fruchterman-Reingold layout

Tento algoritmus je veľmi podobný ako Kamada-Kawai, tiež sa na graf pozerá ako na sústavu energie. Je to tiež silovo orientovaný rozmiestňovací algoritmus.[15] Vyvážený stav energie, teda čo najlepšie rozmiestnenie, dosahuje postupným približovaním vrcholov k sebe. Vrcholy sa prestanú hýbať keď sa

[3] Tvrdenie vychádza z osobného test na JUNG knižnici. Spring layout vyžadoval určitý čas na prípravu.

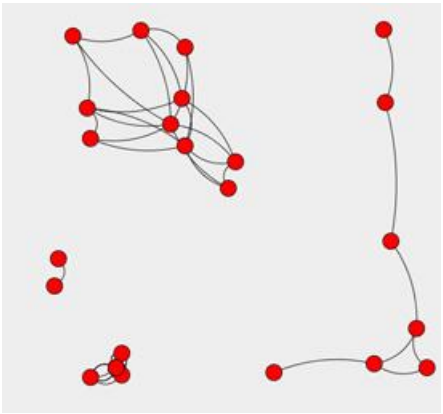
dosiahne vyvážený stav, to však môže trvať dlhú dobu a samotný pohyb vrcholov nie je pre vizualizáciu dobrý.



Obr. 4. : Použitie Fruchterman-Reingold layout za pomoci Jung-u.

2.4.5 ISOM layout

ISOM je založený na princípe invertovaných samo-organizačných máp.[2] Samo-organizované mapy pracujú na podobnom princípe ako silovo orientované algoritmy rozmiestňovania. Tie boli spomínané vyššie, sú to Fruchterman-Reingold, Kamada-Kawai. Rozdiel je v tom, že tento algoritmus sa snaží rozdeliť priestor medzi vrcholy a nie vrcholy do priestoru. Je vhodný pre husté grafy.

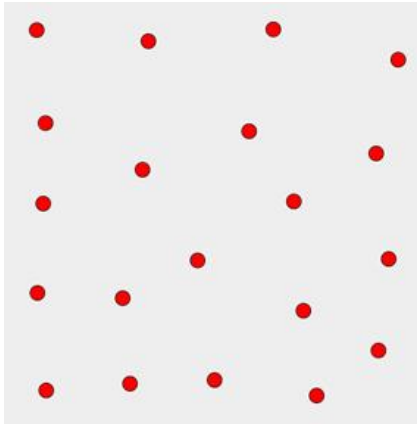


Obr. 5. : Použitie ISOM layout za pomoci Jung-u.

2.4.6 Iné rozmiestňovacie algoritmy

Doteraz som spomínal zložité algoritmy, tie graf rozložia do pekných útvarov. Sú ale časovo a pamäťovo náročné. Rozmiestniť vrcholy môžeme jednoducho náhodne do ohraničeného priestoru. Alebo rovnomerne do daného priestoru. Takto náš algoritmus nastaví pozíciu vrcholu veľmi rýchlo.[4]

[4] Pod pojmom veľmi rýchlo sa mysli čas. zložitosť O(n) kedy sa každému vrcholu nastaví pozícia len raz a už sa nemení.



Obr. 6. : Použité náhodné rozmiestňovanie za pomoci Jung-u.

2.4.7 Najlepší algoritmus rozmiestnenia

Po analýze algoritmov, by sme mali vybrať jeden a označiť ho ako najlepší. Avšak to sa v tomto bode nedá určiť, algoritmy majú rôzne vlastnosti. Vyber algoritmu závisí od štýlu jeho použitia a od faktu či ho naše budúce riešenie bude podporovať. Najlepší algoritmus teda vyberiem neskôr, až po analýze existujúcich riešení v kapitole 3.2.2 Jung a algoritmy rozmiestnenia.

2.5 Typy interakcii grafu s človekom

Medzi základne nástroje alebo možnosti patrí **pohybovanie** sa medzi prvkami siete, je dôležité sa presúvať z jednej strany grafu na inú stranu. S tým súvisí aj **približovanie a oddiaľovanie**, aby sme dokázali sa zamerať na určitú informáciu. Ako sme spomenuli, pohybovať sa medzi vrcholmi môžeme automaticky, manuálne je to vhodne aby sme dokázali **zoskupiť** podobné informácie alebo oddialiť tie informácie, ktoré nepotrebujeme. Zaujímavé je tiež prejdenie z jedného vrcholu na susedný cez danú hranu.

Medzi zložitejšie ovládacie prvky môžeme zaradiť možnosť rôzne farebne odlíšiť prvky, prípadne niektoré zvýrazniť nad ostatnými. **Vyhľadávanie** prvkov, či ich priame filtrovanie podľa hodnôt, sú tiež zaujímavé nástroje. Ak v grafe máme obrovský počet prvkov, časť si **skryjeme** napríklad podľa kľúča alebo hodnoty. A neskôr si môžeme konkrétny prvok nájsť. Mnohé aplikácie majú automatizované filtrovanie. To môže byť nebezpečné, môže byť nesprávne alebo automatizované algoritmy sa môžu učiť na nesprávnych vzorkách.

Zaujímavé filtrovanie vrcholov dosahuje metóda **zabaľuj a rozbaľuj**. Po vybratí určitých vrcholov a nájdení najkratšej cesty medzi nimi sú určíte vrcholy dôležitejšie, tie sa zobrazia. Zvyšné vrcholy, podľa hodnoty ,,treeshold“ sa zabalia do tých dôležitejších. Takto sa zobrazia len dôležité informácie, menej dôležité sú len dočasné schované. Niektoré vrcholy si môžeme rozbaliť a takto získať dodatočné informácie.

3. Analýza

V rámci úvodu sme zistili, že sieť môžeme zobrazovať ako graf a graf môžeme zase zobrazovať za pomoci existujúceho nástroja, ak taký existuje. V ďalšej kapitole sa pozrieme či existujú nejaké riešenia, popíšeme ich a vyberiem to najlepšie. Každý jeden nástroj však potrebuje určitý vstup, lepšie povedané je dôležité sa zamyslieť aj na formáte dát, ktoré vstupujú do nástroja. Preto som sa pozrel aj na túto kategóriu.

3.1 Analýza existujúcich možností

Na podporu vizualizácie informačnej siete existujú rôzne nástroje, knižnice. Tieto nástroje sa od seba pomerne dosť líšia. Niektoré sa zameriavajú na výkonnosť, na spracovanie veľkých dát, iné na pekné zobrazenie alebo len na analýzu grafov. Všetky vlastnosti týchto programov sa nedajú spísať, náhľad nám však trocha pomôže. Niekoľko nástrojov som vybral a spomenul ich v nasledujúcej kapitole. Pri každom jednom nástroji uvádzam odkaz na domovskú stránku projektu, kde je možné získať viac informácii.

3.1.1 Jung

Jung[5] je štandardná knižnica pre univerzálne siete, nezameriava sa na veľké či malé grafy ale skôr na ich kvalitnú vizualizáciu. Ponúka širokú paletu nástrojov, rozhraní, rôzne módy zobrazovania a základné operácie ako je pohyb po grafe a presun prvkov. Funguje ako aj editor a môže pridávať, vymazávať prvky. API Jung-u je kvalitne prepracované, využíva aj ďalšie šikovné knižnice ako Apache collections[6].

Jung využívajú napríklad tieto nástroje ExtC[7], RDF Gravity[8], GUESS[9] a minimálne ďalších 20 nespomenutých nástrojov. Taktiež bol použitý v rôznych odborných výskumoch a prácach. [10]

Jung poskytuje mechanizmy pre anotáciu grafov, entít a vzťahov s metadátami. [16]

[5] http://jung.sourceforge.net/index.html

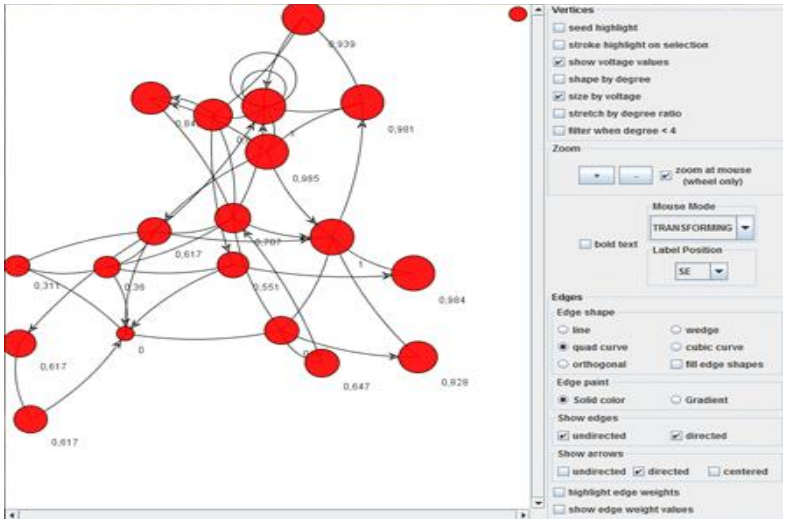
[6] http://commons.apache.org/collections/

[7] Eclipse prídavok - http://code.google.com/p/ext-c/

[8] Vizualizátor pre grafy typu RDF / OWL - http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html

[9] Sieťový analyzátor, spolupracuje s Pythonom - http://www.hpl.hp.com/research/idl/projects/graphs/index.html

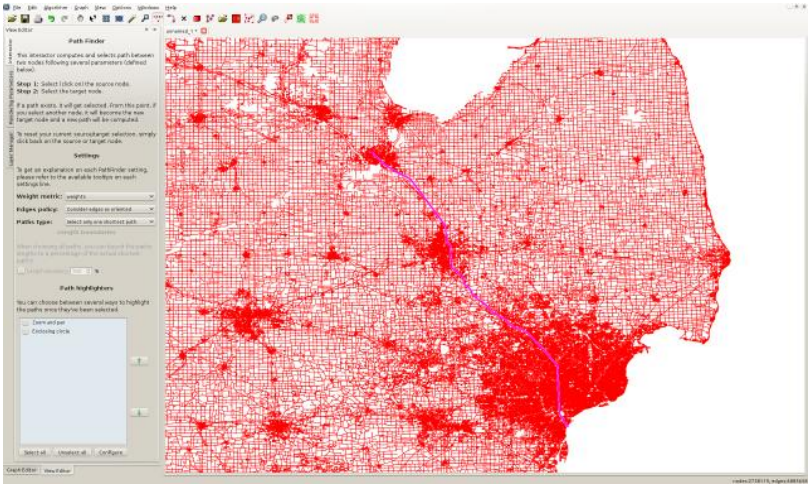
[10] Zoznam výskumných projektov, programov ktoré použili JUNG - http://sourceforge.net/apps/trac/jung/wiki/ProjectsUsingJUNG



Obr. 7. : Znázornené prostredie a možnosti v Jung-u.
Obrázok je úsek z jedného dema aplikácie.

3.1.2 Tulip

Tulip[11] je framework alebo väčší softwarový systém, ktorý je určený pre obrovské grafy. Sám dokáže zobraziť vyše 1 milión vrcholov a hrán na štandardnom osobnom počítači, to všetko dokáže vďaka jeho architektúre. Je určený najmä pre trojrozmernú vizualizáciu grafov, podporuje rôzne zásuvné moduly. A ako aj iné aplikácie rôzne pohybovanie, približovanie, zoskupovanie vrcholov. Zoskupovať vrcholy dokáže aj automaticky. Okrem grafov dokáže zobrazovať aj diagramy. Prvky vykresľuje podľa rôznych farieb a dokáže trocha aj analyzovať dáta. Tulip je naozaj veľmi dobrý kandidát na naše prostredie. Framework je napísaný v C++ pre Windows[4], takže dáva vizualizáciu veľkú silu a výkonnosť, je to objektový jazyk, takže programovanie tiež nebude pomalé. Okrem toho na stránke projektu je prístupná dobrá dokumentácia, rôzne návody, fórum, teda je tu dobrá podpora pre vývojárov . Produkt je pod LGPL licenciou.



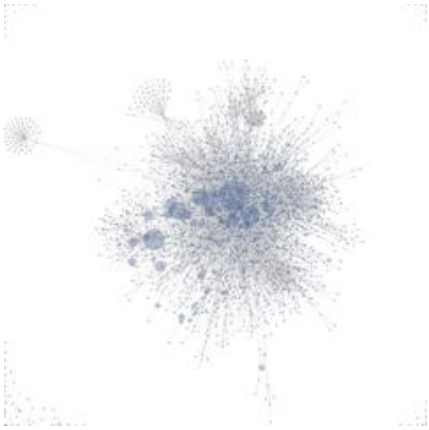
Obr. 8. : Obrázok[12] znázorňuje použitie Tulip-u ako vizualizátora, kde zobrazuje diaľnice a cesty v USA. Pravé načítal graf s 2,758,119 vrcholmi a 6,885,658 hrán.

[11] http://tulip.labri.fr/TulipDrupal/

[12] Obrázok je použitý z http://tulip.labri.fr/TulipDrupal/?q=node/971

3.1.3 Prefuse

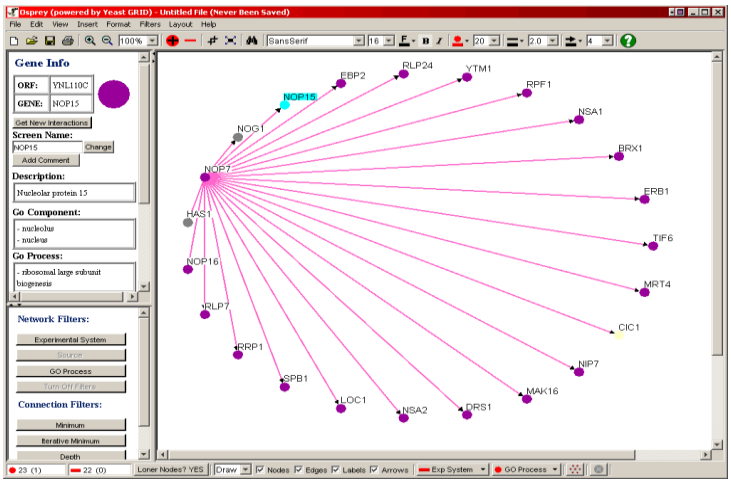
Prefuse[13] nie je samotný nástroj ale je to sada skvelých a interaktívnych nástrojov pre vizualizáciu dát. Pre podporu vizualizácie obsahuje rôzne dátové štruktúry pre rôzne typy dát. Samozrejme aj pre grafy. Dokáže pracovať s databázou v podobe Swing aplikácie alebo Java appletu. Je napísaný v programovacom jazyku Java[7], teda rovnako ako Jung. A je ho možné použiť pre komerciálne aj nekomerciálne účely, lebo je pod BSD licenciou. To umožnilo použiť Prefuse v mnohých projektoch, podobných tejto práci. V sade nástrojov je zahrnutá aj knižnica pozičných algoritmov, navigácie a interaktívnych techník.[7]



Obr. 9. : Obrázok[14] znázorňuje štruktúru liniek na wiki. Štruktúra bola zobrazená za pomoci Prefuse.

3.1.4 Osprey[15]

Je ďalší software na vizualizáciu komplexných sieti, používa sa napríklad na vizualizáciu génov. Existuje distribúcia pre Windows, Linux, Mac OS.



Obr. 10. : Obrázok[16] znázorňuje DNA a rôzne gény za pomoci nástroja Osprey.

---

[13] http://prefuse.org/

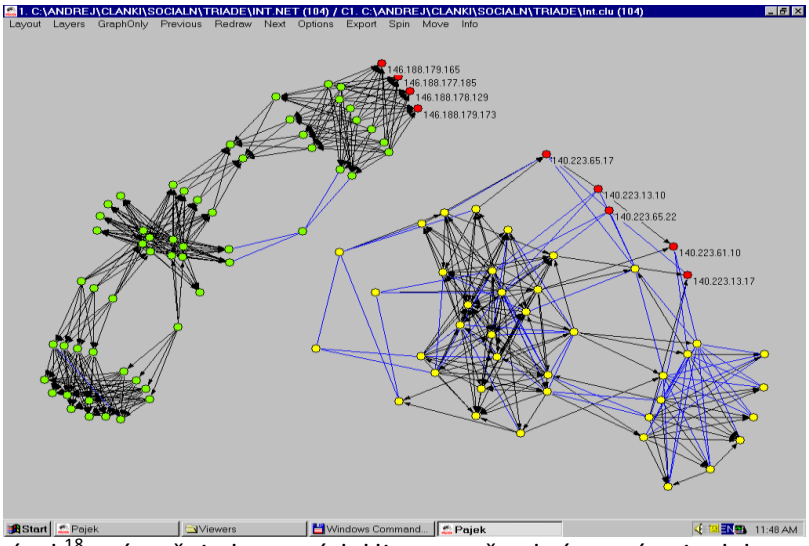
[14] Obrázok je použitý z - http://en.wikipedia.org/wiki/File:Visualization_of_wiki_structure_using_prefuse_visualization_package.png

[15] http://biodata.mshri.on.ca/osprey/servlet/Index

3.1.5 Pajek

Pajek[17] alebo spider je program pre analýzu veľmi veľkých sieti. Je primárne určený pre Windows, na Linuxe beží za pomoci Wine. Podporuje viaceré šablóny rozmiestnenia. Napríklad GD95, GD96, GD97, GD98, GD99, GD00, GD01, GD05.[13] Je voľne šíriteľný pre nekomerciálne účely.[13] Podporuje troj aj dvoj dimenzionálne zobrazovanie, taktiež viaceré formáty súborov. Avšak tento nástroj je pomerne starý.

Mnoho štandardných sieťových algoritmov sú časovo a pamäťovo konzumne a skoro sú nedostatočné pre analýzu takých sad typov sieti.[3]



Obr. 11. : Obrázok[18] znázorňuje koncových klientov a štruktúru v rámci telekomunikačnej sieti. Sieť je zobrazená v programe Pajek.

3.2 Výber riešenia

Je jasné, že takýchto knižníc je hojné množstvo, preto nemá zmysel začať vytvárať vlastnú knižnicu a je lepšie si jednu vybrať. Po rýchlom porovnaní sme hneď vylúčili Osprey, pretože ten nie je práve určený pre vizualizáciu všeobecných sieti ale len génov. Z rovnakého dôvodu sme vylúčili aj Pajek. Pajek je dokonca veľmi starý program.

Jung, Prefuse a Tulip sú si veľmi podobné. Každý jeden z nich obsahuje veľké knižnice, štruktúry a pomôcky pre vizualizáciu a interakciu s používateľom. Každý jeden z nich bol použitý vo viacerých projektoch, takže určite všetky majú veľký význam. Rovnako každý je riešený objektovo orientovaným pohľadom. Teda pri budúcej implementácii by rýchlosť implementácie mala byť dobrá.

Avšak z hľadiska vytvárania okna aplikácie, vytvárania užívateľského prostredia či možnom portovaní aplikácie na iné zariadenie Java vyhráva. Pretože Java jazyk je jednoduchší na portovania, má lepšiu podporu a lepšie spravené knižnice pre vytvorenie príjemného užívateľského prostredia. V tomto smere

---

[16] Obrázok je použitý z - http://biodata.mshri.on.ca/osprey/servlet/PhotoViewer?image_to_show=harpanichi.gif

[17] http://vlado.fmf.uni-lj.si/pub/networks/pajek/

[18] Obrázok je použitý z - http://vlado.fmf.uni-lj.si/pub/networks/doc/gd.01/Pajek2.png

zamietam kandidáta Tulip. Na druhej strane si treba uvedomiť, že možná výkonnosť aplikácie je Javou ohrozená a bude potrebné to nejako riešiť.

Po porovnaní Jung a Prefuse sme si vybrali knižnicu Jung. Dôvodom je o trocha lepšia dokumentácia a podpora pre vývojárov, zároveň viacero príkladov. Túto knižnicu sme podrobnejšie rozobrali v nasledujúcej kapitole.

3.2.1 Jung podrobnejšie

Táto skratka je odvodená od Java Universal Network Graph čo je software alebo knižnica, ktorá poskytuje možnosti pre zobrazenie a úpravu grafov. Samozrejme dokáže aj analyzovať graf a poskytuje ďalšiu širokú paletu nástrojov. Všetky tieto funkcie sú zabalené v jednoduchom a dobre zdokumentovateľnom API. Knižnica je napísaná v Java programovacom jazyku,[16] existujú ale aj jej varianty pre ďalšie jazyky. V našom prípade nám postačí Java, ktorá nám uľahčí a zrýchli implementáciu programu.

Jung dokáže zobraziť grafy malého, veľkého sveta.[16] Od grafov jednoduchých až po multi-grafy či hypergrafy.[16] Má zabudovanú podporu pre priame, nepriame alebo pre jednoduché a paralelné hrany. Ďalšie podporované mechanizmy sú anotácia vrcholu, anotácia celého alebo časti grafu.[16] Vyhľadávanie najkratšej cesty za pomoci rôznych ale najmä dijkstrového algoritmu je pre Jung samozrejmosť. Tieto mechanizmy hovoria samy za seba a Jung tak ma veľa možnosti ktoré využijeme.

Celá architektúra je objektovo orientovaná a každý vrchol, hrana, graf a iný objekt je reprezentovaný triedou.[19] Dáta mimo grafu, napríklad anotácie alebo názvy vrcholov, sú uložené v takzvaných meta-datach a je možné ich oddeliť od grafu, ukladať ich samotne a pod. Jung všade podporuje „Transformer“[20] návrhový vzor a tak aj rôzne dáta môžem ukladať na rôzne miesta, opäť ďalšia výhoda tejto knižnice.

Okrem voľne šíriteľnej dokumentácie ma knižnica sprístupnené zdrojové súbory pod licenciou Apache 2[21]. Táto licencia nám umožní zadarmo použiť knižnicu. V balíku knižnice sú aj príklady využitia tohto softwaru a tieto príklady sú opäť voľne stiahnuteľné.

Celkovo Jung hodnotím za najlepšieho kandidáta pre túto prácu. Vďaka nemu môžem využiť už existujúce nástroje a môžem sa lepšie zamerať na vlastné algoritmy. Na lepšiu vizualizáciu grafu a aj na lepšie užívateľské prostredie, ktoré prinesie lepšiu interakciu. Avšak knižnica nie je dokonalá a má ťažkosti pri zobrazení väčších grafov, tieto problémy som riešil v 4.5 Nevýhody Jung-u a ich riešenie kapitole.

3.2.2 Jung a algoritmy rozmiestnenia

V kapitole 2.4 Algoritmy rozmiestnenia sme vypísali algoritmy rozmiestnenia. Všetky tieto algoritmy JUNG podporuje, samotne ukážky boli práve vytvorene za pomoci Jung-u. Dodatočnou analýzou sme zistili, že väčšina algoritmov ma zložitosť $O(\log^2 n)$[10]. Náhodné a rovnomerne rozmiestnenie ma lepšiu zložitosť $O(n)$[22], avšak nepostačuje pre naše potreby (Vrcholy sú rozdelené rovnomerne, teda tie väčšie nie sú

[19] Tvrdenie vychádza z pozorovania zdrojových kódov knižnice Jung 2.0.1

[20] http://commons.apache.org/collections/apidocs/org/apache/commons/collections/Transformer.html

[21] http://www.apache.org/licenses/

[22] Tvrdenie vychádza z myšlienky, že sa každý vrchol bude prechádzať len raz a nastaví sa mu náhodná pozícia.

priorizovane a susedné vrcholy môžu byt ďalej vzdialené). Následné sme spravili malý experiment na rôznych vzorkách, kde sme pozorovali tieto algoritmy a najlepšie sa správal ISOM layout.

3.3 Súborové formáty

Štruktúra grafu sa najlepšie opíše pomocou množiny vrcholov a množiny hrán. Každý vrchol je zväčša opísaný jedinečnou hodnotou ale štandardne to býva tak, že vrchol ma ďalšie atribúty. Tieto atribúty budeme nazývať metadáta teda "dáta o dátach". Rovnako aj hrany môžu mať rôzne metadáta napríklad dôležité ohodnotenie hrany alebo popis.

Dnes poznáme rôzne štandardy pre formát súborov. Všetky zväčša podporujú metadáta.

3.3.1 GraphML

GraphML vychádza z jazyka XML[23], takže všetky vrcholy, hrany, vlastnosti grafu sú zapísané pomocou tagov[5]. Táto schéma je veľmi dynamická, do štruktúry grafu dokážeme zapísať aj iné údaje. Ak aplikácia nevyužíva určíte tagy alebo ich nepozná, môže ich jednoducho ignorovať. Základne tagy sú graph, edge, node.

Ukážka formátu GraphML:

```
<?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">
<graph edgedefault="directed">
<node id="v1"/>
<node id="v2"/>
<node id="v3"/>
 <node id="v4"/>
<edge source="v1" target="v2"/>
 <edge source="v1" target="v3"/>
 <edge source="v2" target="v4"/>
<edge source="v2" target="v4" directed="false"/>
</graph>
</graphml>[5]
```

3.3.2 Pajek NET

Pajek NET[24] je formát štandardne nachádzajúci sa v textovom súbore. Kde každý riadok súboru ma význam predstavujúci vrchol alebo hranu. Pred zoznamom vrcholov či hrán sa nachádza informácia o zozname a počte prvkov v zozname. Tento formát podporuje väčšina grafových programov, spomedzi spomenutých je to Jung aj Pajek.

---

[23] http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf

[24] Článok, ktorý popisuje formát - https://gephi.org/users/supported-graph-formats/pajek-net-format/

Ukážka formátu Pajek NET:

```
*Vertices 1542
1 "A"
2 "B"
3 "C"
4 "D"
*edgeslist
1 21 22 23 28
2 253
3 124
4 23
5 15 12
```

3.3.3 CSV

Formát CSV, teda hodnoty oddelené čiarkou[19] nie je primárne určený len pre grafy. Vďaka jednoduchosti a čitateľnosti sa používa pre výmenu informácii v rôznych systémom. Modernejší formát je XML. V prípade použitia tohto formátu je potrebné si formát prispôsobiť potrebám riešenia.

Ukážka formátu CSV:

```
vertexA
vertexB
edge1, vertexA, vertexB
```

3.3.4 gSemSearch formát

Nasledovný formát, nazvaný podľa gSemSearch nástroja, bude základným podporovaným formátom. Tento formát využívala Slovenská akadémia vied, pri generovaní menších častí Enron Corpus Grafu.[9] Prečo sa o tento formát a tento graf zaujímame spomenieme neskôr. Na začiatok je dôležité vedieť len ako formát vyzerá a aké ma vlastnosti. Nasledovný príklad je výťažok zopár riadkov zo súboru „enron.graph“[25], teda príklad jednej menšej časti Enron Corpus Grafu.

Ukážka formátu gSemSearch:

```
1 Vertex: Address=>9400 Circle Drive, Austin, Texas 78736        1
1 Vertex: Address=>Communications, 401 Carlson Circle  1
1 Vertex: Address=>Houston - 2929 Allen Parkway        2
1 Vertex: Address=>P.O. Box 15427, Austin, TX 78761-5427        1
1 Vertex: CityName=>Austin      8
1 Vertex: CityName=>College Station      1
1 Vertex: Company=>ACN Power, Inc.      2
1 Vertex: Company=>AES Corp.  1
1 Vertex: Company=>Addition of a        2
1 Vertex: Company=>Adobe      1
2 Edge: (Address=>100 Congress Avenue, Suite 800, Austin, Texas 78701)=>(CityName=>Austin)   2
```

[25] http://ikt.ui.sav.sk/esns/enron/

Po základnej analýze je vidieť, že údaje o vrcholoch začínajú prefixom „1 Vertex“, údaje o hranách majú prefix „2 Edge:“ . Jeden vrchol je na jednom riadku, hrana podobne. Každý vrchol sa skladá z 3 časti v tvare:[8]

```
A=>B C
A = Kľúč
B = Hodnota
C = Ohodnotenie
```

Každá hrana je reprezentovaná dvojicou vrcholov v tvare:

```
 (A)=>(B)  C
A = Zdrojový vrchol
B = Cieľový vrchol
C = Ohodnotenie hrany
```

Podrobnejšou analýzou týchto dát je vidieť, že vrcholy sú zoradené abecedne. Najprv podľa kľúčov, potom podľa hodnôt. Zároveň mnohé vrcholy majú rovnakú kľúče, sú to teda redundantne údaje. Tieto poznatky majú veľký prínos pre základný beh programu s obrovským množstvom dát. Čo si treba tiež všimnúť, najprv sú definované vrcholy a neskôr hrany, údaje nie sú pomiešané.

3.3.5 Výber formátu a záver z tejto časti

V predchádzajúcom závere sme si vybrali Jung riešenie, on sám podporuje väčšinu formátov. Nepodporuje CSV, ten však nechceme mať implementovaný. Pretože je dosť neprehľadný a nehovorí o dátach dostatočné informácie. Druhý formát, ktorý Jung nepozná, je gSemSearch. Tento formát je pre mňa primárny, preto ho musím do implementovať. gSemSearch formát má veľkú prioritu, testy programu budú zväčša na týchto dátach.

4. Návrh riešenia

Naše navrhované riešenie priamo počíta s využitím existujúceho riešenia. Pod existujúcim riešením sa mysli prostredie Jung, ktoré sme už podrobnejšie popísali. Riešenie bude vyhotovené ako základná aplikácia, s jedným oknom. Bude určená najmä pre operačný systém Windows. V krátkosti som spomenul aj nevýhody tejto knižnice, tieto negatíva budem riešiť v tejto kapitole. Čo konkrétne a aké detaily plánujem navrhnúť tiež popisujem v tejto kapitole.

4.1 Riešenie na skutočnej sieti

V rámci dohody so Slovenskou akadémiou vied, pre ktorú by tato práca priniesla prínos sa náš nástroj zameria na vizualizáciu informačnej siete nazvanej Enron Graph Corpus[26] a druhej informačnej siete Gorila. Slovenská akadémia vied využíva vlastný formát už spomínaný gSemSearch.[9] Pre väčšiu kompatibilitu využijeme pravé tento formát.

Podľa gSemSearch formátu, má každý vrchol kľúč, hodnotu reprezentovanú reťazcom a číselným ohodnotením vrcholu[8]. Každá hrana ma len 2 vrcholy a môže mať ohodnotenie.[8] Všetky tieto údaje chceme využiť v maximálnej miere, preto všetky údaje načítame.

Počet vrcholov sa väčšinou pohybuje v miliónoch, počet hrán v desiatkach miliónoch. Napríklad v študovanom súbore Enron Graph Corpus je presný počet 8,269,278 vrcholov a 20,383,709 hrán[8]. Po dohode požiadaviek, aplikácia bude schopná načítať niekoľko stá tisíc vrcholov a stá tisíc hrán. Samozrejme tieto čísla nebudú limity, aplikácia by teoreticky mohla spracovať ešte viac informácii. V konečnom dôsledku schopnosť aplikácie načítať veľké množstvo údajov bude závisieť od dostatku pamäťových zdrojov. Vizualizácia a analýza grafu bude zase závisieť od výkonnosti stroja, od vhodného filtrovania entít a konfigurácii metód zobrazenia.

Hlavnou prioritou sú stále grafy malého sveta. Grafy veľkého sveta budú možno podporované ale nebudú testované a predmetom záujmu.

4.1.1 Enron Graph Corpus

Tento graf[27] obsahuje milióny vrcholov, ktoré reprezentujú emaily, spoločnosti, telefónne čísla, ľudí, adresy atď. Hrany v tomto grafe reprezentujú vzťahy medzi týmito entitami. Graf vznikol vygenerovaním z emailov, ktoré posielali vysoký manažéri v Enron spoločnosti.

4.1.2 Gorila

Tento graf obsahuje vrcholy, ktoré reprezentujú rôzne spoločnosti, politické strany, osoby atd. Graf bol vygenerovaný zo spisu gorila. Cieľom nie je spájať rôzne organizácie ani osoby, vrcholy a fakty v ňom nie sú správne a nie sú založené na pravde. Cieľom je len ukážka, že môj nastroj môže byt použitý na vizualizáciu zaujímavých informácii.

[26] http://ikt.ui.sav.sk/esns/enron/

[27] Informácie o Enron Graph Corpus - http://www.cs.cmu.edu/~enron/

4.2 Vizualizácia grafu

Graf bude zobrazený čo najjednoduchšie, zameriame sa na zobrazenie údajov, nie útvarov alebo pekného výzoru. Vizualizácia bude v dvojrozmernom prostredí.

Samostatný vrchol zobrazíme jednoduchým kruhom, pravdepodobne nevyplnením. Nevyplnením s viacerých dôvodov. Má to zmysel pre výkonnosť vykresľovania a pri zobrazení viacerých vrcholov v malom prostredí nedôjde k úplnému zakrytiu údajov za vrcholmi. Prečo je to dôležité sme spomenuli v 2.2 Vizualizácia sieti. Okraj kruhu bude kreslený viacerými farbami, farba sa určí podľa skupiny, teda kľúča vrcholu. Farba kľúča vrcholu sa určí z určitého spektra farieb, dôležité je aby každý kľuč mal rôznu farbu čo najlepšie odlíšiteľnú.

Tvar vrcholu sa môže meniť podľa počtu hrán, môžu tak vzniknúť rôzne tvary. Typicky to budú trojuholníky. V nastavení zobrazenia bude položka, kde sa zapnú rôzne tvary vrcholu podľa počtu hrán. Veľkosť kruhu bude závisieť od polomeru a ten sa vypočíta pre každý jeden vrchol podľa ohodnotenia vrcholu. Dôvodom je aby vrcholy dôležitejšie boli väčšie. Aká presná bude funkcia na výpočet polomeru sa určí neskôr až pri implementácii.

Nad každým vrcholom bude zobrazený štítok, v ktorom zobrazíme kľuč a hodnotu vrcholu. Najlepšie v pôvodnom formáte, ako zo vstupného súboru. Zároveň plánujeme pridať možnosť nastaviť si tento štítok , aby si používateľ mohol zobraziť aj iné údaje o vrchole.

Hrany medzi vrcholmi sú ohodnotené číslom, ktoré reprezentuje “silu” vzťahu medzi dvoma údajmi. Sila nie je veľmi dôležitá, preto zobrazíme tento vzťah jednoduchou úsečkou spolu s použitím šípky, pretože vzťah nie je obojsmerný.

V rámci vizualizácie bude dôležité zobrazovať údaje, len ktoré sú v oblasti obrazovky . Podľa priblíženia obrazovky sa ale zobrazia niektoré údaje, čím obrazovka bude ďalej tak sa zobrazia len najväčšie a najdôležitejšie údaje. Pri plnom priblížení sa už zobrazia všetky údaje.

Pri potrebe zobraziť veľké množstvo údajov, budú vrcholy podobne alebo vrcholy príbuzné zoskupené alebo zabalené do jedného. Takéto zoskupenie už môžeme chápať ako nástroj, takže to je podrobnejšie popísané v ďalšej kapitole.

4.3 Naše interaktívne nástroje

Vo vyššej kapitole sme vymenovali niektoré interaktívne prvky. V našej aplikácii sa budeme snažiť zahrnúť všetky tieto interakcie v podobe nástrojov. Každý nástroj bude mať vlastnú zodpovednosť, podporovať nejakú funkcionalitu. Cieľom je poskytnúť čo najviac nástrojov, aby používateľ mal možnosť kontrolovať vizualizáciu a mohol analyzovať údaje.

Vizualizátor bude umožňovať pohyb v priestore, presun vrcholov, automatické zaostrenie na vrchol. V rámci pohybu v priestore sa bude dať približovať a oddiaľovať a tiež prejsť z vrcholu na susedný vrchol cez hranu. Toto presmerovanie bude zabezpečené v menu, po kliku na vrchol a ináč myšou. Hneď na začiatku všetky vrcholy budú automaticky rozdelené podľa šablóny (algoritmu rozmiestnenia - 2.4.5 ISOM layout) a potom bude možný len manuálny pohyb.

Zoskupovanie vrcholov bude automatizované, podľa akých špecifikácii sa určí ešte pri implementácii. Možné sú viaceré riešenia ale pravdepodobne pôjde o zoskupenie všetkých vrcholov s rovnakým kľúčom, alebo zoskupenie zopár vrcholov, ktoré priamo nesúvisia z označenými vrcholmi. Napríklad sa výberu 2 vrcholy, za pomoci najkratšej cesty sa zistí čo najlepší vzťah medzi vrcholmi a zvyšné vrcholy sa istou formou zabalia prípadne zhorší sa ich viditeľnosť.

Zo zložitejších nástrojov bude implementované zvýraznenie vrcholov. Vrchol sa zvýrazní tak, že jeho stred sa vyfarbí zelenou farbou. Používateľ najskôr označí skupinu vrcholov a to myšou alebo klávesovou skratkou, následné vyberie zvýrazniť skupinu alebo zvýrazniť jeden vrchol. Keď sa bude pracovať s vrcholmi, používateľ ich napríklad bude presúvať, tak budú automaticky zvýraznenie sivou farbou, po dokončení prace sa ich zvýraznenie zruší. Po kliku na vrchol, bude možné označiť alebo zvýrazniť jeho susedov a to všetko z interaktívneho menu.

Základným nástrojom určite bude filtrovanie vrcholov podľa kľúča alebo hodnoty. Bude to spojene do jednoduchého formulára, do ktorého sa zadá regulámy vyraz. Ďalším filtrovanie bude podľa veľkosti uhla medzi vrcholmi.

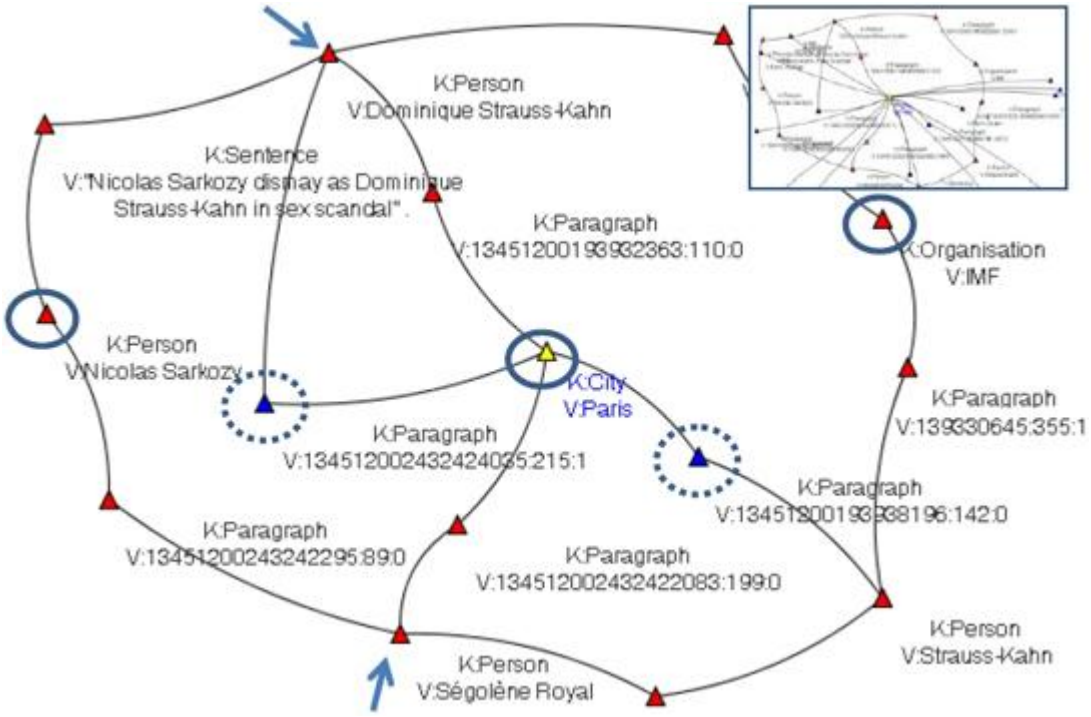
Každý nastroj sa istou formou bude ovládať z menu a tiež by sa mal dať spustiť za pomoci klávesovej skratky. Aby jeho ovládanie bolo interaktívne a rýchle. Keďže naše zadanie je vizualizácia už existujúcich prvkov a nie editácia prvkov, tak nemá zmysel riešiť pridávanie, úpravu čí mazanie prvkov v grafe.

4.4 Metóda zabaľovania a rozbaľovania

V rámci zadania navrhujeme použiť zabaľovanie a rozbaľovanie vrcholov, pričom sa bude pamätať aj na Millerov zákon. Zbalené vrcholy majú výhodu vtom, že ich je možné znovu rozbaliť, no pokiaľ sú zbalené, nezaberajú žiadny vizuálny priestor, nerušia sa s ostatnými vrcholmi a hranami a nevytvárajú neporiadok. [12]

Zabaľovanie a rozbaľovanie môže byt automatické. Ako sme spomenuli vyššie, automatizovaný proces môže byt nebezpečný, preto poskytneme možnosť manuálneho rozbaľovania balíka vrcholov. Pri manuálnom rozbaľovaní vrcholu je dôležitá hodnota *treeshold*, ktorá určuje, či sa majú zobraziť všetci skrytí príbuzní (počet príbuzných < treeshold) alebo len tí, ktorí sú zároveň príbuzní s inými, už zobrazenými vrcholmi. Implicitne, sa osvedčilo to aj podľa [11], kde sa hodnota treeshold stanovila na 7.[12]

Napríklad, ako demonštruje obr. 1, je rozbalený vrchol s hodnotou „Paris“. Tento vrchol má viac príbuzných, než je hodnota treeshold. Aj napriek veľkému počtu (nezobrazených) príbuzných, sú niektoré vrcholy následne zobrazené. Zobrazené sú však len tie, ktoré sú zároveň susedmi iných, už zobrazených vrcholov. Ukážka, ako by to vyzeralo, keby boli zobrazené všetky vrcholy, je v pravom hornom rámčeku v obr.1 a predstavuje vizuálny neporiadok v grafe.[12]



Obr.[12] **12.:** Šípky poukazujú na počiatočné vrcholy, koncové vrcholy sú v krúžkoch. V prerušovanom kruhu sú vrcholy, ktoré sa znovu objavia po rozbalení vrcholu „Paris".

Metóda zabaľovania je veľmi zaujíma, môže naozaj zlepšiť vizualizáciu, preto ju určite využijeme. Hore uvedená metóda bola vytvorená algoritmom najkratšej cesty medzi ,,many to many". V implementácii pravdepodobne pôjde o niečo podobné. Netreba však zabudnúť na to, že toto je automatizovaný proces, používateľ môže prísť o dáta, respektíve si ich nemusí všimnúť. Je preto potrebné pridať možnosť manuálneho rozbalenia. Prípadného úplného vypnutia tejto metódy.

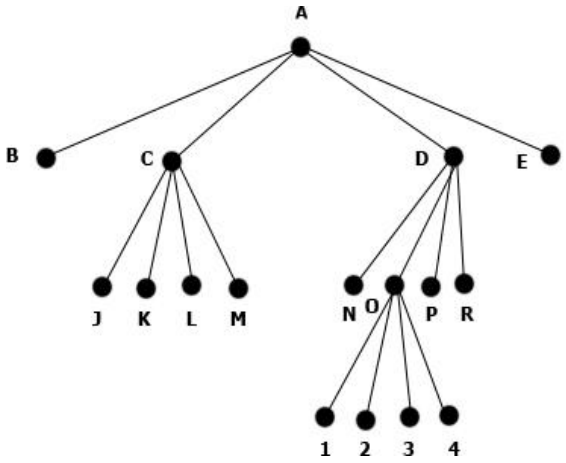
4.5 Nevýhody Jung-u a ich riešenie

Základnou nevýhodou Jung-u je práca s veľkými grafmi, pod pojmom veľkými sa mysli práca s miliónmi entít. Problém si treba rozdeliť do 2 časti, a to vizualizácia a uchovanie týchto dát v pamäti.

4.5.1 Problémová vizualizácia

V rámci vizualizácie Jung zobrazuje všetky vrcholy a entity, hoci priamo nie sú na obrazovke. Ma v sebe zabudované rôzne,, render-y", ktoré sa snažia ako tak vyfiltrovať entity ale ich časová zložitosť je O(n) kde n je počet entít. To nie je výhra, preto v rámci návrhu riešenia navrhujeme do Jung-u zakomponovať dátovú štruktúru Quad strom.

Quad strom je hierarchická dátová štruktúra. Je založená na princípe rekurzívnej dekompozície, podobne ako princíp rozdeľuj a panuj.[18] Teda rozdeľuje prvky v strome podľa ich pozície. Je určená pre dvojrozmerný priestor, je to analógia Octree. Quad strom rozdelí ohraničený priestor do 4 oblasti. Jednu oblasť rozdelí do ďalších 4 oblasti a takto sa to opakuje. Prvok sa priradí len do jednej oblasti podľa jeho pozície.



Obr. 13 : Obrázok znázorňuje Quad strom.

Quad strom dokáže vyhľadať, vložiť, vymazať prvok v časovej zložitosti O(log n). Pamäťová náročnosť je O(n), kde n je počet prvkov v strome. Treba si ale uvedomiť, že postačí keď strom bude uchovávať len referenciu na objekt. Čo je v rámci javy, jazyk v ktorom je napísaný Jung, obrovskou výhodou. Podrobnejšie referencie v Jave popisujem v kategórii 4.5.2 Čiastočný fyzický návrh.

Quad strom dokáže len nájsť oblasť, ktorej prvky treba zobraziť. To zistí pomocou prieniku obrazovky a časti. Zložitosť tejto operácie je O(log x)[1], kde x je počet časti, po výbere oblasti prejdeme všetky vrcholy, či sú v kolízii s pohľadom obrazovky. Zložitosť tejto operácie už je ale O(n), kde n je počet vrcholov v časti.[28] Takéto výsledky sú dostatočne efektívne na to, aby sme Quad strom implementovali. V prípade keď sa používateľ pozerá na určitú oblasť, spracujú sa prvky len s tej oblasti a to sme chceli dosiahnuť.

Do takého stromu dokážeme priraďovať prvky, ktoré majú určitú šírku, vrchol ma nepatrnú šírku a preto je jednoduché ho tam zahrnúť. Hrany sú komplikovanejšie. Hrana ak je spojená s jedným vrcholom na jednej strane sveta a s druhým vrcholom, ktorý je na opačnej strane sveta . Takúto hranu nemá vždy zmysel zobrazovať. Najmä keď sme na pozícii v strede sveta, máme obrazovku priblíženú a hrana tadiaľ prechádza, tak takýto údaj – hranu na obrazovke nepotrebujeme. Riešením je teda zobrazovať hranu, len v prípade keď sa zobrazuje aspoň jeden jej vrchol. Toto v dostatočnej miere pomáha Jung-u vysporiadať sa s veľkým počtom hrán a konečnom vykresľovaní prvkov na obrazovku.

Jung ma ešte ďalšie menšie nedostatky vo vykresľovaní prvkov. Bol spravený tak, aby podporoval rôzne variácie zobrazení, rôzne farby a poskytoval širokú paletu možností pri zobrazovaní. Samozrejme to je dobré ale v rámci môjho riešenia kde potrebujem niekedy zobraziť viac vrcholov ako v rôznych variáciách zobrazenia je lepšie upraviť „render“ časť Jung-u a nastaviť si ju podľa vlastných potrieb.

4.5.2 Čiastočný fyzický návrh

V rámci práce nechceme riešiť fyzicky návrh do detailov ale pri testovaní prototypu sme sa stretli s problémom s dostatkom pamäťových zdrojov. Preto v ďalšej časti spomenieme, niektoré nápady a riešenia pre dostatok pamäte.

[28] Množina vrcholov (v rámci jednej oblasti) v poli nebude zoradená, preto O(1). Zoradená nebude aby sa zachovala zložitosť O(1) pri pridávaní prvkov do poľa.

Každý vrchol má kľuč, hodnotu ako reťazec a ohodnotenie ako číslo . Mnoho vrcholov ma ale rovnaký kľuč, sú tam redundantné údaje. Preto má zmysel vyriešiť redundanciu, a to vytvorením množiny jedinečných kľúčov a tam, kde sa kľuč používa, použiť odkaz. Ak sa zamyslíme nad tým podrobnejšie, v Jave je odkaz referencia, a tá zaberá 4 byty, naopak napríklad ,,char" 2 byty.[29] Vo vzorke Enron corpus graf sa nachádzajú stovky jedinečných kľúčov. Preto má zmysel použiť ,,char" ako index na danú množinu kľúčov.

Ďalším atribútom vrcholu je "hodnota", tá je opäť reprezentovaná reťazcom. Veľa vrcholov ma podobné hodnoty ale nedochádza tu k redundancii. Reťazec v rámci Javy zaberá 8 bytov ako objekt, 4 byty si berie dĺžka reťazca, plus ďalšie byty navyše za každý jeden znak. Výsledkom je obrovské množstvo údajov pre každý vrchol. Podľa mojich testov takéto množstvo údajov na 1 milión vrcholov skoro nedokážeme uchovať v pamäti počítača.[30] Riešením je buď zmena nastavení Java virtuálneho stroja alebo moje vlastné riešenie za pomoci pamäťovo na mapovaných súborov. Takže primárne každý vrchol bude identifikovateľný pomocou kľúča a indexu, teda do ktorej kategórie patrí a indexom, ktorý hovorí o pozícii v kategórii.

Tretí atribút, ohodnotenie vrcholu sa dá reprezentovať primitívnou hodnotou "int", v rámci návrhu nemôžeme použiť objekt "Integer", pretože ten zaberá o 8bytov viac.

Pri výbere reprezentácie atribútov hra veľkú rolu aj zaokrúhľovanie bytov, po viacerých testoch a skúšaní možnosti ako najlepší návrh pre vrchol je:

```
char  key;
int   pos   = -1;
int[] incoming;
int[] outcoming;
float x;
float y;
```

Ako dodatočné atribúty Jung vyžaduje pozíciu vo svete to je ,,x, y", kde opäť šetríme použitím float-u pred double. A tiež potrebujeme zoznam susedov, to je ,,incoming" a ,,outcoming". Abstraktne vysoké zoznamy údajov v Java sú ,,set", ,,HashMapa" či ,,linkedlist", tie však majú vysokú dodatočnú pamäť na každý element a najjednoduchšie je použiť jednoduché pole čísel.

V rámci hrán potrebujeme 2 odkazy na vrcholy, opäť je výhodnejšie použiť 2x ,,int" odkaz ako referenciu. Tretím atribútom hrany je ohodnotenie, to môžeme tiež reprezentovať primitívnom typom ,,int".

4.5.3 Pamäťovo namapované súbory

V rámci neskorších verzii Javy sa zistilo, že základne operácie vstupu a výstupu a práca so súbormi je pomalá. Preto neskôr vývojári vytvorili nový vstupný /výstupný (input / output), skrátene **NIO** balík, ktorý nám umožňuje namapovať existujúce súbory do pamäte a pracovať tak s väčšou pamäťou. Takáto práca s novou pamäťou je dostatočné efektívna, rýchla a pamäť sa neustále vymieňa (swapuje). Na maximalizáciu výkonnosti určite využijeme všetky črty java.nio balíka.[17] Takže hoci fyzicky sa v systéme nachádza málo pamäte, dokážeme vytvoriť novú pamäť s niekoľkými gigabajtmi, čo je úplné skvele. Týmto sa vyrieši problém s uchovávaním nadmerne veľkých dát.

---

[29] Veľkosti boli overené na 32bit systéme vlastným testom.

[30] Samozrejme pri štandardnom nastavení Javy, ktorá má približné 200 – 300 MB pamäťové limity na aplikáciu.

4.5.4 Grafové databázy

Na uchovávanie veľkých grafových dát môžeme použiť grafovú databázu. Takéto databázy sú už prispôsobené takým dátam, ich výkonnosť aj efektívnosť je maximalizovaná. Známa grafová databáza je napríklad neo4j[31]. Často krát takéto databázy využívajú pamäťovo na mapované súbory na uchovávanie dát. Avšak ak by sme chceli použiť grafovú databázu, museli by sme pripraviť aplikáciu na komunikáciu s ňou. Pri použití aplikácie na inom stroji by sme museli aplikáciu prekonfigurovať. To nehovorím ak chceme presúvať dáta, museli by sme presunúť aj celú databázu . Ak by sme chceli vizualizovať nový graf, museli by sme ho pridať najprv do spomínanej databázy . Výsledkom je že grafová databáza ma výhody, je ju dobre použiť. Ale nie pri tomto zadaní kde chceme mať štandardnú aplikáciu bez závislosti od iných externých prvkov.

4.6 Načítanie súboru a jeho spracovanie

Načítanie súboru a ďalšie spracovanie by malo byť jednoduché ale nie je. Keďže naša vzorka Enron corpus graf ma veľkosť 2,6G, nemôže isť o štandardné načítanie. Cieľom nie je načítavanie veľkých dátových súborov. Avšak rýchlosť načítania a maximálna možná kapacita ma vplyv aj na menšie dátové súbory. Spracovanie dát musí byť rýchle, najlepšie paralelne. Paralelne spracovanie využijeme najmä pri hranách, kde hľadáme ku ktorým vrcholom je hrana priradená,. Budeme vytvárať pre vrchol zoznam susedov a v hrane udržiavať odkaz na vrchol teda musíme nájsť index vrcholu.

V rámci spracovania dát sa údaje budú postupne pridávať do grafu, do Quad stromu a do šablóny. Algoritmus na rozloženie vrcholov priestore ma veľký vplyv na rýchlosť spracovania súboru.

Cele načítavanie bude pekne zobrazené v užívateľskom dialógu, kde zobrazím aktuálnu pozíciu spracovania. Samozrejme pri spracovaní sa budú kontrolovať vstupne údaje, čí veľkosť dát neprevyšuje kapacitu premenných alebo či ich počet nie je nadmerný a v poslednom rade sa bude kontrolovať či sú dáta korektne a hrany neodkazujú na neexistujúce vrcholy.

Pred načítaním súboru bude zobrazené informatívne okno v aplikácii, s možnosťou vybrať si nejaký súbor, ktorý sa následné spracuje. Až po spracovaní bude možné si pozrieť informačnú sieť.

4.7 Možnosti úpravy vizualizácie

Používateľ by mal mať možnosť nastaviť si čo sa bude zobrazovať a čo nie. Preto plánujem pridať panel a menu, kde si používateľ bude môcť nastaviť rôzne parametre aplikácie. Určite bude mať základnú možnosť vybrať si, ktoré kľúče (kľúč vrcholu) sa zobrazia, ďalej aký formát štítkov vrcholov sa zobrazí. Napríklad samotný kľúč, kľúč a hodnota, kľúč a hodnota s ohodnotenim vrchola atď. V rámci ďalších nastavení si napríklad bude môcť zapnúť a vypnúť automatické filtrovanie vrcholov podľa priblíženia, či auto zabaľovanie, automatické zvýrazňovanie susedov alebo tvar vrcholu podľa počtu hrán.

Vďaka posuvníku si bude môcť zvoliť určitú hodnotu pre ,,treeshold“, automaticky filter podľa ohodnotenia vrcholu, automaticky filter podľa uhlu. Všetky hodnoty v posuvníku budú v určitom intervale, to aby používateľ nemohol nastaviť nepovolené hodnoty.

[31] http://www.neo4j.org

V aplikácii možno pribudnú ďalšie nastavenia pre vizualizáciu hrán a pre ďalšie spôsoby zobrazenia vrcholov. Spôsob rozmiestnenia vrcholov bude pravdepodobne trvalé, používateľ ho nebude môcť meniť.

4.8 Diagramy štruktúry

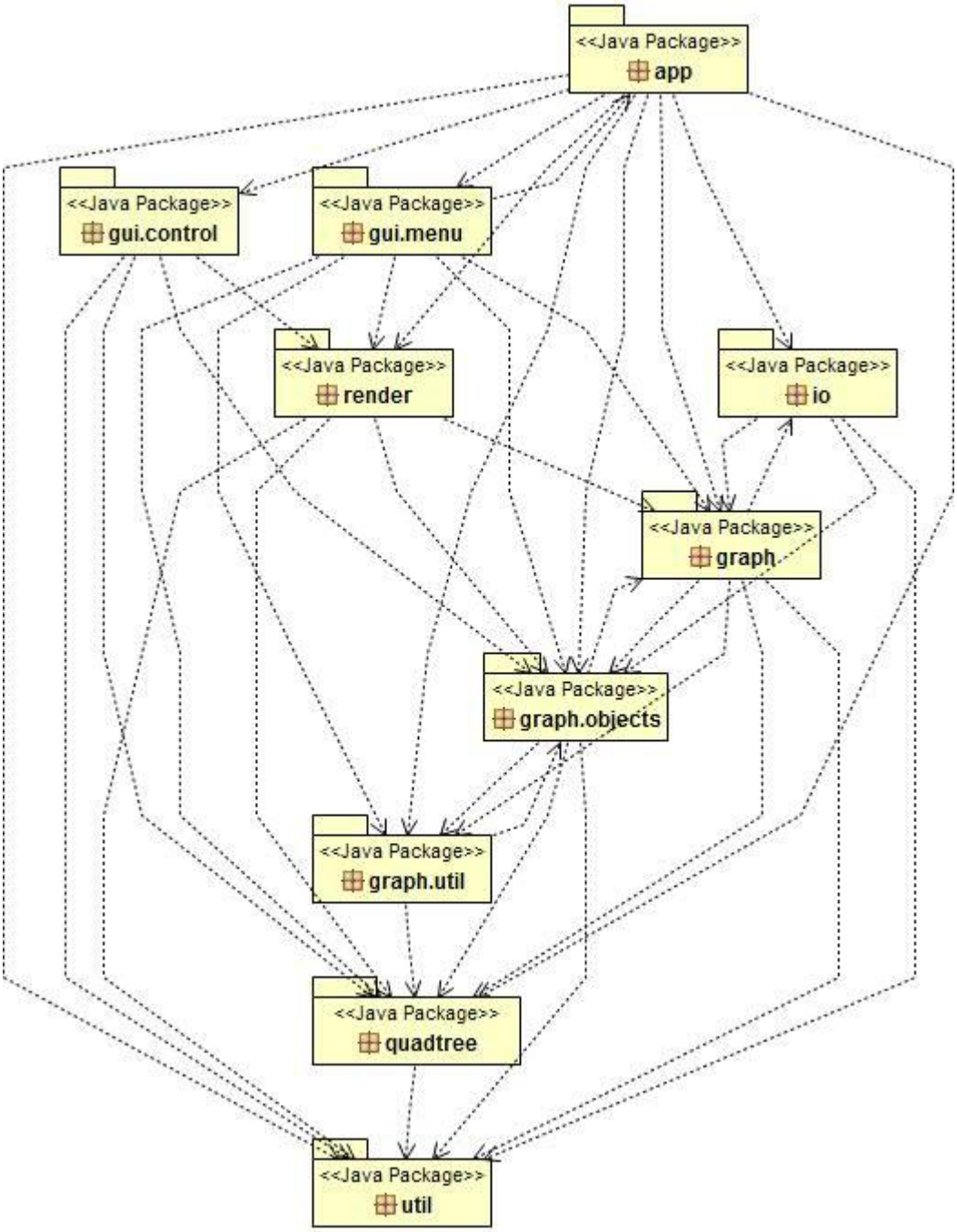
Štruktúra systému bude komplikovanejšia, rozdelíme ju pekne do základných objektov. Teda vznikne objekt vrcholu - ,,Vertex", objekt hrany bude reprezentovaný unikátnym číslom. Skupina vrcholov bude v dátovej štruktúre ArrayList, ktorá ma najmenšie dodatočné pamäťové nároky na každý element. Túto skupinu nazveme ,,GroupInfo", ktorá bude uchovávať ďalšie atribúty skupiny. Napríklad unikátne číslo, kľúč vo forme reťazca, farbu skupiny. Do tejto skupiny pôjdu vrcholy len s rovnakým kľúčom. Všetky tieto skupiny budú združené vo ,,VerticesPool". V konečnom stave cez ,,VerticesPool" sa budeme môcť dostať k celej množine vrcholov, či k určitej skupine. Za pomoci čísla skupiny (teda kľúča) a potom podľa indexu, indexu v skupine sa bude možné dostať priamo ku vrcholu. Všetko sa deje teda v zložitosti O(1), prvky sú podelene do časti, všetky skupiny nemusíme mať v pamäti, môžeme pristupovať k prvkom paralelne. Výsledok tejto hierarchie je veľmi prínosný.

Ďalší objekt je „GraphHolder", bude v sebe uchovávať celý graf. Graf rozdelíme na „VerticesPool" teda vrcholy a ,,Edges" teda hrany. ,,Edges" bude udržiavať zoznam všetkých vrcholov, tento krát priamo. Pretože prakticky hrany nemáme ako kategorizovať. Ku hrane sa dostaneme priamo cez index číslo, teda každá hrana bude mať vlastný index - unikátne číslo.

Všetky prvky budú pekne zaradené do balíkov, podľa toho do ktorej hierarchie patria. O týchto objektoch sa dá podrobnejšie písať ale lepšie to vysvetlia diagramy.

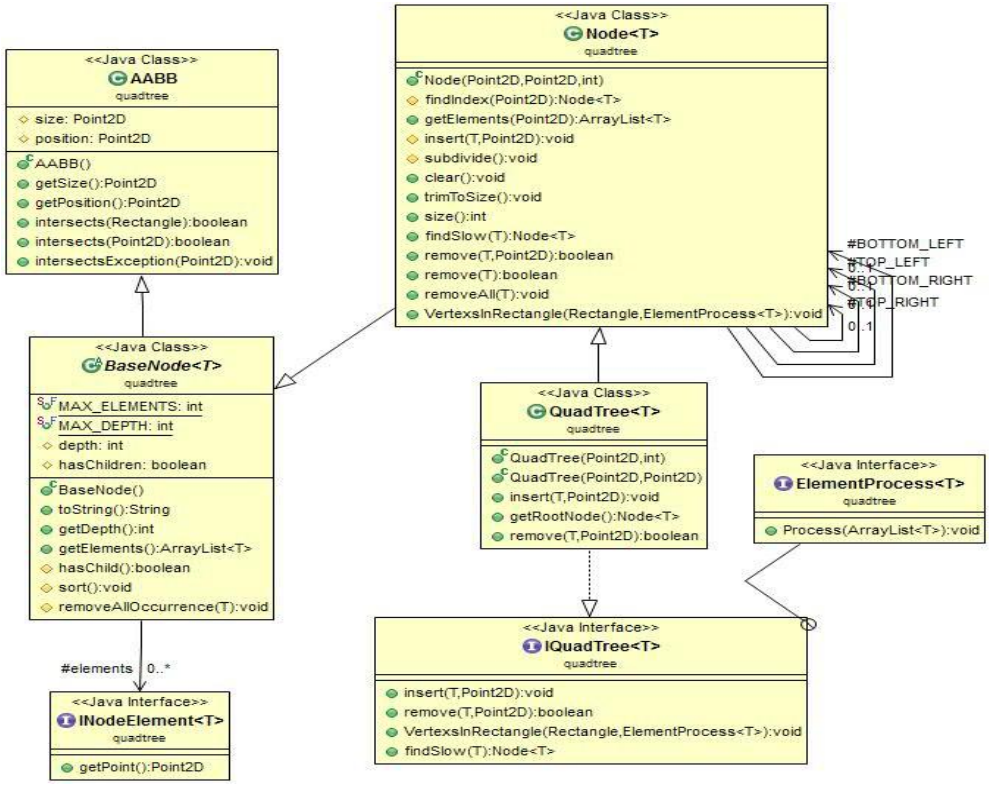
4.8.1 Diagram balíkov

Každý balík predstavuje priestor názvov, predstavuje aj hierarchiu štruktúry. Základnými balíkmi sú ,,app", ,,util", ,,io". Balík ,,Gui" sa stará o užívateľské prostredie, to delíme na menu aplikácie a ostatné ovládacie prvky („control"). O ostatné zobrazovacie časti sa stará ,,render" balík. ,,Quadtree" je balík, určený pre dátovú štruktúru Quad strom, ktorá bola spomenutá vyššie. Celkovo je v diagrame vidieť veľa závislosti, je to preto lebo balíky sú naozaj závislé od seba. Balík v dolnej časti grafu ,,util" sa používa v každom balíku, pretože bude obsahovať rôzne utilitky.

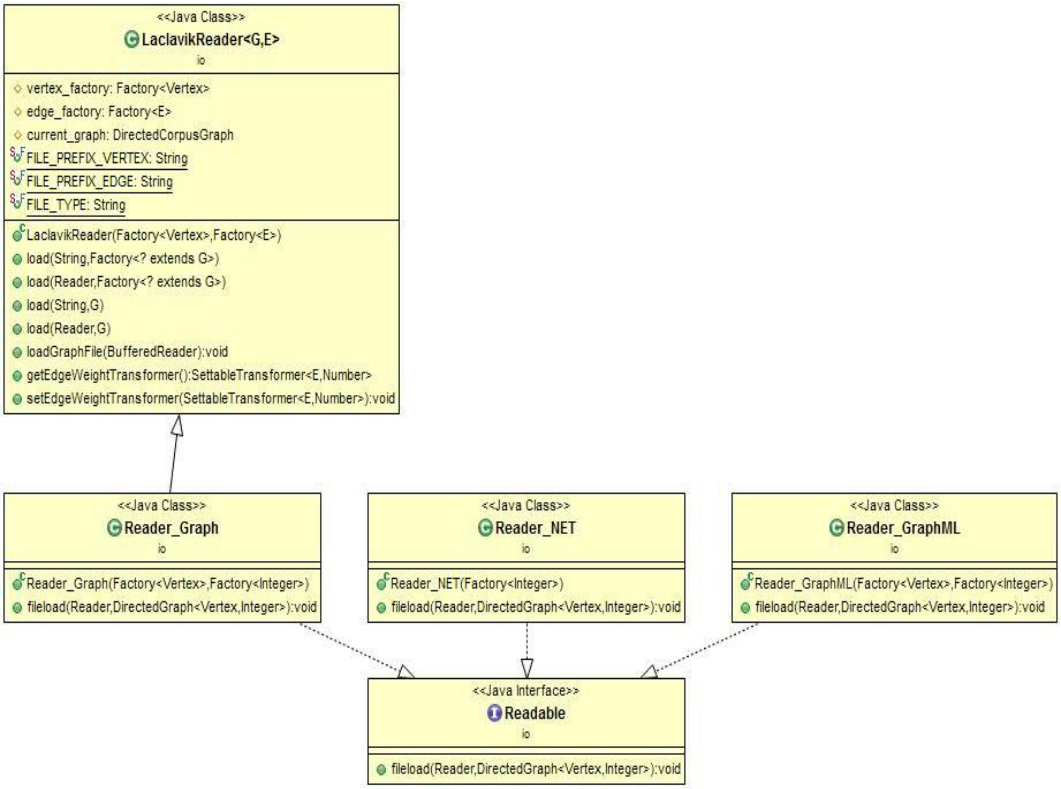


Obr. 14. : Diagram balíkov.

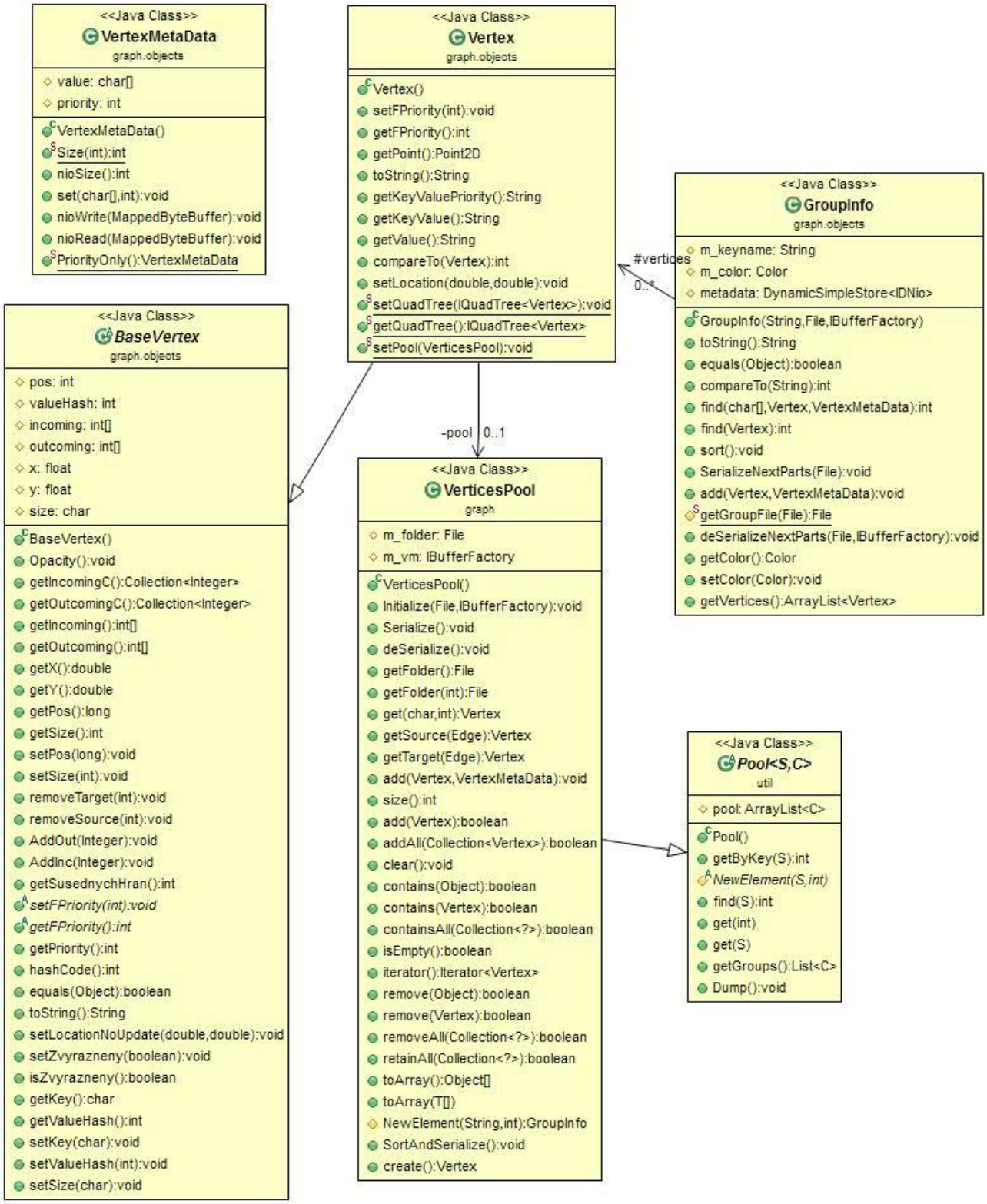
4.8.2 Class diagram



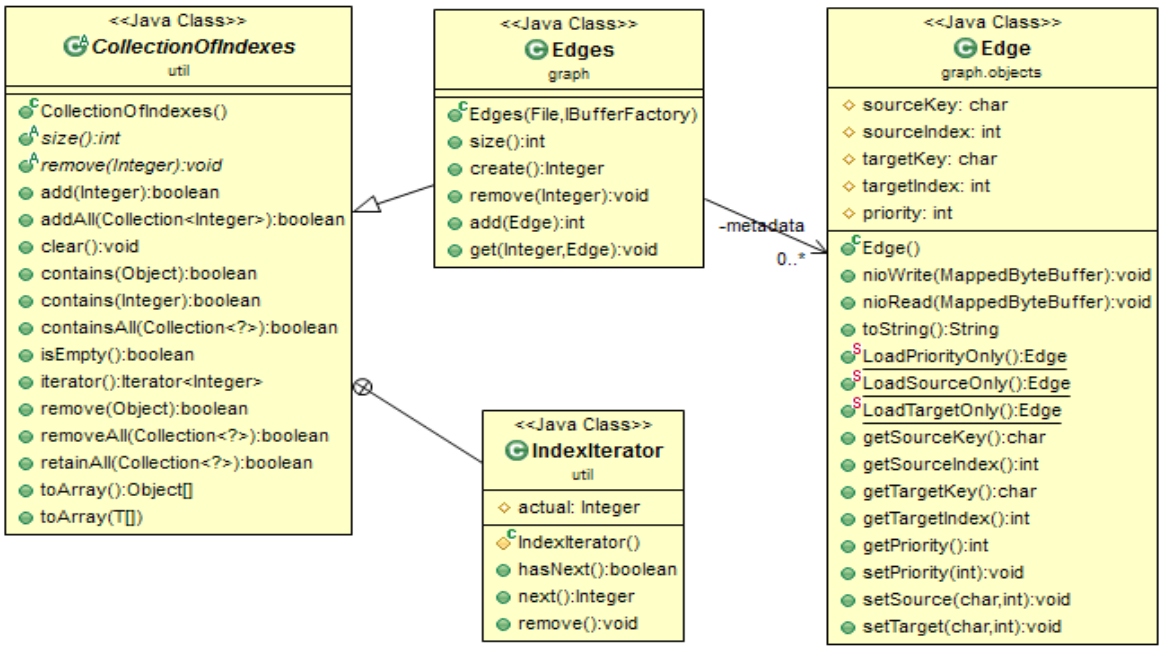
Obr. 15 : Diagram tried pre Quad Strom.



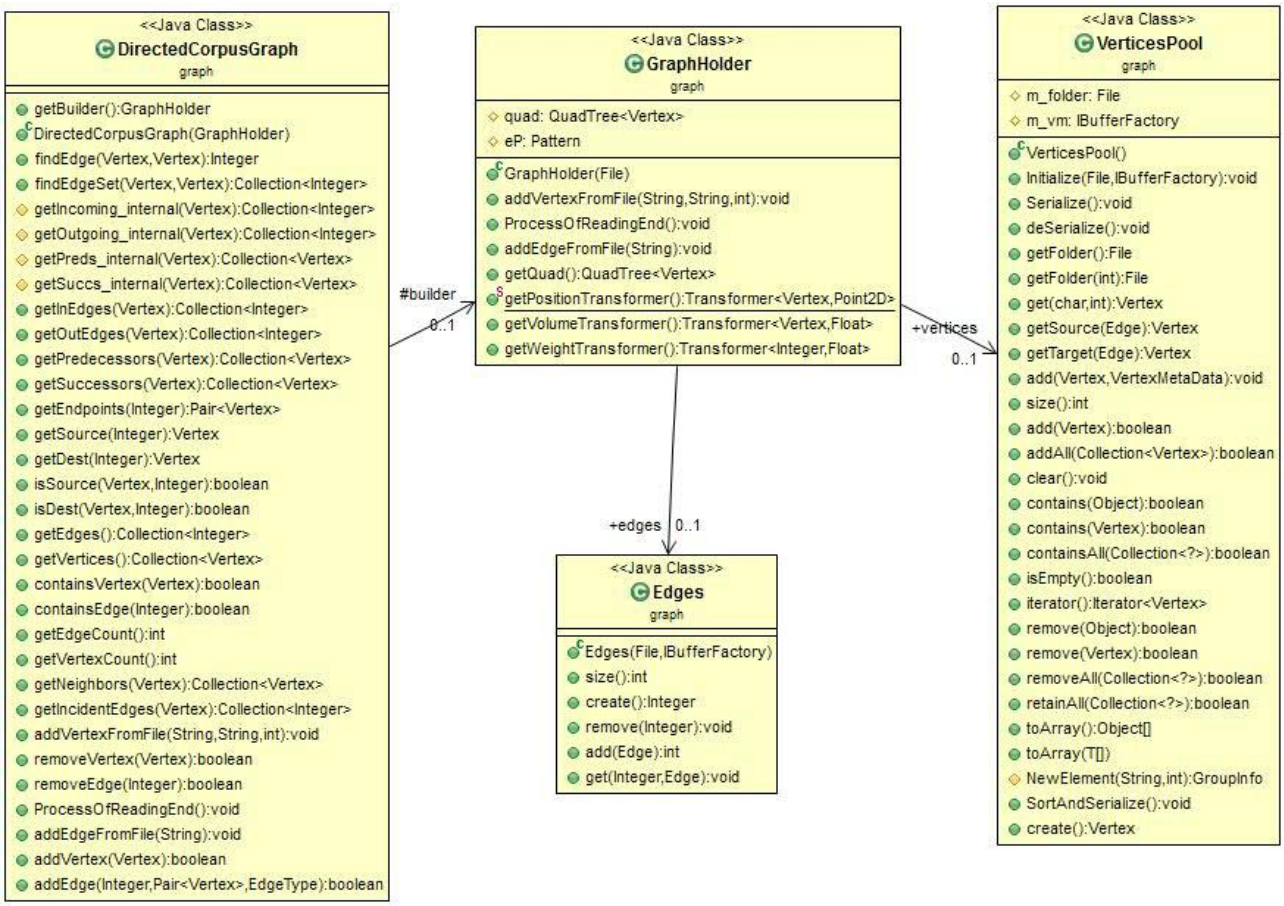
Obr. 16. : Diagram tried pre Načítavanie súborov.



Obr. 17. : Diagram tried, ktorý zobrazuje vzťah vrcholu so skupinou vrcholov (GroupInfo) a všetkými vrcholmi (VerticesPool).



Obr. 18. : Diagram tried zobrazuje vzťah hrany (Edge) so všetkými hranami (Edges).

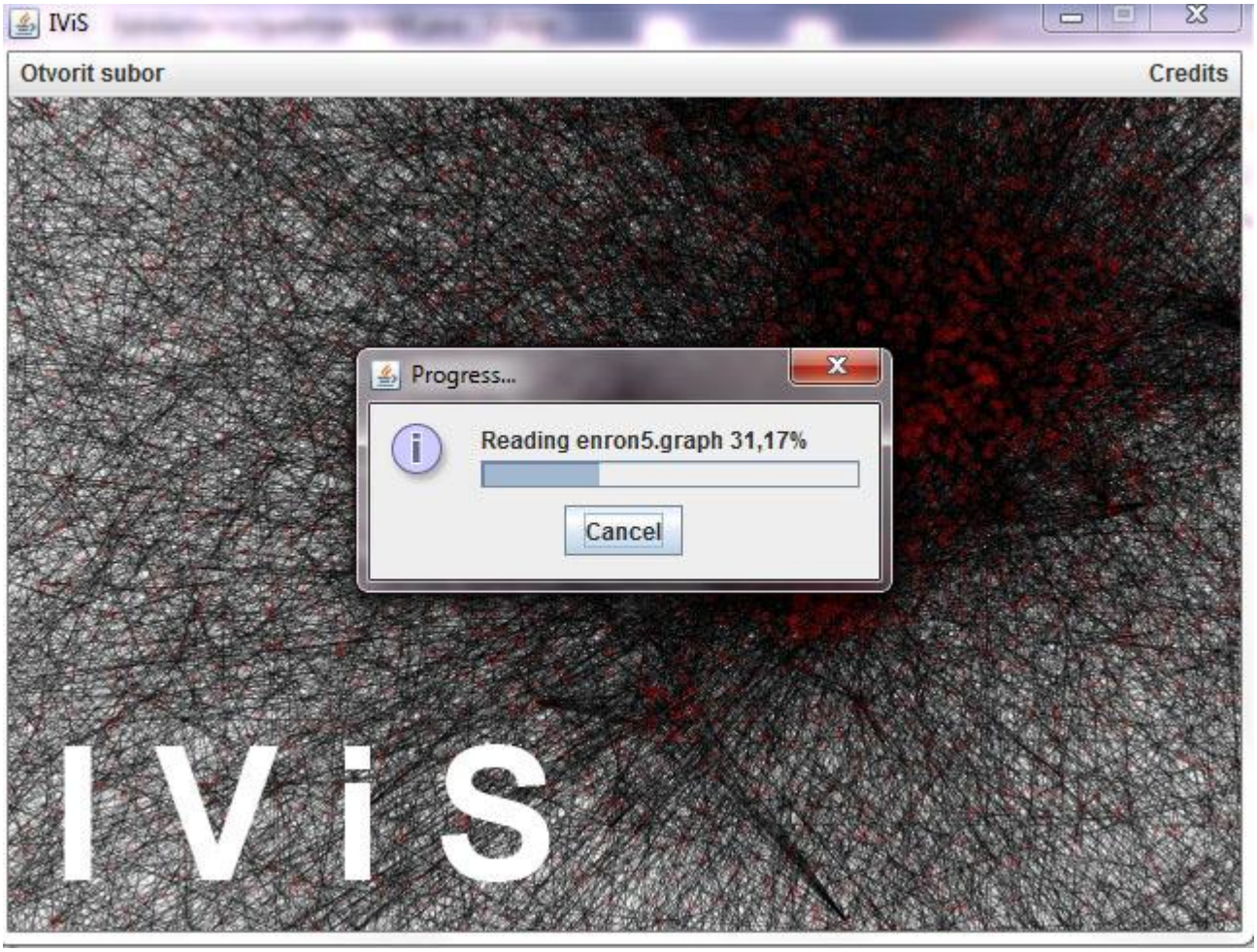


Obr. 19. : Diagram tried zobrazuje vzťah, kde hrany (Edges) a vrcholy (VerticesPool) sú uložené v GraphHolder. Všetko to je zabalené v DirectedCorpusGraph, teda celkový graph.

4.9 Návrh užívateľského prostredia

Používateľské prostredie bude vyzerať ako typická Java aplikácia s použitím Swing-u[32]. V okne aplikácie bude horné menu, kde bude tlačidlo pre otvorenie súboru a druhé tlačidlo pre otvorenie okna s popisom aplikácie. V strede okna bude pozadie a na ňom text „IVIS“, teda názov aplikácie.

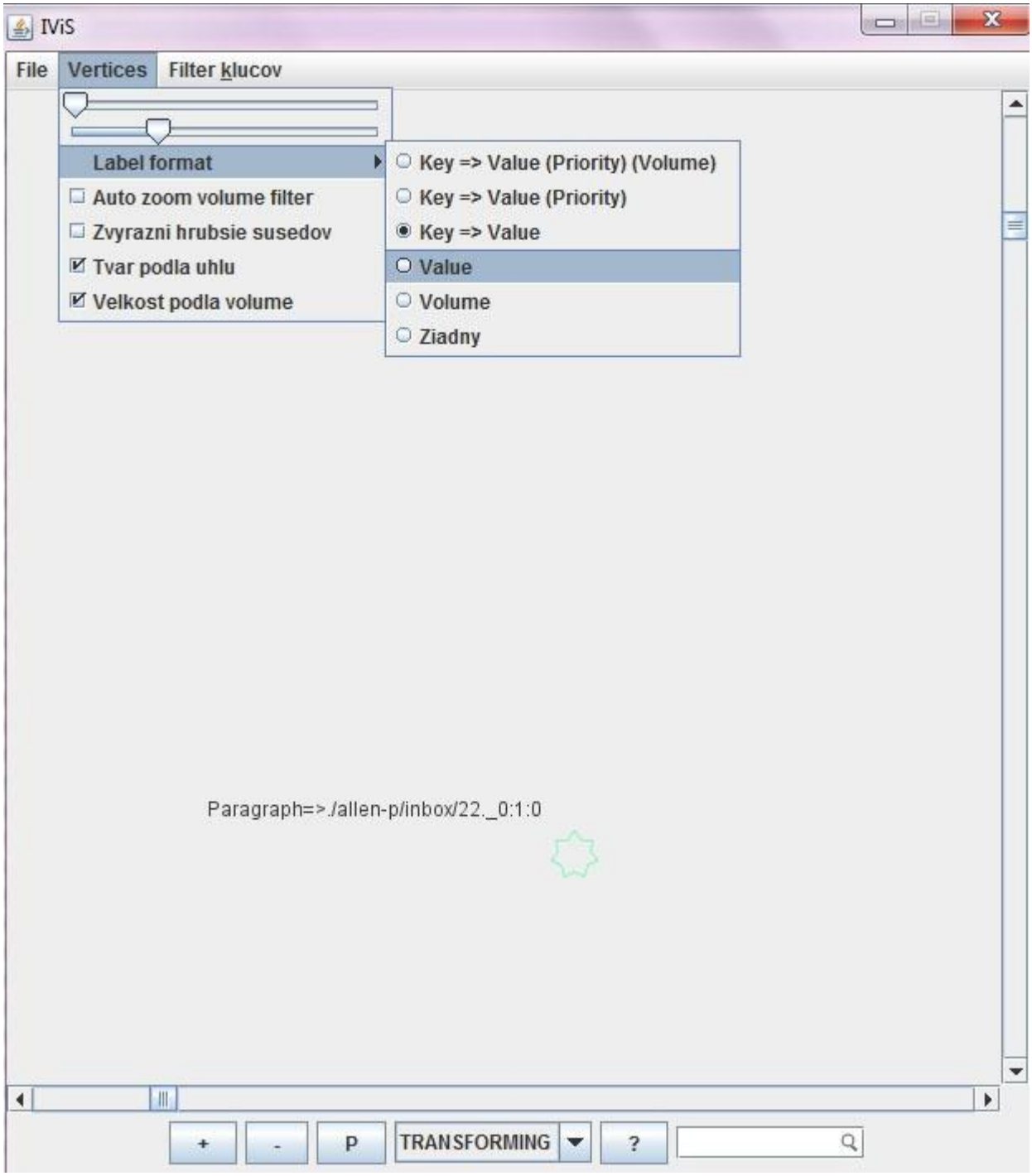
Pred otvorením súboru sa otvorí dialóg, kde sa zobrazí zoznam súborov a používateľ si bude môcť jeden vybrať. Následne sa zobrazí okno kde sa ukáže stav spracovania súboru a možnosť zrušiť spracovanie.



Obr. 20. : Úvodná obrazovka prototypu aplikácie spolu s oknom o stave spracovania.

Používateľské prostredie po spracovaní súboru je znázornené na obrázku č. 21. V prostredí sa zmenilo horné menu, ktoré už poskytuje kontrolu nad vizualizáciou grafu. Sú tam všetky spomínané nastavenia, ktoré by aplikácia mala minimálne obsahovať. Veľkosť filtrovania podľa hodnoty a podľa uhlu vrcholov sa nastavuje pomocou posuvníka.

[32] http://en.wikipedia.org/wiki/Swing_(Java)



Obr. 21. : Používateľské prostredie pre vizualizáciu grafu.
V grafe je zobrazený len jeden vrchol.

V dolnej časti budú tlačidlá pre približovanie a odďaľovanie. Jednoduché tlačidlo pre odfotenie obrazovky a následne uloženie obrazu do súboru. Vo vysúvacom menu budú položky, cez ktoré sa nastaný mód používania myši. Napríklad pohyb po grafe alebo presúvanie elementov. Ďalšie tlačidlo bude mať funkciu pomoci, kde sa po kliku zobrazí návod a klávesové skratky. Poslednou časťou bude vstup pre filtrovanie vrcholov podľa hodnoty (popisu) vrcholu.

Obr. 22. : Otvorené menu „Filter kľúčov“.
V zozname sú všetky kategórie vrcholov farebne odlíšene.
V zátvorke je počet prvkov v jednej kategórii.

5. Vývoj

Počas implementácie programu nastalo mnoho rôznych problémov, ktorých riešenie malo za následok malé zmeny v diagrame tried, preto som neskôr finálne diagramy tried poskytol v prílohe.

Išlo najmä o úpravy Vertex triedy, kde som bol nútený pridať ďalšie atribúty napríklad informácie o viditeľnosti vrcholu a či je daný vrchol zabalený. Postupným pridávaním ďalších čŕt sa program nabaľoval a bolo ho potrebne viackrát refaktorovať , takže mnohé veci sa premiestnili a premenovali.

5.1 Problémy zobrazenia veľkého počtu entít

Na začiatku práce sme mali zadefinované vysoké požiadavky ako zobraziť veľký počet entít na obrazovke. Celá logika architektúry, použitie dátovej štruktúry quad strom, pamäťovo mapované súbory a paralelne distribuovanie problému, to všetko napomohlo tomu aby som to splnil.

Avšak reálne nie je možné v Jave zobraziť niekoľko 100 tisíc entít, respektíve aplikácia sa pomaličky zasekáva a práca už nie je dostatočné interaktívna. Podotýkam že som upravil aj samostatne renderovanie v rámci JUNGu, to pomohlo zlepšiť výkon o ďalších 10%. Ostatné spomalenie nastáva najmä v Java grafickom API, čo už nie je možné upraviť.

V rámci vývoja ďalších nástrojov pre spravovanie vizualizácie sme sa rozhodli pridať takzvanú mini mapu do aplikácie. Daná mini mapa by zobrazovala vrcholy, ich pozíciu a pozíciu používateľa v malom okne. Mini mapa by tak napomohla používateľovi sa lepšie zorientovať v grafe, používateľ by sa mohol aj rýchlejšie pohybovať po obrazovke ak by klikol na mini mapu. Takýto nastroj sme postupne začali implementovať, avšak narazili sme na závažný problém – akú zvoliť šírku a výšku okna pre mini mapu. Daná mini mapa by bola súčasťou hlavnej obrazovky a tak by bránila vo výhľade na vrcholy. Nemôže byť teda príliš veľká. Zároveň je to ďalšia informácie navyše a zhoršila by tak viditeľnosť iných informácií.

Pri malej veľkosti mini mapy som sa stretol s ďalším problémom. Každý jeden vrchol som zakreslil ako jeden pixel a aj napriek tomu hustota grafu v priestore bola tak veľká, že cela mini mapa bola vyplnená a zafarbená jednou farbou (farbou podľa kľúča vrcholu). Takže v konečnom dôsledku sme sa rozhodli mini mapu vypnúť.

6. Užívateľské prostredie a jeho kvalita

Na to aby sme dokázali ohodnotiť užívateľské prostredie a jeho kvalitu, rozhodli som sa využiť 10 jakobsonovich pravidiel. Tieto pravidla sú podrobne opísané v *Usability Engineering*[14] , kde Jakob Nielsen hovorí, že pomocou týchto pravidiel sa maximalizuje kvalita užívateľského prostredia a teda aj produktu. [14]Takže v nasledujúcej kapitole zhrnieme pravidla a napíšeme kedy, kde a ako sme ich použili. V druhej časti v krátkosti opisujeme niektoré vybrané use case modely, pripájam k nim aj heuristické ohodnotenie koľko krokov a myslenia vyžadujú časté operácie.

6.1 Jakobsonové pravidla

6.1.1 Viditeľnosť stavu systému

V rámci tohto pravidla by sme mali používateľa informovať o stave systému.[14] Preto sme do užívateľského prostredia zakomponoval status bar – miesto kde sa zobrazujú hlášky o stave systému. Status bar sa nachádza na obr. 24 v bode 6. Je dobre viditeľný a pri zmene stavu sa animuje aby si to používateľ všimol. Ľahko používateľa informujem aby si zapol a vypol nejaký filter podľa potreby. Pripadne ho informujem o dĺžke najkratšej cesty, ak si ju vybral.

6.1.2 Vzťah medzi systémom a skutočným svetom

Pravidlo hovorí o použití jazyka, ktoré je blízko jazyku používateľa.[14] Zároveň v kapitole „5.1 Simple and Natural Dialogue“ Jakob Nielsen vraví že mame: „udržať jednoduché a prirodzene dialógy“.[14] Preto som obrazovky rozdelil do dvoch časti, ktoré opisujem v Príloha B - Užívateľská príručka. A Dialógy sú maximálne zjednodušené. Zároveň náš typicky používateľ by mal ovládať termíny ako sú vrchol, hrana a pod. Žiadne iné špeciálne termíny sme nepoužili.

6.1.3 Napovedá a dokumentácia

Aby sme splnili aj toto pravidlo, každému nástroju a každému tlačidlu sme pridali nápovedu v podobe popisu. Zároveň sme pridali rýchlu nápovedu, obr. 24 bod 10, ktorá sa spustí po stlačení tlačidla. V tejto rýchlej nápovede je krátky návod k základnému ovládaniu celého programu.

6.1.4 Ostatné pravidla

Všetky ostatné pravidlá, ako napríklad minimalizácia zaťaženia krátkodobej pamäti používateľa, konzistencia, klávesové skratky, atď. sú istou formou súčasťou užívateľského prostredia. Avšak tieto pravidlá nie je možne objektívne posúdiť čí sú kvalitne spravené, preto sme pripravili aj use case. Tieto use case modely sme využili aj pri experimente. Kde sme sa pýtali používateľov ako hodnotia kvalitu užívateľského prostredia.

6.2 Use case

Use case legenda

T – používateľ musel myslieť
C – používateľ musel kliknúť
M – používateľ musel spraviť pohyb za pomoci myši

UC 1. Používateľ ide otvoriť graf

Predpokladá sa, že používateľ dokázal spustiť program a je v úvodnej obrazovke programu. Používateľ sa premiestni myšou k ľavému hornému rohu, vyberie otvoriť súbor kliknutím. Následné nájde súbor, klikne otvoriť.

T = 2x C= 3x M= 3x

UC 2. Používateľ ide vypnúť/zapnúť skupinu kľúčov

Predpokladá sa, že používateľ dokázal otvoriť súbor a zobrazuje sa mu graf. Používateľ sa premiestni myšou k hôrnemu menu, vyberie Skupiny vrcholov. Následné posunom v menu vyberie skupinu, ktorú hľadá a klikne na ňu.

T=2x C=2x M=2x

UC 3. Používateľ ide vypnúť/zapnúť Automatický filter podľa priblíženia

Predpokladá sa, že používateľ dokázal otvoriť súbor a zobrazuje sa mu graf. Používateľ sa premiestni myšou k hôrnemu menu, vyberie Vrcholy. Následné posunom vyhľadá Automatický filter podľa priblíženia, klikne na to.

T=2x C=2x M=2x

UC 4. Používateľ ide nastaviť formát popisu vrcholov

Predpokladá sa, že používateľ dokázal otvoriť súbor a zobrazuje sa mu graf. Používateľ sa premiestni myšou k hôrnemu menu, vyberie Vrcholy. Presunie sa na Formát popisu. Následné z menu vyberie formát, ktorý chce .

T=3x C=2x M=3x

UC 5. Používateľ chce prejsť k návodu aplikácii

Predpokladá sa, že používateľ dokázal otvoriť súbor a zobrazuje sa mu graf. Používateľ sa premiestni myšou k dolnému panelu nástrojov, vyberie tlačidlo označené ako **?**.

T=1x C=1x M=1x

6.3 Test case

TC 1. Používateľ chce zistiť informácie o strane z grafu

Používateľ otvorí graf (súbor gorila.graph). Používateľ chce nájsť všetky možné informácie a spojitosti o strane, preto bude hľadať vo všetkých kategóriách. Používateľ sa uistí, že filter podľa počtu susedov je nastavený na najmenšiu hodnotu a automaticky filter podľa priblíženia je vypnutý (nachádza sa v bode 2 - Vizualizačná obrazovka). Používateľ prejde do dolného pravého rohu k vyhľadávaniu (Vizualizačná obrazovka bod 12), kde napíše regulárny výraz „.*NÁZOVSTRANY.*“[33] .

V obrazovke sa mu zobrazia viaceré vrcholy (teda fakty o strane smer), používateľa však zaujíma najväčší zdroj informácii takže si oddiali obrazovku. Nájde vrchol ktorý je jasne najväčší alebo ma jasne najviac hrán a priblíži sa obrazovkou k nemu. Následné pravým kliknutím a po výbere prejsť k susedovi zisti, že strana je spojovaná s údajmi ako sú MEDIA, SIS, STV, SSE. Používateľ môže zistiť ďalšie údaje o strane, ak sa pozrie aj na iné vrcholy po filtrácii.

TC 2. Používateľ chce zistiť informácie o Osobe

Predpokladá sa, že používateľ vie že je to osoba a bude v kategórii Person. Používateľ otvorí graf (súbor gorila.graph). Používateľ v ,,Skupine vrcholov“ odškrtne všetky kategórie okrem ,,Person“. Používateľ chce mat zobrazené všetky údaje preto v časti ,,Vrcholy“, ,,filter podľa počtu susedov“ nastaví na najnižšiu hodnotu a odškrtne „automatický filter podľa priblíženia“.

Používateľ do vyhľadávania zadá vyraz „.*MENO.*“[34]. Postupne sa prekliká (pomocou šípky vpravo – Vizualizačná obrazovka bod 14) až k vrcholu Person=>MENO. Následne zistí, že vrchol má príliš málo hrán, teda susedov a teda vstupne dáta neposkytujú dostatočnú informovanosť o tejto osobe.

TC 3. Používateľ chce zistiť prepojenie medzi osobou a stranou.

Používateľ si vyhľadá vrchol osoby a vrchol strany, napríklad pomocou spomínaného vyhľadania. Používateľ sa prepne do módu označovania a označí obidva vrcholy. Následné vyberie najkratšiu cestu od prvého k druhému vrcholu a zisti, že žiadne prepojenie neexistuje, teda žiadne spojenie medzi nimi nie je v rámci grafu Gorila.

[33] Namiesto NÁZOVSTRANY sa dnu vloží názov strany, o ktorej hľadáme informácie.

[34] Namiesto MENO sa dnu napíše meno osoby, o ktorej hľadáme informácie.

7. Experiment

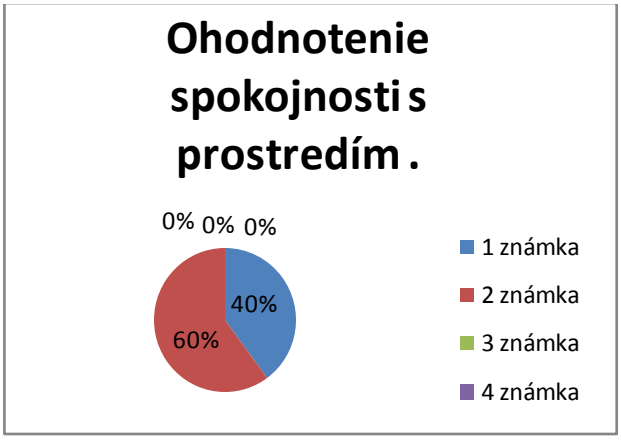
Už v predchádzajúcich častiach som písal, že ohodnotiť vizualizáciu a interakciu sa síce dá ale je tu problém objektivity. Preto predchádzajúce test casi som ponúkol vzorke ľudí, pozoroval som ich a spýtal sa na ich názor. Výsledok je zhrnutý v tabuľke.

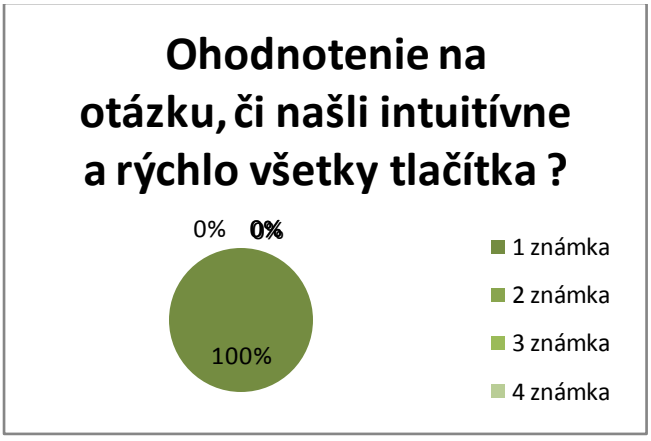
Problém alebo časť kroku test case.	Počet ľudí, ktorý dosiahli očakávaný výsledok.	Počet ľudí, ktorý narazili na problém a tak nedosiahli výsledok.
Otvorenie súboru, kde je uložený graf	9	1[35]
Odškrtnutie automatického filtru podľa priblíženia	5	0
Nastavenie filtrov a formátu popisu Vrcholov	5	0
Práca s menu „Skupiny vrcholov“	5	0
Nájdenie zabudovaného návodu v aplikácii	5	0
Prepnutie ovládacieho módu myši	5	0
Používateľ chce zistiť informácie o strane z grafu	4	1
Používateľ chce zistiť informácie o osobe	4	1
Používateľ chce zistiť prepojenie medzi osobou a stranou	3	2

Pri pozorovaní som si všimol, že pre používateľa mnohé nástroje nie sú štandardne . Preto aby ich používateľ vedel použiť mal by sa ich najprv naštudovať v manuáli. Príkladom na to sú regulárne vyrazy.

Používatelia testovali aplikáciu na menších častiach Enron Corpus grafu, a gorila grafu s rôznou veľkosťou vstupov. Aby som otestoval aj iné formáty, ponúkol som aj jeden súbor s GraphML formátom.

[35] Pri danom probléme používateľ nemohol nájsť grafový súbor, pre jeho slabé skúsenosti s pracou na počítači.





Výsledkom experimentu je, že používatelia nemali problém pracovať s nástrojmi aplikácie. Pri vyhľadávaní informácii, kedy bolo potrebné skombinovať nástroje a nájsť spojenie medzi údajmi nastali malé problémy.

8. Zhodnotenie

Vo finálnom stave naše riešenie predstavuje interaktívnu aplikáciu, s kvalitným užívateľským prostredím. Samozrejmosťou je zobrazovanie vzťahov medzi vrcholmi a vrcholy samotné. Vrcholy zobrazujem cez rôzne tvary a rôzne farby, tak aby to používateľa zaujalo. Pestrosť farieb zase neodpudzuje od toho najdôležitejšieho a to od dát. Používateľ má pocit, že sa mu zobrazili všetky dáta ktoré chcel a tie ktoré o ktoré nemá záujem nevidí. Ovládanie aplikácie je interaktívne a intuitívne. Používateľ po rýchlom prečítaní návodu spozná klávesové skratky a všetky zákutia aplikácie. To, že či aplikácia naozaj všetko spĺňa sa otestovalo na určitej vzorke ľudí, ktorý vyplnili dotazník.

Krátkym dotazníkom som dokázal objektívne zistiť, či aj iní používatelia sú spokojní s aplikáciou, či nemajú problém s prostredím. Zistil som či používatelia dostatočne chápu aplikácii, či rozumejú nástrojom a vedia s nimi interaktívne pracovať.

Z technických cieľov aplikácia dokázala načítať formáty GraphML a gSemSearch. Bolo vytvorených viacero nástrojov a filtrov na filtrovanie údajov. Graf sa spracováva čo najrýchlejšie, pričom program je schopný pracovať dokedy bude mať voľné zdroje a kapacity. Vo vizuálnej obrazovke je pohyb po priestore hladký. Rovnako hladko by malo prebiehať približovanie a oddiaľovanie obrazovky a zabaľovanie vrcholov. Hromadné operácie na grafe, napríklad filtrovanie alebo automatické zabaľovanie vyžaduje istý čas. Pri vzniku chyby pri načítaní, spracovaní alebo nedostatku zdrojov sa aplikácia snaží správne informovať používateľa.

Aplikácia určíte nie je dokonalá a obsahuje rôzne nedostatky. Mnohé jej časti sa mohli ďalej vylepšovať. Ďalšie rôzne nástroje mohli vzniknúť. Hlavným cieľom ale bola vizualizácia informácii a s pomocou nástrojov to uľahčiť. To si myslím, že som splnil. A okrem toho som spomenul aj iné zaujímavé oblasti, ktoré súvisia s vizualizáciou dát.

Bibliografické odkazy

[1] AMET, H. - SAMET, H. The Quadtree and Related Hierarchical Data Structures. In. 1984. no. 2. .

[2] AU, S. - LECKIE, C. Efficient visualization of large routing topologies. In *International Journal of …* [online]. 2004. [cit. 2012-11-20]. Dostupné na internete: <http://onlinelibrary.wiley.com/doi/10.1002/nem.511/abstract>.

[3] BATAGELJ, V. - MRVAR, A. Pajek – Program for Large Network Analysis. In. 1999. s. 1–11. .

[4] DAVID AUBER, P.M. Data Visualization Software | Tulip. In [online]. [cit. 2013-05-05]. Dostupné na internete: <http://tulip.labri.fr/TulipDrupal/>.

[5] EIGLSPERGER, M. - PICH, C. Graph markup language (GraphML). In. 2004. .

[6] ELLIS, G. - DIX, A. A Taxonomy of Clutter Reduction for Information Visualisation. In *IEEE Transactions on Visualization and Computer Graphics* [online]. 2007. Vol. 13, no. 6, s. 1216–1223. Dostupné na internete: <http://eprints.lancs.ac.uk/12933/>.

[7] HEER, J. et al. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* [online]. [s.l.]: ACM, 2005. s. 421–430. Dostupné na internete: <http://portal.acm.org/citation.cfm?id=1055031>.

[8] LACLAVÍK, M. et al. Enron Emails as Graph Data Corpus for Large-scale Graph Querying Experimentation. In *ups.savba.sk* [online]. [cit. 2013-05-05]. Dostupné na internete: <http://ups.savba.sk/~misos/publications/enron_gccp_final.pdf>.

[9] LACLAVÍK, M. - ŠELENG, M. Emails as Graph : Relation Discovery in Email Archive. In. 2012. s. 841–846. .

[10] MAYR, E.W. et al.Eds. *Graph-Theoretic Concepts in Computer Science* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. ISBN 978-3-540-59071-2.

[11] MILLER, G.A. The Magical Number Seven. In SALA, S. DELLAEd. *The Psychological Review* [online]. 1956. Dostupné na internete: <http://www.musanim.com/miller1956/>.

[12] MOJŽIŠ, J. - LACLAVÍK, M. CEGV : Vizualizácia vzťahov v grafoch a sieťach. In . .

[13] MRVAR, A. Networks / Pajek. In [online]. [cit. 2013-05-05]. Dostupné na internete: <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>.

[14] NIELSEN, J. *Usability Engineering* [online]. Ed. J Nielsen. [s.l.]: Morgan Kaufmann, 1993. ISBN 0125184069.

[15] NOACK, A. An energy model for visual graph clustering. In *Graph Drawing* [online]. 2004. [cit. 2012-11-20]. Dostupné na internete: <http://www.springerlink.com/index/bnkwtl1vanl222ln.pdf>.

[16] O'MADADHAIN, J. et al. The JUNG (Java Universal Network/Graph) Framework. In *University of California Irvine California* [online]. 2003. Vol. 03, no. UCI-ICS 03-17, s. 3–17. Dostupné na internete: <http://www.datalab.uci.edu/papers/JUNG_tech_report.html>.

[17] PUGH, W. - SPACCO, J. Mpjava: High-performance message passing in java using java. nio. In *Languages and Compilers for Parallel Computing* [online]. 2004. [cit. 2013-05-05]. Dostupné na internete: <http://link.springer.com/chapter/10.1007/978-3-540-24644-2_21>.

[18] SAMET, H. The quadtree and related hierarchical data structures. In *ACM Computing Surveys (CSUR)* [online]. 1984. no. 2. [cit. 2013-05-05]. Dostupné na internete: <http://dl.acm.org/citation.cfm?id=356930>.

[19] SHAFRANOVICH, Y. [online]. .[s.l.]: IETF, 2005. Dostupné na internete: <http://www.ietf.org/rfc/rfc4180.txt>.

[20] SHNEIDERMAN, B. - ARIS, A. Network visualization by semantic substrates. In *IEEE Transactions on Visualization and Computer Graphics* [online]. 2006. Vol. 12, no. 5, s. 733–740. Dostupné na internete: <http://www.ncbi.nlm.nih.gov/pubmed/17080794>.

Príloha A - Technická dokumentácia

Architektúra programu je organizovaná do balíkov a tried. Balíky už boli opísané v kapitole 4.8.1 Diagram balíkov. Jednotlivé triedy a vzťahy medzi nimi sú dostatočné vysvetlené v diagrame balíkov v kapitole 4.8.2.

V balíku App, sa nachádzajú triedy HomeScreen, VisualizerScreen ktoré dávajú dokopy cele užívateľské prostredie a spolu vytvárajú celú aplikáciu, ktorá je zabalená v App triede(použitý singleton návrhový vzor).

Program obsahuje viacero ďalších tried, ktoré ani neboli spomenuté. To vzhľadom na to, že ich funkcionalita nie je až tak zaujímavá. Naopak medzi zaujímavé určíte patri nio, quadtree, graph.objects balíky.

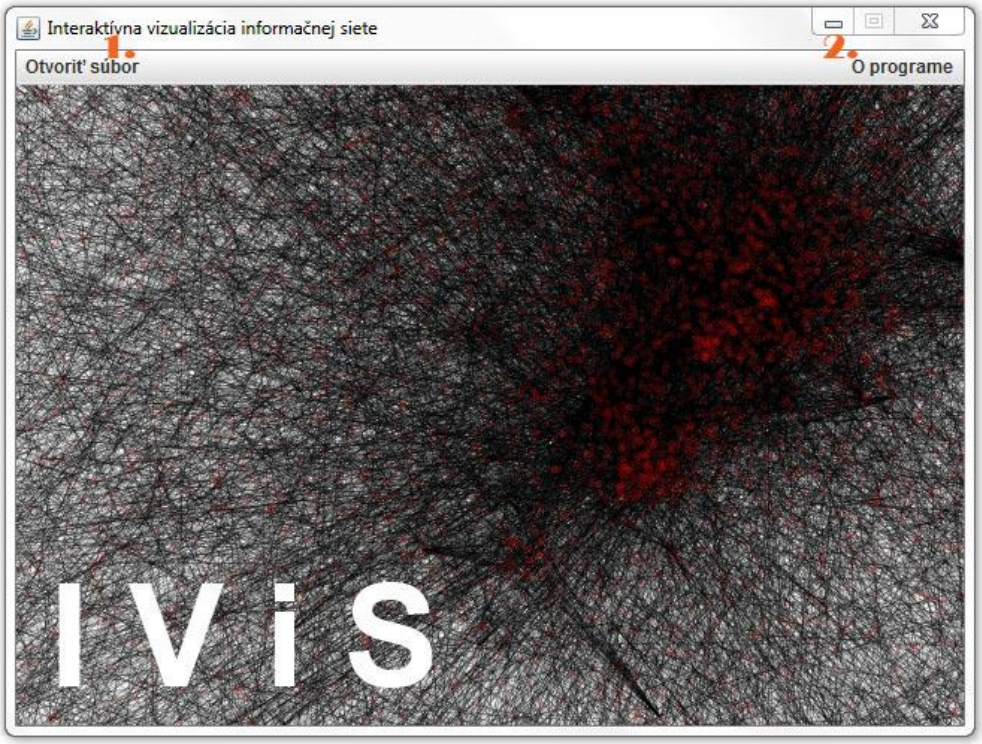
Program tiež používa log4j knižnicu, vytvára tak záznamy počas behu aplikácie a vytvára priestor pre ľahšie debugovanie. Samozrejme záznamy o činnosti sú zapnute len vo vývojovej verzii.

Príloha B - Užívateľská príručka

Pri spustení programu sa objaví úvodná obrazovka, ktorá ma za úlohu odprezentovať program. Z technického hľadiska je to obrazovka, kde si užívateľ môže vybrať súbor - graf, ktorý sa otvorí. Po vybraní súboru sa otvorí už Vizualizačná obrazovka na ktorej je pracovná plocha.

1. Úvodná obrazovka

Na ďalšom obrázku sú znázornené ovládacie prvky úvodnej obrazovky:



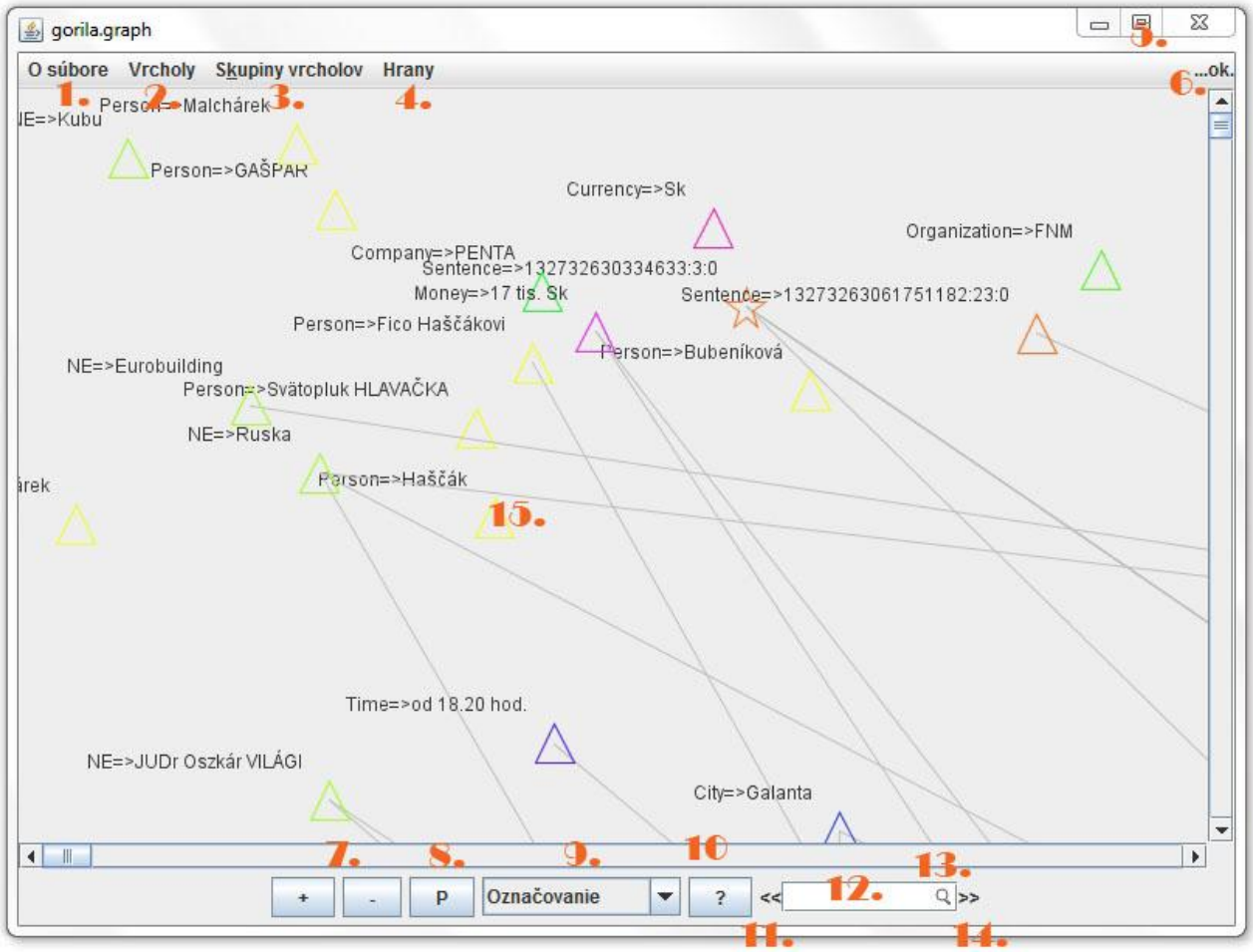
Obr. 23. : Úvodná obrazovka programu.

Ovládacie prvky:

1, **Otvoriť súbor** – po kliknutí sa otvorí menu, kde používateľ si môže súbor vyhľadať a ten sa otvorí

2, **O programe** - po kliknutí sa otvorí okno s informáciami o programe a autorovi

2. Vizualizačná obrazovka



Obr. 24. : Vizualizačná obrazovka programu.

Ovládacie prvky, časť horné menu:

1, **O súbore** – po kliknutí sa otvorí menu s informáciami o danom súbore

2, **Vrcholy** – menu nastavení a filtrov pre zobrazovanie Vrcholov

3, **Skupiny vrcholov** – menu so zoznamom skupín Vrcholov, dané skupiny sa tu zapínajú a vypínajú

4, **Hrany** - menu nastavení a filtrov pre zobrazovanie Hrán

5, **Ikonky** – štandardne ikonky na minimalizovanie / maximalizovanie / ukončenie aplikácie

6, **Status bar** – informuje používateľa o stave aplikácie, dáva mu rôzne tipy na nastavovanie filtrov, ukazuje mu výsledky nájdenia najkratšej cesty

Ovládacie prvky, časť dolný panel nástrojov:

7, **+ / -** – tlačidlá pre približovanie / oddiaľovanie pracovného priestoru

8, **P** – možnosť spraviť fotku aktuálneho zobrazeného pracovné priestoru

9, **Označovanie** – obsahuje menu, kde sú vypísané používateľské módy pre ovládanie myši a kláves

a. Pohyb a rotácia – pomocou myši sa dá pohybovať po grafe, presúvať vrcholy
b. Označovanie – mód len pre označovanie vrcholov. Tzv. sa vytvorí určitá množina vrcholov s ktorou je možne pracovať. Funkcie sú následné aplikovane na celu množinu.

c. Poznámkový mód – mód pre manažovanie poznámok na pracovnej ploche

10, **?** - po kliknutí sa otvorí okno s krátkym návodom ako použiť aplikáciu
11, **<<** - tlačidlo na navrátenie k hľadajúcemu vrcholu
12, **Vstupný formulár** – miesto pre zadanie výrazu, ktorý sa bude vyhľadávať
13, **Lupa** - po kliknutí sa zobrazia len tie výrazy , ktoré spĺňajú vyhľadávací vyraz
14, **>>** - tlačidlo k prejdeniu na ďalší vrchol, ktorý splna hľadaní výraz

Najzaujímavejšia časť **15** je v strede aplikácie, nazýva sa pracovná plocha alebo priestor. Tam sa zobrazujú všetky vrcholy a hrany načítané zo súboru. Používateľ s nimi pracuje. Názov grafu sa nachádza v názve okna aplikácie.

2.1 Použitie hľadania

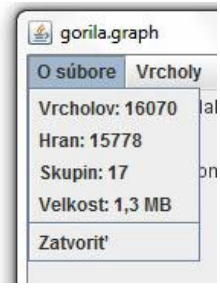
Vstupný formulár (bod 12.) sa dá využiť dvoma spôsobmi :

- Buď ako filter, teda sa zadá text a klikne sa na Lupu (bod 13.) a zobrazia sa len vrcholy, ktoré spĺňajú hľadaný vyraz
- Ako vyhľadávanie vrcholov, zadá sa text (bod 13.), potom sa stláčajú tlačidlá 11 a 14. Obrazovka sa postupne presúva k vrcholom, ktoré hľadaný výraz spĺňajú.

Obidve módy hľadania sa dajú kombinovať a dajú sa takto skryť všetky nepotrebné vrcholy. Pre opätovné zobrazenie všetkých vrcholov, je potrebné zadať prázdny text (do bodu 12.) a stlačiť lupu (bod 130).

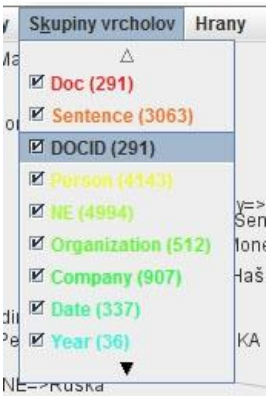
Do vstupného formulára (do bodu 12.) sa vždy zadáva len regulárny vyraz !

2.2 Ďalšie položky vo Vizualizačnej obrazovke

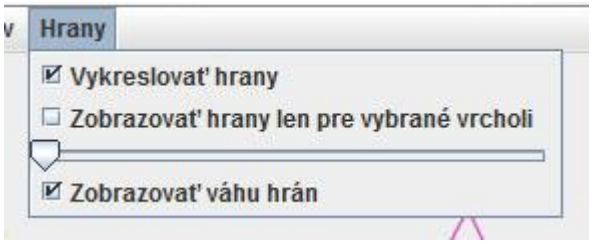


Obr. 25. : O súbore – menu po kliknutí.

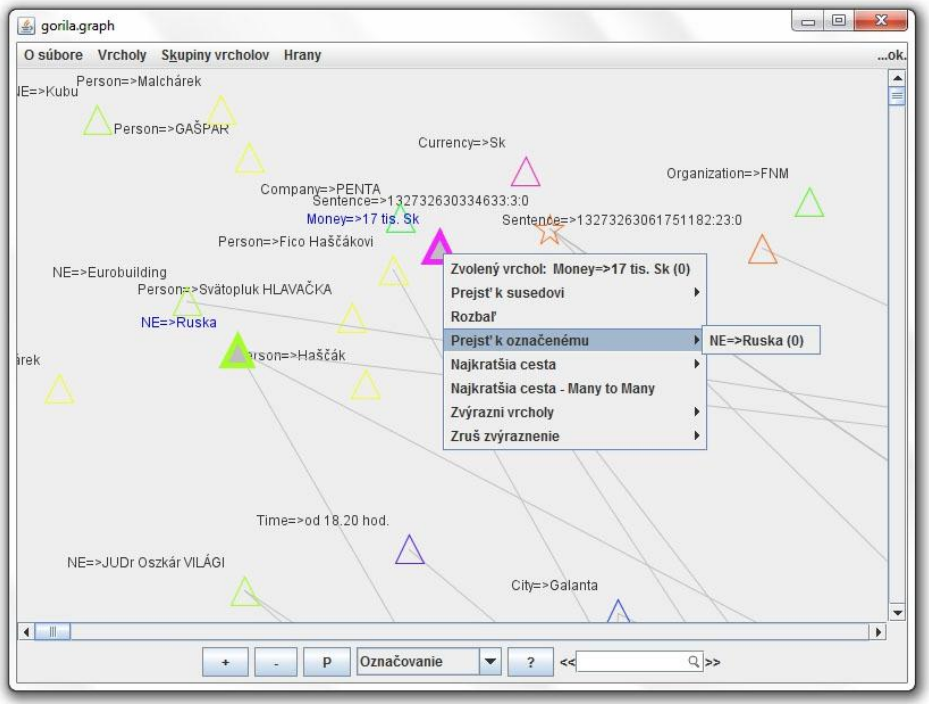
Obr. 26. : Vrcholy – menu po kliknutí.



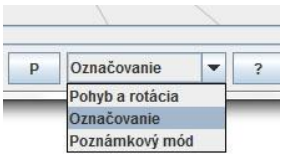
Obr. 27 : Skupiny vrcholov – menu po kliknutí



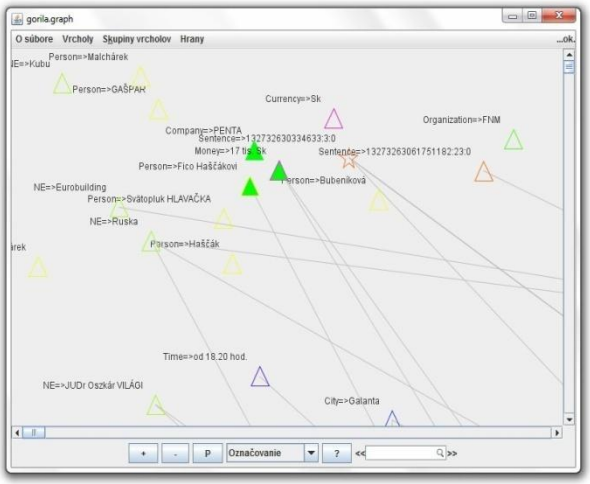
Obr. 28. : Hrany – menu po kliknutí.



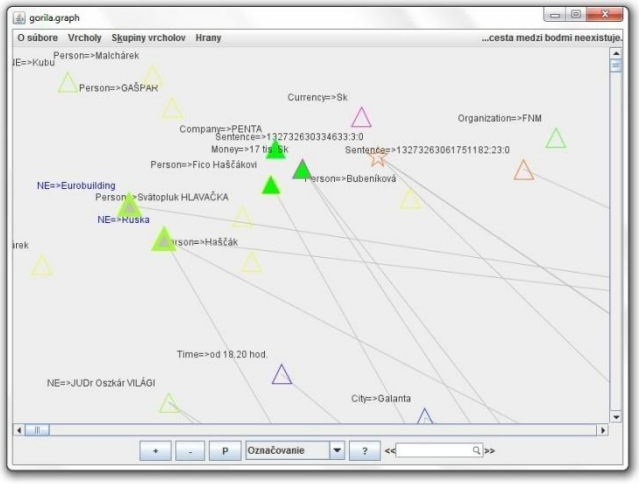
Obr. 29. : Menu po kliknutí pravým tlačidlom myši na vrchol.



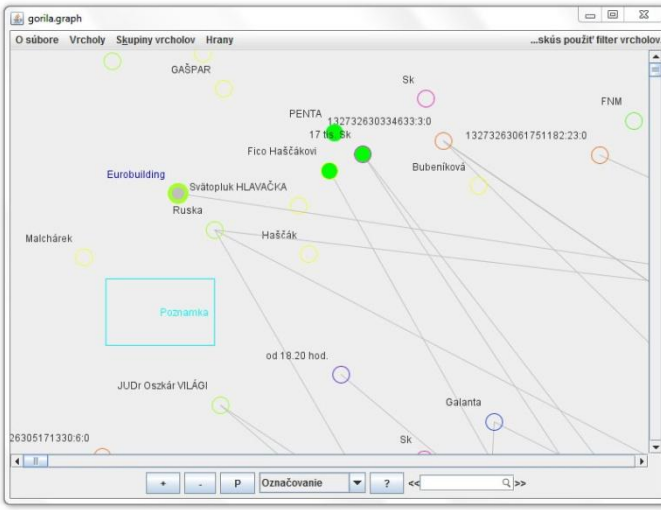
Obr. 30. : Označovanie – menu po kliknutí



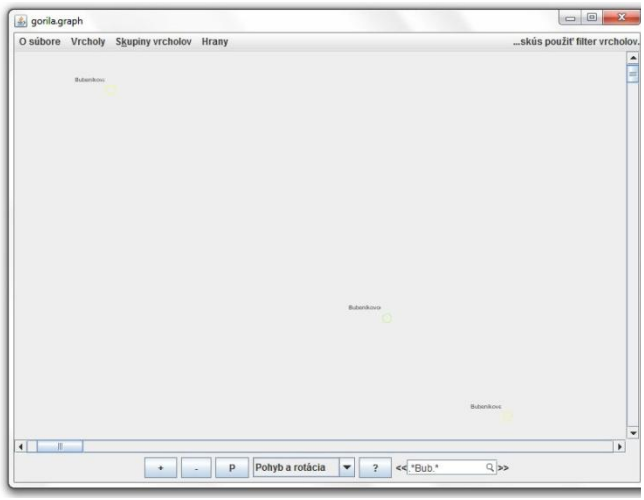
Obr. 31. : Zvýraznené vrcholy.



Obr. 32. : Zvýraznené vrcholy a ďalšie 2 sú označené. Medzi označenými bol pokus o nájdenie najkratšej cesty, výsledok akcie je v pravom hornom rohu.



Obr. 33. : Na obrázku sú vrcholy po vypnutí funkcie, tvar podľa počtu susedov. Modrým textom je pridaná poznámka.



Obr. 34. : Na obrázku je odprezentovaná funkcia vyhľadávania.

2.3 Zoznam klávesových skratiek

Tieto klávesové skratky je možné použiť len vo vizualizačnej obrazovke a len v určitom používateľskom móde pre ovládanie myši a kláves. Väčšina z nich len zapína a vypína funkcie aplikácie a pod. Takže urýchľujú prácu a používateľ sa nemusí k nim preklikávať cez menu.

Pre prepnutie do módu „Pohyb po obrazovke“ stlač **T.**

- **Ľavý klík a ťahanie myšou** - pohyb po obrazovke
- **Shift + Ľavý klík a ťahanie myšou** - rotačný pohyb
- **Ctrl + Ľavý klik a ťahanie myšou** - shear pohyb
- **Pohyb kolieskom na myši** - približovanie a oddiaľovanie

Pre prepnutie do módu „Pridávanie poznámok“ stlač **Y.** Následné jednoducho pravým kliknutím na plochu sa dá pridať poznámka. Rovnako sa dá vytvoriť ľavým potiahnutím obdĺžnik pre skupinovú poznámku.

Pre prepnutie do módu „Označenie vrcholov“ stlač **R.**

- **Ľavý klík a ťahanie myšou** - označenie skupiny vrcholov
- **podrž Shift + Ľavý klík** - jednotlivé označenie vrcholov
- **Ľavý klik** - označenie jedného vrcholu
- **Ctrl** - presun kamery ku vrcholu

Prílohy C – Štruktúra dátového disku

Na priloženom dát. disku sa nachádzajú:

\	- obsahuje elektronické verzie bakalárky v *doc* a *pdf* formáte
\project	- koreňový adresár projektu
\project \doc	- vygenerovaná *java-doc* dokumentácia k triedam
\project \nio	- zdrojové súbory balíka *nio*, spomenutý v 4.5.3 Pamäťovo namapované súbory
\project \src	- obsahuje zdrojové súbory a balíky pre aplikáciu
\project \uml	- obsahuje vygenerované diagramy vo formáte *ucls* a *jpg*
\ivis_developer	- ide o vývojovú verziu, ktorá robí záznamy o svojej činnosti
\ivis	- obsahuje spustiteľný program **ivis.jar**
\ivis\media	- obsahuje ďalšie súbory potrebné pre spustenie programu a vzorové vstupy

Pre spustenie programu je potrebné skopírovať zložku **ivis** na pracovnú plochu. Vzorové grafy, ktoré sa dajú otvoriť sú v zložke pracovná plocha **ivis/media.** Súbory je potrebné skopírovať a otvoriť z pracovnej plochy pretože si program vytvára dočasné súbory. Ďalej je vyžadovaná minimálne jdk1.6.0_21.