Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Projektová dokumentácia

Analyzátor sociálnych sietí

Tím č. 15, Socky (socky@kmit.sk)

členovia tímu:
Lucia Jastrzembská
Tomáš Jelínek
Tomáš Konečný
Katarína Kostková
Ľuboš Omelina

Predmet: Tvorba softvérového systému v tíme I
Tvorba informačného systému v tíme I
Pedagogický vedúci: Ing. Michal Barla

Ak. rok: 2007/2008

Zadanie – analyzátor sociálnych sietí

Pojmom sociálna sieť označujeme množinu ľudí, organizácií alebo iných sociálnych entít prepojených (sociálnymi) vzťahmi ako napríklad (ne)priateľstvo, príbuzenstvo, spolupráca alebo výmena informácií. Vzťahy medzi entitami pritom môžu, ale nemusia byť symetrické. Za sociálnu sieť môžeme označiť aj samotný Web s hypertextovými odkazmi medzi jednotlivými stránkami.

V poslednej dobe sa na Webe stali populárnymi tzv. sociálne portály (myspace.com, orkut.com, mojiznami.sk a iné), ktoré evidujú milióny používateľov budujúcich svoje sociálne siete. Spolu s nimi rastie dôležitosť analýzy sociálnych sietí, ktorá nám pomáha lepšie pochopiť statické a dynamické vlastnosti siete ako celku ale aj usudzovať o vlastnostiach jednotlivých uzlov siete. Zaujíma nás kto je významný rozbočovač (angl. hub), aké zhluky sa dajú nájsť v sociálnej sieti, kto vystupuje ako spojka medzi viacerými zhlukmi, kto je izolovaný, kto je v centre diania a kto naopak na periférii a pod.

Analýza sa opiera o formálnu (grafovú) reprezentáciu siete, čo umožňuje nasadenie vizuálnych a matematických metód analýzy. Základný proces analýzy pozostáva z predspracovania (vyčistenia) dátovej vzorky, aplikovania vybranej metódy analýzy a vizualizácie výsledkov. Konečným výstupom analýzy môže byť aj export zistených hodnôt do rôznych formátov či dátových úložísk externých systémov.

Sociálne siete a ich potenciál sa v poslednej dobe stali populárnym výskumným problémom a sú aj súčasťou výskumných projektov, ktoré sa riešia na Ústave informatiky a softvérového inžinierstva FIIT STU. Analýza sociálnych sietí však nie je úplne nový problém bolo pre ňu vymyslených a naimplementovaných množstvo zaujímavých metrík a metód. Skúmajte problematiku analýzy sociálnych sietí. Navrhnite, implementujte a zdokumentujte softvérové prostredie, ktoré predstavuje rámec pre vykonávanie analýzy sociálnych sietí. Riešenie by malo podporovať pridávanie nových metód analýzy, riadenie samotného procesu analýzy a prácu s výsledkami. Súčasťou riešenia je overenie rámca analyzovaním reálnych dát z existujúcich sociálnych sietí (získané z externých zdrojov).

Úvod

Predkladaný dokument je výsledkom práce tímu číslo 15 počas jedného semestra v predmete Tvorba softvérového/informačného systému v tíme. Zadaním projektu bolo zanalyzovať oblasť sociálnych sietí, navrhnúť a implementovať softvérový systém, schopný vykonať analýzu siete.

Dokument je rozčlenený na dve časti, prvá sa venuje samotnému softvérovému systému, jeho analýze, špecifikácii, návrhu a výsledkom prototypovania. Druhá časť dokumentu s názvom Riadenie obsahuje ponuku, plán a tiež zápisnice z formálnych stretnutí tímu s pedagogickým vedúcim.

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHONOLÓGIÍ

Časť I. - Softvérový systém

Obsah

Úvod

Nasledujúca časť dokumentácie sa venuje nášmu softvérovému systému, analyzátoru sociálnych sietí. V úvode práce na projekte sme museli zanalyzovať oblasť sociálnych sietí, tiež oblasť už existujúcich softvérových riešení a existujúcich technológií, resp. postupov, ktoré sme v práci na projekte využili.

Analýze sme venovali podstatnú časť práce na projekte, pretože dobrá analýza je predpokladom úspešnej špecifikácie a návrhu analyzátora. Preto 1. a 2. kapitola sa venujú analýze.

Kapitola 3. sa venuje špecifikácii systému, špecifikáciu sme rozdelili na špecifikáciu funkcionálnych a špecifikáciu nefunkcionálnych požiadaviek na systém. Podrobnejšie sa venujeme jednotlivým prípadom použitia.

4. kapitola sa venuje návrhu architektúry systému a návrhu používateľského rozhrania.

5.tu kapitolu sme venovali podrobnému návrhu systému. Zjemnili sme hrubý návrh z kapitoly 4 a doplnili ho. Táto kapitola bola nutným predpokladom prototypovania, v ktorom sme mali v úmysle nie len si overiť, niektoré predpoklady ale prototypované často sme mali v úmysle ďalej rozvíjať do výslednej aplikácie.

Nasledujúcu 6.tu kapitolu sme venovali prototypu, jednak opisu častí, ktoré sme prototypovali spolu s dôvodmi, prečo sme sa rozhodli prototypovať práve tieto časti a tiež sme v tejto kapitole opísali závery prototypovania.

V 7mej, predposlednej kapitole celej práce sme sa venovali výslednému produktu, systému Mitandao – analyzátor sociálnych sietí. V kapitole sme opísali implementačné detaily zaujímavých častí systému, možnosti rozšírenia systému a jednu podkapitolu sme venovali aj opisu požiadaviek na systém, ktoré sa nám nepodarilo splniť, spolu s odôvodnením.

Posledná kapitola, záver, je zhodnotením celej práce počas semestra a tiež zhodnotením častí dokumentácie.

Samozrejme nezabudli sme ani na časť s názvom použitá literatúra, ktorá sa nachádza na konci dokumentácie a obsahuje zoznam dokumentov, s ktorých sme počas práce na projekte čerpali. Za použitou literatúrou sú zoradené jednotlivé prílohy, zväčša ide o príručky pre používateľov.

Skratky

Zoznam použitých skratiek v práci:

API	- **A**plication **P**rogram **I**nterface
DML	- **D**ata **M**anipulation **L**anguage
GUI	- **G**raphical **U**ser **I**nterface
HITS	- **H**ypertext **I**nduced **T**opic **S**election
JMPI	- **J**AVA **M**essage **P**assing **I**nterface
JRE	- **J**ava **R**untime **E**nvironment

UC - **U**se **C**ase

XGMML - e**X**tensible **G**raph **M**arkup and **M**odeling **L**anguage

XML - **E**xtensible **M**arkup **L**anguage

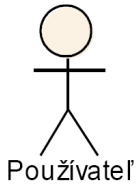
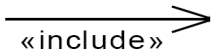
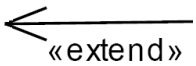
Vysvetlenie pojmov

Pri analýze sociálnych sietí sú najdôležitejšími pojmami uzol a hrana, tieto však môžu byť pomenované aj inými ekvivalentami. Preto uvádzame tabuľku obsahujúcu možné, ekvivalentné pomenovania týchto pojmov, ktoré môžete v práci nájsť.

uzol	subjekt, vrchol
hrana	spojenie, väzba, vzťah
graf	sieť, sociálna sieť

Použitá notácia

Diagramy použité v dokumentácii boli navrhované v UML nástroji Enterprise Architect a ich notácia je vysvetlená v nasledujúcej tabuľke.

Asociácia	
Hráč	
Prípad použitia	
Závislosť	
Rozšírenie	

1. Analýza problémovej oblasti

Keďže oblasť sociálnych sietí nie je našou doménou museli sme si túto oblasť a možnosti jej skúmania dôsledne preštudovať. Poznatky sme sa rozhodli získať z knihy **Introduction to Social Network Methods** [2], ktorá nám poskytla veľmi dobrý základ. Dozvedeli sme sa o tom, ako sociálne dáta skúmať, na čo sa pri ich analýze zamerať a tiež aké metódy existujú na ohodnocovanie uzlov v sieti. Nasledujúce podkapitoly 1.1. a 1.2. sa venujú práve týmto otázkam.

Podkapitola 1.3. je venovaná existujúcim softvérovým riešeniam. Najviac nás zaujal softvér Weka, ktorý je určený na dolovanie v dátach a softvér UCINET na analýzu sociálnych sietí.

Posledná podkapitola tejto časti, 1.4. sa venuje algoritmom HITS a Pagerank, ktoré sa používajú pri skúmaní webu. Web však možno tiež chápať ako sociálnu sieť a preto je ich využitie možné aj v nami navrhnutom analyzátore.

1.1. Sociálne siete

Sociológia ako veda sa zaoberá spoločnosťou, vzťahmi v spoločnosti a analýzou týchto vzťahov. Najlepšou možnosťou predstavy spoločnosti a vzájomných vzťahov medzi jednotlivcami je sieť, preto hovoríme o sociálnej sieti.

Sociálna sieť však nemusí reprezentovať len ľudí a ich vzťahy, ale napríklad aj webové stránky alebo vedecké články, ich vzájomné prepojenia sú potom chápané ako ich vzťahy.

1.1.1. Sociologické dáta

Medzi dátami používanými v klasickom sociologickom výskume a v sociálnych sieťach existujú viaceré rozdiely.

Pri **klasickom sociologickom výskume** sú dáta uložené v tabuľke, kde riadky predstavujú rôzne subjekty a stĺpce atribúty. Jednotlivé bunky obsahujú samotné hodnoty atribútov, ktorých porovnávaním porovnávame rôzne subjekty. Výskum sa zameriava na subjekt a jeho atribúty, pričom subjekty, jednotlivci zo skupiny, sa vyberajú náhodne.

Pri **sociálnych sieťach** sú dáta v tabuľke uložené tak, že riadky aj stĺpce obsahujú tie isté subjekty a hodnoty v bunkách opisujú vzťah subjektov. Výskum je v tomto prípade zameraný na subjekt a jeho vzťahy s ostatnými, pričom porovnávanie subjektov sa deje na základe ich vzťahov s ostatnými. Výskum zahŕňa všetky subjekty zo skupiny (skúma sa celá skupina).

Zbieranie dát

1. **analýza celej siete**
   - zahŕňa všetky subjekty z určitej skupiny
   - poskytuje kompletný obraz o vzťahoch vrámci skupiny
   - získavanie dát je náročné, preto je vhodné na analyzovanie malých skupín s dobre viditeľnými hranicami (napr. trieda v škole)

2. **metóda snehovej gule**
   - na začiatku sa vyberú náhodné subjekty, ktoré vymenujú ďalšie subjekty, tie ďalšie atď. So získavaním údajov sa skončí, ak už nie sú identifikované žiadne ďalšie subjekty alebo nájdené subjekty sú príliš vzdialené od skúmanej skupiny.
   - využíva sa na získavanie informácií o špeciálnej skupine (napr. poštový doručovatelia)
   - nevýhodou je, že sa nenájdu izolované osoby a pri nesprávnom výbere počiatočných subjektov sa môžu vynechať celé podmnožiny subjektov

3. **ego-centrický (s prídavnými vzťahmi)**
   - na začiatku sa vyberú počiatočné subjekty, ktorých sa spýta na subjekty, s ktorými ich spája skúmaný vzťah. Potom sa všetky takéto subjekty skontaktujú a skúmajú sa vzťahy vrámci takto získanej skupiny
   - takto sa dajú získavať dáta z veľmi veľkých skupín, napríklad koľko vzťahov majú uzly a ako úzko sú previazané navzájom
   - touto metódou sa dajú skúmať lokálne podskupiny väčších skupín, dá sa odhadnúť hustota siete, reciprocita vzťahov a podobne
   - touto metódou sa nedajú určiť centrálne uzly, vzdialenosti medzi uzlami, ani porovnávanie pozícií subjektov

4. **ego-centrický (iba ego)**
   - zameriava sa na jednotlivca, nie na sieť ako celok
   - zbierajú sa iba údaje o vzťahoch jednotlivca s inými subjektami, ale už nie vzťahy týchto subjektov
   - zistením, že niektoré subjekty majú viac vzťahov sa dá zistiť, že v sieti existujú rôzne postavené subjekty a dá sa odhadnúť, ako ich postavenie vplýva na ich správanie
   - nedá sa skúmať ani celková hustota siete, ani mikro-štruktúra siete

Ohodnotenie vzťahov

1. **nominálne**
   a) **binárne**

- určuje, či vzťah existuje (kóduje sa 1) alebo nie (0)
- pre toto ohodnotenie existuje najviac algoritmov i teórií grafov

b) multi-kategórie

- subjekty sú požiadané, aby nielen určili, či vzťah existuje, ale aby určili aj druh vzťahu (výberom z viacerých možností)
- najčastejší spôsob analýzy je, že sa pre každú kategóriu vytvorí samostatná binárna sieť

2. ordinálne

- snaží sa opísať „silu“ vzťahu
- príkladom je napríklad označenie vzťahov ako pozitívne (1), neutrálne (0) alebo negatívne (-1)
- ordinálne dáta poskytujú viac informácií o vzťahoch ako nominálne, ale keďže existuje málo algoritmov na ich spracovanie, často sa prevádzajú na binárnu formu (určením zlomového bodu) alebo na interval

3. intervalom

- najpresnejšie popisuje daný vzťah
- vhodným zvolením otázky sa dá vzťah kvantifikovať (napríklad ako často si posielajú dva subjekty e-maily)
- mnohé algoritmy, ktoré boli pôvodne navrhnuté pre binárne ohodnotenie vzťahov, boli rozšírené pre spracovanie intervalov
- napriek tomu väčšina algoritmov pracuje s binárnymi dátami, preto sa tieto dáta často konvertujú na binárne (určením zlomového bodu)

1.1.2. Základné vlastnosti sietí a subjektov

Na sieť sa dá pozerať ako na celok alebo ako na vzťah jednotlivých subjektov a od nich odvodzovať vlastnosti siete. Ak sa na sieť pozeráme ako na spojenie jednotlivých subjektov, analyzujeme vlastnosti subjektov na základe ich spojení s inými subjektami, pomocou diád a triád.

Diády sú dva subjekty a ich nejaké vzájomné spojenie. Ak berieme do úvahy, že spojenie je orientované, tak môžu nastať 4 rôzne prípady:

- subjekty spojené nie sú
- prvý subjekt vysiela, ale neprijíma
- druhý subjekt vysiela, ale neprijíma
- obidva subjekty vysielajú aj prijímajú

V **triádach** ide o vzťah 3 subjektov. Tu je možných kombinácií až 64. Niektorí vedci (zďaleka však nie všetci) si myslia, že v triádach sú zakotvené všetky základné vzťahy, ktoré v spoločnosti vôbec môžu existovať.

Na **sieť** sa môžeme pozerať aj ako na **celok**, vtedy asi najzákladnejšou vlastnosť siete je jej veľkosť. Je pravdepodobné, že ak je malá sieť (povedzme 5 ľudí v jednom tíme na školskom projekte) tak sa budú všetci poznať a budú v nej iné vzťahy ako vo veľkej sieti (povedzme 5000 ľudí vo firme).

Ďalšou zo základných vlastností sociálnej siete sú **spojenia**. Ide tak o spojenia, ktoré skutočne existujú, ako aj o tie, ktoré by existovať mohli, ale neexistujú. Zaujímavá je vlastnosť siete určená ako podiel všetkých existujúcich spojení a všetkých možných spojení. Takto sa dá určiť nakoľko „priateľská“ daná sieť je.

Pri veľkých sieťach sa často izolujú od seba nezávislé skupiny, ktoré so zvyšnou časťou siete spojené vôbec nie sú.

1.2. Metriky na ohodnocovanie sociálnych sietí

Pri analýze sociálnych sietí môžeme analyzovať buď uzly samotné alebo sieť ako celok. Existuje viacero vlastností, ktoré môžeme v sieti skúmať a takisto existuje viacero metód na skúmanie sily a centrálnosti jednotlivých uzlov.

1.2.1. Veľkosť, hustota a stupeň siete

Veľkosť siete je jednou z najzákladnejších vlastností siete vôbec. Je určená jednoduchým spočítaním všetkých subjektov v sieti. Ak si označíme tento počet ako *k,* počet dvojíc v sieti je potom *(k * (k – 1))* ak spojenie AB nie je to isté ako spojenie BA. Z toho je jasne vidieť, že pri malých sietiach je ešte možné (a časté), že všetky možné spojenia existujú aj v skutočnosti, pri veľkých je však počet tak vysoký, že je pre ľudí nemožné tieto spojenie udržiavať.

Keďže sme si ukázali, koľko rôznych spojení v sieti v skutočnosti môže existovať, a je v reálnom svete veľmi zriedkavé, že sa všetky spojenia skutočne aj uskutočnia, je zaujímavé skúmať veličinu nazývanú **hustota siete**, ktorá určuje, nakoľko blízko k úplne spojenej sieti je daná sieť. Hustota siete je určená ako podiel existujúcich a možných spojení.

Keď si uvedomíme, že spojenia v sieťach sú orientované (subjekt môže informácie vysielať aj prijímať) tak môžeme tieto dva rozdiely zobrať do úvahy a skonštruovať analýzu, ktorá pre každého účastníka siete vyjadruje, koľko percent možných spojení uskutočnil a aká časť týchto spojení je smerom von a aká smerom dovnútra. Takto sa dajú zistiť údaje o tom, ktorý subjekt je nakoľko dôležitý, ktorý je skôr zdroj informácií a ktorý skôr cieľ.

Dosiahnuteľnosť vyjadruje, či je jeden subjekt dosiahnuteľný iným, či existuje

medzi nimi spojenie. Ak sú spojenia orientované, tak sa môže stať, že subjekt A je dosiahnuteľný subjektom B, ale subjekt B nie je dosiahnuteľný subjektom A.

Reciprocita a tranzitivita

Za istých okolností môže byť zaujímavé skúmať, aj vlastnosti reciprocita a tranzitivita. **Reciprocita** znamená, že ak subjekt A je dosiahnuteľný subjektom B, potom aj subjekt B je dosiahnuteľný subjektom A. Pomocou tejto vlastnosti sa dá zistiť, či je niekto viac zdroj alebo cieľ, či je skôr informačná agentúra alebo veľmi vážený subjekt, ktorému veľa iných poskytuje informácie.

Ak si uvedomíme, koľko rôznych spojení môže existovať pri triáde, je jasné, že sa musíme zamerať iba na niektoré z nich. Najčastejšie sa pri triádach zameriavame na **tranzitívne vzťahy**. **Tranzitivita** znamená, že ak A je spojené s B a B je spojené s C, potom aj A je spojené s C. Špeciálnym prípadom tranzitivity je takzvaná **teória vyváženosti**. Vychádza z predpokladu, že ak subjekt A má rád subjekt B a subjekt B má rád subjekt C, potom aj subjekt A bude mať pravdepodobne rád subjekt C (tento vzťah platí aj pri iných spojeniach, ako je napr. nenávisť, nepriateľstvo atď.).

Vzdialenosť

Nakoľko sa sociálne siete často reprezentujú ako orientovaný alebo neorientovaný graf, cesta od jedného subjektu ku druhému je popísaná ako cesta z jedného vrchola do iného cez sled hrán a iných uzlov. Pri tejto ceste môže nastať niekoľko možných alternatív.

- ten istý vrchol aj hrana sa môže viackrát opakovať
- tá istá hrana sa opakovať nemôže, vrcholy sa však opakovať môžu
- nemôžu sa opakovať vrcholy ani hrany

Pri analýze sociálnych sietí sa najčastejšie používa práve posledná z uvedených alternatív. Vzdialenosť medzi subjektom A a B potom najčastejšie znamená, koľko bolo spravených krokov pri prechode od uzla A do uzla B, pričom sa uzly ani hrany nemôžu opakovať.

1.2.2. Diameter a geodetická vzdialenosť

Geodetická vzdialenosť

Najčastejšie používaná vzdialenosť medzi dvoma subjektami v sieti je takzvaná geodetická vzdialenosť. Tá je určená ako najkratšia možná cesta od subjektu A ku subjektu B. Najdôležitejšia je preto, lebo práve po nej tečie najviac informácií. Dôvod je veľmi jednoduchý. Ak napríklad subjekt A pozná subjekt B priamo (vzdialenosť 1), ale pritom tiež subjekt A pozná aj subjekt C, ktorý pozná subjekt D a ten subjekt E a ten subjekt B (vzdialenosť 4), je najpravdepodobnejšie, že ak potrebuje odovzdať nejakú

informáciu subjektu B, použije priame spojenie.

Diameter

Ak je sieť úplne spojená, môžeme sa zaujímať, ako rýchlo sa dokáže šíriť informácia, ak začne na ľubovoľnom mieste. Určíme si teda geodetickú vzdialenosť medzi všetkými vrcholmi a najväčšia z nich je takzvaný **diameter**. To nám určí, nakoľko sú od seba vzdialené najvzdialenejšie subjekty v sieti.

1.2.3. Tok, súdržnosť a ovplyvňovanie

Maximálny tok je počet všetkých možných ciest od subjektu A ku subjektu B. Myšlienka je v tom, že čím väčším počtom rôznych ciest môže subjekt A dosiahnuť subjekt B, tým je spojenie silnejšie.

Hubbelova a Katzova súdržnosť' sú veľmi podobné ako maximálny tok, rozdiel je len v tom, že sa vzdialenosti váhujú. To znamená, že ak je vzdialenosť od A ku B 1, je to omnoho dôležitejšie, ako keď je vzdialenosť napr. 10. Rozdiel medzi Hubbelovým a Katzovým prístupom je v tom, že Hubbel považuje spojenie samého do seba za najsilnejšie, kým Katz nie.

Prístup nazývaný **Taylorove ovplyvňovanie** sa používa, ak nás zaujíma predovšetkým to, aká časť spojení každého jednotlivca je smerom von a aká smerom dnu.

1.2.4. Sila a centrálnosť uzlov

Existuje niekoľko metód na skúmanie sily a centrálnosti subjektov v sieti. Základné metódy sa zameriavajú na určovanie stupňa uzla, blízkosti dvoch uzlov a toho či je uzol prostredníkom pri toku dát.

Stupeň uzla určuje počet spojení, ktoré uzol má s inými uzlami. Čím viac spojení uzol má, tým má väčšiu silu. Okrem počtu spojení sa rozlišuje aj to, či ide o tok do alebo z uzla. Ak ide viac tokov do uzla, uzol je prestížny a dôležitý, na druhej strane ak ide väčšie množstvo tokov z uzla, uzol vo veľkej miere vplýva na ostatné uzly.

Blízkosť uzlov ako ďalšia možnosť určovania sily a centrálnosti uzla sa zakladá sa vzdialenosti uzlov. Vzdialenosť v tomto prípade znamená, na koľko krokov sa uzol nachádza od všetkých uzlov v sieti. Blízkosť sa potom vypočíta ako vzdialenosť deleno počet uzlov v sieti krát sto percent.

Uzol je **prostredníkom,** ak tok informácií medzi dvoma uzlami ide cez neho. Sila uzla pri tejto metóde narastá s počtom uzlov, ktoré sú na ňom závislé. Znamená to, že čím viac uzlov musí posielať tok informácií cez uzol X, tým väčšiu silu uzol X má. Avšak ak sú dva uzly spojené viacerými cestami a uzol X sa nenachádza na všetkých ich cestách, jeho sila klesá.

Metóda vlastného vektora sa snaží nájsť najcentrálnejší uzol v rámci celej siete a menšiu pozornosť venuje lokálnym centrálnym uzlom. Vlastný vektor je pritom zbierka vlastných hodnôt a vlastná hodnota je pozícia každého uzla s ohľadom na jeho rozlohu.

Pri **centrálnosti toku** sa prihliada na to, či uzly ležia na ceste, spojení iných uzlov, či sú prostredníkmi. Rolu prostredníka môžeme chápať ako silu. Ak sú však uzly spojené viacerými cestami, radšej použijú iné cesty ako tie, na ktorých je prostredník, aj keď sú dlhšie. Predpokladáme teda, že účastníci využívajú všetky cesty, ktoré ich spájajú úmerne ich dĺžke.

Rola prostredníka (betweenness) je vyčíslená ako miera celkového toku na dráhach účastníka. Pre všetkých účastníkov narastá podľa toho, ako je účastník zapletený vo všetkých tokoch medzi všetkými účastníkmi. Táto miera narastá s rozsahom a hustotou siete. Počíta sa v pomere k celkovému medzi toku (betweeness flow), ktorý sa týka účastníka.

1.2.5. Skupiny a podgrafy v sieti

Siete sú vo všeobecnosti budované spájaním diád a triád do veľkých a úzko spájaných sietí. Veľké siete sa teda skladajú z malých a pevných. Ak sa v sieti vytvoria skupiny, ktoré majú aj niektorých spoločných členov, informácie sa šíria rýchlejšie a celá sieť sa rýchlejšie monitoruje.

Idea sub-štruktúry, skupín a partií vnútri siete je silným nástrojom na pochopenie sociálnej štruktúry a vnorania individuálnych uzlov.

Metóda zdola nahor hovorí, že aj 2 spojené uzly tvoria skupinu. Postupným rozširovaním vznikajú veľké skupiny. Dôležité je najprv pochopiť jedincov a to ako sú včlenení do siete.

Partia (clique) je množina členov, ktoré sú spolu viac prepojení ako s členmi mimo svojej partie. U ľudí sú to partie zložené z ľudí rovnakého veku, pohlavia, rasy atď.

Definícia partie je však veľmi striktná, každý člen musí mať priamy spoj so všetkými členmi partie. Existujú spôsoby ako možno túto definíciu zjemniť. V **N-partii** je každý člen so všetkými členmi spojení na viac ako 1 krok (zvyčajne 2). N určuje max počet krokov. Každý člen je teda tiež spojený so všetkými členmi, ale s niektorými priamym spojom a s inými na N krokov.

Pri N-partii sa však môže stať, že členovia sú spojení na N krokov, avšak aj cez členov, ktorí do danej partie nepatria. V **N-klane** je potrebné, aby bola splnená nie len podmienka maximálnej vzdialenosti uzlov, N krokov, ale aby tieto išli len medzi členmi partie.

V **K-plexe** je členom partie veľkosti N každý, kto má priamy spoj s N-K členmi partie. Týmto spôsobom nájdeme veľké množstvo malých zoskupení.

K-jadro (K-core) je maximálna skupinu členov, z ktorých všetci sú spojení s K členmi skupiny. Čím je K menšie tým väčšiu skupinu nájdeme.

Metóda zhora nadol sa pozerá na celú štruktúru a identifikuje podštruktúry ako časti, ktoré sú lokálne hustejšie ako ostatná časť. Hľadá diery a slabé miesta, definuje

miesta, kde môže byť sieť rozdelená do menších častí.

Komponenty sú tie časti, ktoré sú prepojené vnútri, ale nie navzájom medzi sebou. Izolanty sú tiež komponentami. Zaujímavejšie sú tie, ktoré delia sieť do oddeliteľných častí a kde každá časť má niekoľko členov, ktorí sú spojení navzájom. Zaujíma nás blízkosť spojov.

Bod rezu, určuje taký uzol, ktorý ak zo siete vyberieme, rozdelí sieť na časti, **bloky**. Môžeme hľadať maximálne nedeliteľné podgrafy grafu, práve lokalizovaním bodov rezu.

Lambda množina je cesta, resp. spojenie medzi uzlami, ktoré keď odstránime, tak rozdelí graf. Musíme skúmať cesty v sieti podľa množstva toku, ktorý cez daný spoj ide, potom vieme určiť práve body rezu, ktoré ak odstránime sieť sa rozpadne. Táto metóda nám ukazuje kde je sieť najzraniteľnejšia.

Ďalšou možnosťou je rozdelenie siete na **partície**, ktoré maximalizujú podobnosť vzorov v spojeniach účastníkov vnútri skupiny. Vnútri skupinu musí byť maximálna podobnosť, medzi skupinami maximálna rozdielnosť.

1.2.6. Pozícia v sieti a sociálne skupiny: Idea ekvivalencie

Pri analyzovaní sociálnych sietí uvažujeme pozíciu jednotlivých vrcholov v sieti, avšak na sociálnu sieť treba hľadieť aj ako na celkovú štruktúru a jednotlivé podštruktúry, ktoré sú vytvorené z orientovaných hrán grafu siete. Pri takomto pohľade na sociálnu sieť je možné jednotlivé vrcholy zoskupovať a nahrádzať jedným vrcholom. Pri zoskupovaní sa zoskupujú iba podobné (alebo ekvivalentne) vrcholy. Pohľad na takúto sociálnu sieť nám dáva možnosť lepšie pochopiť jej organizáciu a ponúka nám použitie algoritmov analýzy, ktoré by nebolo možné (alebo by nedávali korektné výsledky) použiť pri jej analyzovaní.

Na to, aby sme mohli zoskupovať jednotlivé vrcholy potrebujeme určiť (zadefinovať) podobnosť vrcholov. Túto ekvivalenciu (mieru podobnosti) je možné určiť pomocou viacerých kritérií, vzhľadom na to, čo nás na konkrétnej sieti pri analyzovaní zaujíma.

Štrukturálna ekvivalencia

- 2 vrcholy sú úplne ekvivalentné, ak majú všetky vzťahy s ostatnými vrcholmi rovnaké. Ak sú dva uzly štrukturálne ekvivalentné, tak sú aj automorfné a regulárne ekvivalentné. Pretože úplná štrukturálna ekvivalentnosť je pomerne ojedinelá, skôr nás zaujíma čiastočná štrukturálna ekvivalentnosť (jednoducho sa musia dva vrcholy podobať vzťahmi s ostatnými) a jej stupeň (na koľko, ako veľmi, sa podobajú).

Automorfnej ekvivalencii

- vypĺňa akúsi medzeru, medzi štrukturálnou a regulárnou ekvivalenciou. Každá štrukturálna ekvivalencia je aj automorfná a každá automorfná je aj regulárna, naopak to však neplatí. Automorfnej ekvivalencii sa venuje podkapitola 1.2.8.

Regulárna ekvivalencia

- 2 vrcholy sú regulárne ekvivalentné, ak majú rovnaký profil väzieb s členmi ostatných skupín Príklad: 2 mamy sú regulárne ekvivalentné, pretože každá má väzbu s deťmi a so svojim manželom. Štrukturálne ekvivalentné nie sú, pretože nemajú väzbu s tým istým manželom. Sú teda podobné, pretože majú vzťahy s rovnakými skupinami vrcholov.

1.2.7. Meranie štrukturálnej ekvivalencie

Pearsonov korelačný koeficient (Pearson correlation coeficient)

- meranie pomocou tohto koeficientu spočíva v ohodnotení spoločných vzťahov s ostatnými vrcholmi od -1 až po 1. Hodnota -1 znamená, že dva vrcholy majú presne opačné vzťahy k tretiemu vrcholu. Hodnota 0 znamená, že iba jeden z dvoch porovnávaných vrcholov má vzťah s tretím vrcholom a hodnota 1 znamená, že oba porovnávané vrcholy majú rovnaké vzťahy s tretím vrcholom. Na výpočet tohto koeficientu ako podobnosti vrcholov sa urobí priemer vzťahov so všetkými vrcholmi, s ktorými porovnávané vrcholy susedia.

Euklidova vzdialenosť (Euclidean distance)

- toto meranie je meranie **ne**-podobnosti a princíp spočíva v spočítaní štvorcov rozdielov (druhej mocniny rozdielov) medzi vrcholmi. Čím menší výsledok, tým sú vrcholy podobnejšie. V prípade, že sú vrcholy úplne ekvivalentné výsledná hodnota je 0.

Percento presného porovnania (Percent of Exact Matches)

- zaujíma nás, do akej miery je stupeň väzby do vrchola X presne rovnaký ako ten, čo korešponduje so spojom s vrcholom Y. Znie to zložito, ale je to jednoduché. Dva vrcholy majú mieru podobnosti 0.xy čo vyjadruje, že tieto dva vrcholy majú XY% rovnakých vzťahov s ostatnými vrcholmi.

Jaccardov koeficient (Jaccard coefficients)

- v niektorých sieťach sú väzby medzi vrcholmi riedke. V sieťach s nízkou hustotou väzieb (vzťahov) nie je vždy výhodné použiť predošlé metódy. Princíp tohto prístupu je výpočet čísla ako percentuálneho počtu koľko krát majú dva vrcholy vzťah (alebo rovnaký smer vzťahu) s tretím vrcholom oproti všetkým vzťahom, ktoré vrcholy majú.

1.2.8. Automorfná ekvivalencia

V predchádzajúcich častiach sme sa venovali hľadaniu podobnosti, čiže ekvivalencie medzi jednotlivými vrcholmi ako aj podskupinami vrcholov vrámci grafu. Boli definované a vysvetlené pojmy ako štrukturálna a regulárna ekvivalencia. Bola taktiež spomenutá aj automorfná ekvivalencia, ktorú si teraz bližšie charakterizujeme.

Z pohľadu hierarchie ekvivalencií zapadá automorfná ekvivalencia, vypĺňa akúsi medzeru, medzi štrukturálnu a regulárnu. Každá štrukturálna ekvivalencia je aj automorfná a každá automorfná je aj regulárna, naopak to však neplatí.

Definícia v kontexte štrukturálnej a regulárnej ekvivalencii

Kým štrukturálna ekvivalencia znamená, že konkrétne individuálne vrcholy sa môžu vzájomne zameniť (dva vrcholy majú úplne rovnaké väzby na ostatné vrcholy a ich výmenou sa v grafe nezmení žiaden vzťah), tak automorfná ekvivalencia znamená, že podštruktúry grafu (podgrafy, skupiny vrcholov) sa môžu navzájom vymeniť. Inými slovami existujú také dve skupiny vrcholov, že ich vonkajšie väzby so zvyšnými vrcholmi sú totožné. Regulárna ekvivalencia zachádza ešte ďalej a zovšeobecňuje konkrétne vrcholy do rolí alebo tried vrcholov a o ekvivalencií rozhoduje na základe ekvivalencie týchto skupín. Ťažisko pohľadu štrukturálnej ekvivalencie sa pri analýze zameriava na pozičné skúmanie individuálnych vrcholov a ich zapojenie do siete, regulárna ekvivalencia zameriava ťažisko skôr na role, ako indivíduá a skupiny a automorfná sústredí svoj pohľad medzi tieto dva extrémy.

Hľadanie automorfnej ekvivalencie je ale veľmi zložité. V princípe môže byť identifikovaná metódou *brute force*, ktorá preskúmava každú možnú permutáciu grafu a hľadá zhodu s originálnym. To však nie je reálne pri väčších grafoch ako aj pri váhovaných, či orientovaných hranách, kde sa situácia výrazne komplikuje a počet permutácií narastá veľmi rýchlo čím sa pre náročnosť stáva výpočet nerealizovateľný a existencia hľadaných podgrafov sa dá len ťažko predpovedať.

Preto sa hľadá skôr „približná“ automorfná ekvivalencia, keď jednotlivé podgrafy sú si podobné, ale nie úplne ekvivalentné. S podobnou ekvivalenciou pracuje aj softvér UCINET, zahrnutý v analýze existujúcich riešení.

Geodetická ekvivalencia je ďalšia možnosť ako skúmať podobnosti vrámci grafu. Vychádza z už definovanej geodetickej vzdialenosti. Pre každý vrchol možno napísať maticu s vektormi geodetických vzdialeností k ostatným uzlom. Po aplikovaní Euklidovkej metodiky na počítanie vzdialeností na maticu, dostaneme akýsi profil geodetickej vzdialenosti pre jednotlivé vrcholy. To znamená, že dva vrcholy sú geodeticky ekvivalentné, keď majú rovnaký profil (súbor) geodetických vzdialeností k ostatným vrcholom.

Na **hľadanie geodetickej ekvivalencie** sa používa napríklad algoritmus *maxism,* ktorý zoradí maticu geodetických vzdialeností pre jednotlivé uzly od najväčšej vzdialenosti a až potom aplikuje Euklidov vzorec a priebežne porovnáva profily jednotlivých vrcholov.

Z rozdelenia na základe geodetickej ekvivalencie možno potom vychádzať aj pri hľadaní automorfnej ekvivalencie.

Ďalším spôsobom ako nájsť automorfnú ekvivalenciu je použitím algoritmu *tabu search*, ktorý taktiež operuje na matici grafu a rozdeľuje ju na bloky na základe podobnosti na základe Euklidovských vzdialeností a súčtu jednotlivých vzťahov vrámci blokov.

1.2.9. Regulárna ekvivalencia

Hoci sme ju už v predchádzajúcich častiach analýzy definovali, je zo sociologického pohľadu na sociálne siete veľmi významná, preto sa sa k nej ešte v krátkosti vrátime.

Regulárna ekvivalencia je „najvoľnejšia“, najmenej striktná spomedzi všetkých ekvivalencií. Napriek tomu je zo sociologického hľadiska asi najdôležitejšia, keďže jej definícia je veľmi podobná popisu roly vrámci sociologického pohľadu na nejakú skupinu s definovanými vzťahmi.

Formálna definícia (Borgatti, Everett, and Freeman, 1996): Dva uzly sú regulárne ekvivalentné ak existuje rovnaké prepojenie na také susedné uzly, ktoré sú opäť vzájomne ekvivalentné, čiže majú vzťahy s rovnakými skupinami ekvivalentných uzlov.

Ak Jana má dcéru Evu a Barbora má dcéru Zuzanu, Jana a Barbora sú si ekvivalentné, pretože majú rovnaký vzťah k ostatným, teda k dcéram Zuzane a Eve, ktoré sú si tiež ekvivalentné. V tomto prípade pri skúmaní vzťahu nás nezaujíma konkrétne matka či dcéra.

Jednotlivé roly, na ktoré sa vrcholy delia sú definované na základe vzťahov medzi nimi. Napríklad manžel – manželka, muž – žena, zamestnávateľ – zamestnanec. Čiže role sa nedefinujú na základe atribútov jednotlivých vrcholov, ale hľadajú sa pri analýze podobnosti vo vzťahoch. Regulárna ekvivalencia môže byť opäť buď absolútna alebo približná.

Na rozdiel od predchádzajúcich ekvivalencií tu možno vytvárať triedy ekvivalentných hráčov na základe rôznych pravidiel, teda existuje viacero rovnocenných rozdelení do tried ekvivalencií.

Hľadanie regulárnej ekvivalencie

Existuje niekoľko typov algoritmov, ktoré analyzujú ekvivalenciu. Jedným zo spôsobov je otypovanie „susedov” daného uzla a potom hľadanie iných uzlov s podobne otypovanými susedmi.

Druhý spôsob je už spomínaný tabu search algoritmus. Tentokrát však pracuje tak, že uzly rozdeľuje na základe toho či k ním smerujú iba vstupné, výstupné či oba type hrán. Potom sa snaží spájaním uzlov z týchto skupín tvoriť ekvivalentné triedy.

1.2.10. PageRank a HITS

Web je možné vnímať aj ako sociálnu sieť, kde každý vrchol je osoba a každá

hrana ich prepojenie. Z tohto dôvodu je opodstatnené študovať prístupy na analyzovanie sociálnych sietí aj v kontexte dolovania v dátach na webe.

Metrika **stred** sa zaujíma o linky smerujúce von z uzla (odchádzajúce linky) pre orientované grafy, alebo o neorientované grafy. Tiež je možné skúmať stupeň a blízkosť a to, či je uzol prostredníkom. Pri **stupni uzla** sa za stred považuje ten vrchol, ktorý má najviac hrán, spojení, pri **blízkosti uzla** je stredom ten vrchol, ktorého vzdialenosť je od všetkých najmenšia a pri **prostredníku** je stredom ten vrchol, ktorý spája najviac inak nespojených vrcholov. V podkapitole 1.2.4. sme sa týmto centrálnostiam venovali bližšie.

Sofistikovanejšie ako skúmať len počet spojení je skúmať aj ich smer. Skúmanie **prestíže vrcholu** sa realizuje nad orientovanými grafmi. Zaujíma sa o hrany smerujúce do uzla. Najprestížnejší je potom ten vrchol, ktorý má najviac spojení smerom dovnútra. Pri **prestíži susedstva** (angl. proximity prestige) sa uvažujú všetky spojenia, nie len priame. V prípade **prestíže postavenia** (angl. rank prestige) sa uvažuje aj to, kto na daný vrchol ukazuje. Ide o spočítanie ohodnotenia postavenia vrcholov.

Bibliografické prepojenia a spolucitovanie

Spolucitovanie (Co – Citation) je spôsob na zhlukovanie podľa podobností. Ak prácu *j* a *i* cituje tá istá práca *k,* potom sa dá predpokladať, že práce i a j sú podobné.

Bibliografické prepojenie je zrkadlovým obrazom spolucitácie. Teda ak práce i a j citujú prácu k, potom sú si i a j pravdepodobne podobné. Táto metóda sa tiež používa na zhlukovanie.

Pri **PageRank algoritme** sa rozlišujú vnútorné a vonkajšie odkazy. Za vrcholy sa považujú stránky, za hrany odkazy medzi nimi. Využíva sa prirodzená demokracia na Internete a každá linka sa považuje za hlas danej stránky na tú, na ktorú sa odkazuje.

Nech i a j sú stránky a nech sa obe odkazujú na stránku k. Rozlišuje sa ale aj ohodnotenie stránky, ktorá sa odkazuje. Ak by mala stránka i väčšie ohodnotenie ako stránka j, potom jej hlas je dôležitejší. Ohodnotenie stránky sa ale nekopíruje automaticky na všetky stránky, na ktoré sa odkazuje. Vždy sa jej ohodnotenie delí počtom odkazov.

Silné a slabé stránky PageRank

Silné:

- dobrá odolnosť voči spamovým stránkam
- ohodnotenie stránky sa počíta nezávisle od dotazu používateľa, to znamená, že v dobe zadania požiadavky je nesmierne rýchly

Slabé:

- nedokáže rozoznať medzi kvalitnou stránkou a kvalitnou stránkou v závislosti na dotaze. Je to problém v prípade hľadaných výrazov ako je puma, jaguár atď.
- neberie do úvahy to, že na staršiu stránku sa v priebehu času naakumulovalo viac odkazov. Riešia to prístupy ako TimedPageRank.

TimedPageRank

PageRank sa počíta väčšinou tak, že sa po množine stránok hýbe náhodný používateľ, pričom existuje pravdepodobnosť, že zo stránky odskočí. TimedPageRank rieši problém popísaný v predchádzajúcej kapitole tak, že sa pravdepodobnosť, že náhodný používateľ príde na staršiu stránku zmenšuje a radšej odskočí niekam mimo.

HITS

Identifikuje dva druhy stránok:

- **Rozbočovač (hub)**: stránka, ktorá sa odkazuje na veľa kvalitných stránok
- **Autorita**: stránka, na ktorú sa odkazuje veľa kvalitných stránok

Počítať sa začína v momente zadania požiadavky. Funguje v nasledujúcich krokoch:

1. používateľ zadá dotaz
2. z databázy sa vyberie zoznam s počtom *t* stránok zaujímavých pre dané kľúčové slovo (koreň, W).
3. táto množina sa zväčší o stránky, ktoré sa odkazujú na niektorú z koreňovej množiny a o tie, na ktoré sa odkazuje niektorá zo stránok z koreňovej množiny. Aby táto množina nebola príliš veľká, tak sa to obmedzuje tak, že každá stránka môže zatiahnuť len istý počet iných.
4. vypočíta sa stupeň rozbočovača a stupeň autority každej stránky.

Silné a slabé stránky HITS

Silné:

- dokáže nájsť relevantné výsledky v závislosti od používateľského dotazu

Slabé:

- rozbočovač je ľahko napadnuteľný spamerom
- pri rozvíjaní koreňa sa do zoznamu môže dostať veľké množstvo nerelevantných stránok
- pomalé v okamihu zadania požiadavky

Vzťah HITS, spolucitácie a bibliografických prepojení

Autorita je to isté ako spolucitácia vo webovom kontexte.

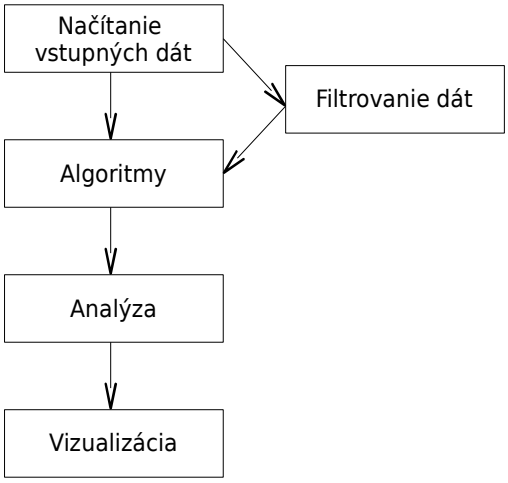
Bibliografické prepojenia sú to isté, ako rozbočovač vo webovom kontexte.

1.3. Existujúce softvérové riešenia

1.3.1. Weka

Weka je kolekcia algoritmov na strojové učenie na dolovanie v dátach. Je možné ju použiť samostatne alebo volať ako časť vlastného Java kódu.

Tok



Obrázok 1

Načítanie vstupných dát

Dáta môžu byť načítané buď z databázy alebo z CSV súboru, do ktorého však treba pridať anotácie. Weka si z takéhoto vstupu vytvorí formát ARFF. Dáta z tohto súboru je možné filtrovať, napr. aby sa analýza uskutočnila iba na niektorých atribútoch. Potom sa na dátach vykoná vybraný algoritmus na strojové učenie.

WEKA GUI

WEKA ponúka štyri možnosti spustenia:

1. **príkazový riadok**
2. **explorer mód**

- podporuje základnú funkcionalitu
- celú dátovú množinu načíta do pamäte – použiteľné len pre relatívne malé množiny

3. **knowledge flow mód**
   - podporuje základnú funkcionalitu
   - dokáže pracovať so prúdovým vstupom – podstatne väčšie množstvo dát
   - priateľskejšie používateľské rozhranie (drag and drop)

4. **experimenter mód**
   - dáva možnosť nastaviť viac algoritmov nad viaceré dátové množiny alebo viac rôznych parametrov nad rovnaký algoritmus a dátovú množinu
   - dokáže počítať aj distribuovane

Implementačné riešenia

Distribuované počítanie

Nejedná sa o distribuované počítanie jedného algoritmu, ale o distribuované riešenie viacerých algoritmov (každý algoritmus s vlastnými parametrami a vlastnými vstupnými dátami môže byť riešený na vlastnom počítači). Použitá je technológia Java RMI.

„Drag and drop“ používateľské rozhranie

Jedná sa o veľmi efektné používateľské prostredie. Používateľovi umožňuje použitím metódy drag and drop nastaviť vstupný súbor, všetky filtre, použité algoritmy ako aj vizualizáciu.

Jedná sa o vlastnú implementáciu od Weky, ktorá je silne závislá od ostatnej časti systému a jej premigrovanie do iného prostredia by stálo veľké úsilie.

Riešenie pluginovateľnosti

Každý plugin (algoritmus) musí byť samostatná aplikácia s vlastnou **public static void main(String[] args)** metódou. Musí byť odvodený od abstraktnej triedy Classifier čo znamená, že musí spĺňať jej rozhranie. Celá logika výpočtu je v tejto aplikácii, pričom sa používateľ nemusí zaujímať o technické podrobnosti načítavania, filtrovania alebo zobrazovania dát.

Dáva to používateľovi veľkú možnosť napísať veľkú aplikáciu na analyzovanie sietí, ktorá bude Weku volať. Pri tomto riešení sa vychádza z predpokladu, že dolovanie v dátach je často len veľmi malá časť podstatne väčších aplikácií, celá logika teda nemá byť na softvéri, ktorý v dátach doluje.

1.3.2. UCINET

Ucinet je jedným z existujúcich softvérov na analýzu sociálnych sietí vyvíjaný spoločnosťou Analytic Technologies.

Na webovej stránke Analytic Technologies je možné si tento softvér na 30 dní stiahnuť a pozrieť a tiež obsahuje odkaz na knihu **Introduction to social network methods**, ktorú sme študovali v rámci objasnenia si základných pojmov a metód v sociologickom výskume.

Ucinet je softvérom ponúkajúcim veľké množstvo funkcií. Jeho nevýhodou však je neprehľadnosť a to, že používateľ musí mať znalosti z danej oblasti.

Ponúka množstvo metrík na skúmanie siete (komponenty, bloky, podskupiny, K-jadrá), určovanie centrálnych a silných uzlov (stupeň, prostredník, blízkosť), je možné farebne vyznačiť rovnocenné a nerovnocenné hrany (angl. reciprocal ties).

Dá sa nastaviť pozadie, farby a tvary uzlov, hrúbka a farba hrán. Tiež je možné zapnúť a vypnúť označenia toku dát. Používateľovi je možné prispôsobiť tiež veľkosť uzlov a ich popis. Je možné zobraziť váhu jednotlivých spojení.

Ponúka aj ego network viewer, ktorý umožňuje zobrazovať uzly po jednom, spolu s uzlami, s ktorými sú priamo spojené a tiež ich spojeniami navzájom.

Veľmi dobrá je vizualizácia prostredníctvom grafu, dá sa zobraziť aj 3D graf. Prispôsobenie sa používateľovým požiadavkám v zmysle zvýraznenie, zväčšenia a zafarbenia uzlov a hrán je tiež veľkým plusom.

Za nevýhodu by sme označili práve množstvo funkcií, ktoré tento softvér ponúka, pretože sú na úkor intuitívnosti a používateľovi sa v nich potom ťažko orientuje. Nie vždy tiež systém reaguje na danú požiadavku používateľa.

1.3.3. InFlow

Inflow je softvér na analyzovanie sociálnych a organizačných sietí. Pod organizačnou sieťou pritom tvorcovia InFlow chápu sociálnu sieť vrámci veľkých firiem a organizácií.

Podporované algoritmy

- stred siete
- siete “malého sveta” (angl. small world networks)
- klastrovanie
- hustota siete
- prestíž / ovplyvňovanie
- štrukturálna ekvivalencia
- susedstvo
- pomer externého a interného toku
- váhovaná priemerná vzdialenosť

- najkratšia cesta a distribúcia ciest

Zaujímavé vlastnosti

- Spojenia aj vrcholy môžu byť farbené na základe nejakých vlastností. Napríklad všetci autori budú inou farbou ako ich diela.
- Technológia čo - ak (What – If). Jedná sa o možnosť po zanalyzovaní a zobrazení sociálnej siete pridávať/odoberať hrany alebo vrcholy. Po každej takejto akcii sa analýza vykoná znova. To umožní zistiť napríklad, čo by sa stalo, ak by sme do spoločnosti prijali ďalšieho experta na komunikáciu medzi oddeleniami a pod.

2. Analýza technológií a postupov

2.1. Knižnice na vykresľovanie grafov

Vrámci analýzy bolo nevyhnutné zamyslieť sa nad spôsobom vizualizácie sietí v podobe grafov a taktiež výsledkov prevedených operácii nad dátami. Keďže samotné prevedenie vykreslenia grafov je samostatná problematika, ktorá priamo nesúvisí s podstatou nášho cieľa, rozhodli sme sa využiť niektorú z voľne dostupných vizualizačných knižníc s otvoreným zdrojovým kódom (angl. open source) pre programovací jazyk JAVA. Pri užšom výbere knižníc sme sa nechali inšpirovať referenciami iných softvérových riešení z podobnej doménovej oblasti. Na základe takýchto informácií sme bližšie preskúmali možnosti knižníc PREFUSE a JUNG.

2.1.1. Jung

JUNG (Java Universal Network/Grafic Framework) je univerzálnou, rozšíriteľnou knižnicou pre modelovanie, analyzovanie a vizualizácia dát reprezentujúcich sieť alebo graf.

Základné vlastnosti v kontexte našich požiadaviek:

- na vstupe dokáže čítať a spracovať xml
- obsahuje už implementované základné algoritmy z teórie grafov, dolovania v dátach a sociálnych sietí.
- podporuje rôzne schémy (angl. layout) pre zobrazovanie, čo nám dáva širšie možnosti pri návrhu výsledného používateľského rozhrania
- umožňuje pridávať uzlom atribúty, váhy, značky
- umožňuje vyberať podmnožiny z uzlov na základe atribútov, aplikovať boolovské funkcie na vrcholy
- podporované algoritmy pre zobrazovanie:
  - Fruchterman-Rheingold
  - Kamada-Kawaii
  - Eades
  - Self-Organizing Maps
  - Circle
- implementované algoritmy sociálnych sieti
  - štrukturálna ekvivalencia,
  - vzdialeností, toku a dôležitosti uzla
  - konektivita
- vlastné štatistiky a merania na grafe
- možno rozšíriť o CERN Colt package, ktorý zahŕňa
  - stupeň uzla

  - najkratšia cesta
  - koeficient zhlukovania
- v súčastnosti používaný v niekoľkých projektoch zaoberajúcich sa sociálnymi sieťami
- ľahké na programovanie.
- dobre zdokumentovaný
- reprezentácia grafov

2.1.2. Prefuse

PREFUSE je rovnako ako JUNG vizualizačný nástroj pre modelovanie a vizualizáciu.

Je však trochu chudobnejší čo sa týka ponúkaných už implementovaných funkčností týkajúcich sa problematiky sociálnych sietí. Je užšie špecializovaný na zobrazovanie.

Základné vlastnosti v kontexte našich požiadaviek:

- možno tvoriť samostatné aplikácie aj vnorené vizuálne komponenty.
- používa JAVA 2D knižnicu
- vhodný najmä pre SWING
- podpora pre interaktívne použitie, presúvanie, úprava vrcholov
- podpora viacerých schém rozloženia (angl. layouts)
- vstavaný DML pre prácu s dátami
- optimalizované na výkon vykresľovania
- taktiež ľahko použiteľný a dobre zdokumentovaný

Zhodnotenie

Oba popísane projekty disponujú pre náš účel potrebnou funkcionalitou pre vykresľovanie siete, respektíve grafu. Z tohto pohľadu sú rovnocenné i keď Prefuse deklaruje vyššiu optimalizáciu, čo by bolo výhodou pri väčších sieťach. Jung je však oveľa rozsiahlejší projekt, čo umožňuje využitie aj iných vstavaných funkcií a tým urýchliť samotnú implementáciu nášho softvérového riešenia pre analyzátor sociálnych sietí. Závisí od bližšej špecifikácie očakávaných funkčností, ktorá knižnica bude vhodnejšia, preto definitívne rozhodnutie bude spravené vrámci prototypovania.

2.2. Reprezentácia grafov

Náš systém používa na vnútorné spracovanie knižnicu JUNG. To znamená, že túto knižnicu používame aj na vnútornú reprezentáciu grafov v našom systéme. Táto knižnica natívne podporuje importovanie formátov Pajek a graphML. Naša aplikácia podporuje importovanie týchto formátov, avšak do tried knižnice JUNG je možné importovať graf aj pomocou dynamicky vytvorených štruktúr, čo umožňuje vytvoriť podporu viacerých formátov ako napríklad aj graf uložený v relačnej databáze.

2.2.1. Pajek

Pajek je formát súborov na ukladanie štruktúry a vlastností sociálnych sietí. Tento formát je založený na zoznamoch jednotlivých entít v grafe. Ako prvý je uvedený zoznam vrcholov, následne sú uvedené typy hrán a nakoniec samotný zoznam hrán.

Výhody použitia tohto formátu:

- ohodnotenie vrcholov a hrán viacerými atribútmi
- vytvorenie viacerých typov hrán
- ukladanie zobrazovaných vlastností grafu (farba, tvar, názov, ...)
- kompatibilita s viacerými softvérovými riešeniami
- jednoduchý na upravovanie a zrozumiteľná reprezentácia pomocou textového typu súboru
- existuje viacero konvertorov do iných formátov na reprezentáciu siete alebo grafu
- možnosť ukladania kardinality vzťahov

Tento formát podporuje viacero programov používaných na analýzu sociálnych sietí, čo umožňuje získať viacero testovacích údajov pri tvorbe a testovaní nášho systému. Rovnako je podporovaný knižnicou JUNG, na vstupe aj výstupe, ktorú budeme používať ako základ našej aplikácie. Použitím tohto formátu sa rozšíria možnosti a kompatibilita našej aplikácie.

2.2.2. GraphML

Je komplexný ľahko použiteľný formát na reprezentovanie grafu, ktorý je založený na formáte XML. Skladá sa zo základných značiek na popis štruktúry grafu a zo všeobecných značiek, ktorými sa dajú opísať vlastnosti konkrétnej aplikácie. Syntax tohto formátu je definovaná XML schémou.

Výhody použitia tohto formátu

- všeobecnosť a možnosť rozšírenia o vlastnosti a údaje, ktoré nesúvisia s grafom a vzťahmi.
  - pridaním štruktúrovaného obsahu v podobe elementov `<data>` a `<deafult>`
  - pridaním vlastností ku ktorémukoľvek elementu v dokumente graphML
- zadávanie vlastností rozsahom typových hodnôt (int, float, long, ...)
- pridávanie štrukturálnych elementov vo forme vlastností, čo umožňuje napríklad uložiť množinu vrcholov do jedného (zhlukovanie)
- kompatibilita s viacerými programami, určenými nielen na zobrazovanie
- možnosť použitia akejkoľvek XML knižnice na prácu s týmto formátom
- možnosť referencií na externé údaje

Tento formát je podporovaný aj knižnicou JUNG. Táto knižnica dokáže načítavať a zobrazovať grafy v tomto formáte a preto predstavuje pre nás rozšírenie možností našej aplikácie.

2.2.3. XGMML (eXtensible Graph Markup and Modeling Language)

Je značkovací jazyk založený na XML. Používa značky na popis uzlov a hrán v grafe. Účel, pre ktorý bol tento formát navrhnutý, bolo štandardizovanie formátu na výmenu grafov medzi rôznymi zdrojmi grafov. XGMML je optimalizovaný pre reprezentáciu sietí webových stránok, preto sa často používa pre robotov prechádzajúcich po Internete. XGMML bol použitý napríklad pre W3PAL, čo je systém navrhnutý na analýzu a syntézu webových stránok.

V našom projekte by bolo výhodné použiť ho ako výstupný formát, kvôli rozšíreniu kompatibility, avšak nie je vhodný pre vnútorné spracovanie a reprezentáciu, pretože údaje, ktoré sa vyskytujú v našom systéme nie sú len štruktúry založené na webových stránkach a použitie tohto formátu by znamenalo ukladanie veľkého prebytku údajov a veľké nároky na pamäť.

2.3. Distribuované počítanie

Aplikovanie jednotlivých algoritmov sociálnych sietí nad zadaným vstupným grafom môže byť pri väčšej sieti časovo veľmi náročná úloha, keďže výpočtová náročnosť rastie nelineárne v závislosti od počtu uzlov. To je dôvodom, aby sme sa zamysleli nad možnosťami distribuovaného počítania, v predchádzajúcich kapitolách spomínaných, vlastností a metrík sociálnej siete.

Na distribuované počítanie sa možno pozerať dvomi spôsobmi. Prvým z nich je aplikovanie súčasne viacerých rôznych algoritmov paralelne na viacerých počítačoch a sumarizácia ich výsledkov. Toto je jednoduchšia možnosť, keďže si nevyžaduje špeciálnu úpravu algoritmov, stačí implementovať možnosť zdieľania výsledkov po sieti medzi jednotlivými inštanciami aplikácie. S takýmto riešením sme sa stretli aj pri skúmaní existujúcich riešení, konkrétne pri softvéri Weka.

Druhým spôsobom je hľadanie možností ako paralelizovať jednotlivé algoritmy teda rozdeliť incidenčnú maticu grafu na časti, aplikovať nad každou z nich ten istý výpočet a parciálne výsledky potom správne skomponovať pre ich interpretáciu vrámci celého grafu. S takýmto riešením sme sa nestretli ani u jedného zo skúmaných, už existujúcich programov.

Z pohľadu teórie grafov existujú na teoretickej úrovni rozpracované riešenia pre niektoré grafové výpočty ako sú Dijsktrov, či Floydov algoritmus pre hľadanie najkratšej cesty v grafe, no s implementáciou algoritmov pre počítanie prostredníka, či blízkosti alebo koeficientu zhlukovania sme sa nestretli.

Distribuovať počítanie parametrov ako stupeň, dôležitosť uzla, či hustotu siete nepredstavuje závažný matematický ani implementačný problém, keďže incidenčnú maticu možno pomerne jednoducho rozdeliť a výsledky jednoducho spočítať. Takúto distribúciu možno previesť napríklad s použitím JMPI (JAVA Message Passing Interface). Vzhľadom na fakt, že tieto parametre majú iba lineárnu zložitosť, je použitie distribuovaného počítanie neopodstatnené.

Naopak pri prostredníkovi a blízkosti sa zložitosť pohybuje okolo N^2, tu však nemožno rozdeliť incidenčnú maticu na časti a pri výpočte pre jednotlivé uzly treba rátať so všetkými ostatnými uzlami a prepojeniami. Možno však počítať tieto parametre pre jednotlivé uzly nezávisle na rozličných počítačoch, je k tomu ale nutná kompletná sada dát. V konečnom dôsledku sa vykonajú niektoré operácie duplicitne a konečný počet operácií bude väčší. Pri veľkom počte vrcholov by však bol tento rozdiel zanedbateľný a rozdelenie výpočtu na N počítačov by ho N-krát urýchlilo.

Zhlukovanie či hľadanie ekvivalencií, ktorých zložitosť je exponenciálna, nemožno tak jednoducho distribuovať ako tomu bolo v predchádzajúcich prípadoch a nie je nám známy ani žiaden matematický model ako postupovať.

Ak sa rozhodneme medzi funkčnosti našej aplikácie zaradiť aj distribuované počítanie, musíme počítať s faktom, že neexistuje žiadna využiteľná knižnica, či komponent a ťažisko algoritmu budeme pre jednotlivé problémy musieť sami implementovať. Môžeme prípadne implementovať programovú podporu pre spoluprácu jednotlivých aplikácii po sieti a samotnú implementáciu paralelnej verzie algoritmov ponechať ako možné rozšírenie do budúcnosti.

3. Špecifikácia

3.1. Funkcionálne požiadavky

V tejto podkapitole sme identifikovali jednotlivé funkcionálne požiadavky, ktoré sme z dôvodu vytvorenia matice sledovateľnosti požiadaviek (viď. Príloha A) označili R01 až R37.

Vnútorná práca s údajmi

Podpora viacerých formátov vstupných údajov – pojmom vstupné údaje je myslená reprezentácia sociálnej siete (napr. vo forme grafu).

R01: Aplikácia podporuje vstup údajov z relačnej databázy

R02: Aplikácia podporuje vstup údajov zo súboru PAJEK

Podpora výstupných údajov vo viacerých formátoch – pod pojmom výstupné údaje sú myslené výsledky analýz a spracovania.

R03: Aplikácia podporuje výstup údajov do súboru PAJEK

R04: Aplikácia podporuje výstup údajov v používateľskom rozhraní

R05: Aplikácia podporuje výstup údajov v grafickom formáte do súboru

Podpora typových hrán vo vnútornom spracovaní – aplikácia umožňuje načítať, spracovať a poskytnúť na výstupe hrany rôzneho typu. S týmito hranami pracujú aj základné vnútorné algoritmy.

R06: Aplikácia umožňuje nastaviť typ hrán, ktoré majú byť zahrnuté do analýzy.

Logovanie aplikácie a priebehu analyzovania – aplikácia zaznamenáva priebeh od jej spustenia až po ukončenie vykonávania.

R07: Aplikácia umožňuje nastaviť rôzne úrovne logovania

R08: Aplikácia umožňuje nastaviť logovanie pre každý modul zvlášť

Spracovanie a vnútorná funkcionalita

Podpora viacerých algoritmov analýzy údajov - pri výbere algoritmu analyzovania (modulu analyzátora) je možné vybrať viac algoritmov. V prípade výberu viacerých sa aplikujú postupne, pričom každý algoritmus dostane na vstup pôvodnú sociálnu sieť v takej forme, v akej bola pred spustením analýzy.

R09: Aplikácia podporuje algoritmy implementované v knižnici JUNG 1.7.6

Vizualizácia a vizuálne úpravy v údajoch

Vizualizácia sociálnej siete je realizovaná vo forme grafu - výstupom analýzy vstupných dát je predovšetkým graf.

R10: Aplikácia podporuje možnosť vizualizovať len časť grafu

R11: Aplikácia podporuje možnosť vizualizovať údaje vo forme 2D grafu

R12: Aplikácia podporuje možnosť vizualizovať údaje vo forme 3D grafu

Práca s grafom – aplikácia umožňuje dynamicky pracovať so sieťou.

R13: Vyberať susedné vrcholy do hĺbky N

R14: Vyberať uzly s počtom hrán N

R15: Vyberať uzly na základe označenia (napr. ID < N alebo ID > N)

R16: Vyberať uzly náhodne (ručne)

Vizuálne úpravy zobrazenia sociálnej siete – aplikácia umožňuje vizuálne úpravy uzlov a hrán vo vykreslenom grafe:

R17: Nastavenie farebného označenia uzlov

R18: Nastavenie tvaru uzla

R19: Nastavenie veľkosti uzla

R20: Nastavenie popisu uzla

R21: Nastavenie hrúbky hrán

R22: Nastavenie farby hrán

R23: Skryť uzly a im prislúchajúce hrany

R24: Zobraziť uzly a im prislúchajúce hrany

R25: Skryť hrany

R26: Zobraziť hrany

R27: Skryť izolanty

R28: Zobraziť izolanty

Úpravy štruktúry sociálnej siete - aplikácia umožňuje používateľovi pracovať s výstupným grafom nasledujúcim spôsobom:

R29: Pridať nové uzly a im prislúchajúce hrany

R30: Odobrať uzly a im prislúchajúce hrany

R31: Pridať hrany

R32: Odobrať hrany

R33: Odobrať izolanty

Ukladanie a načítanie nastavení

Používateľ si môže uložiť svoje nastavenia do externého súboru nastavenia aplikácie:

R34: Aplikácia umožňuje uložiť a načítať nastavenia používateľského rozhrania

R35: Aplikácia umožňuje uložiť a načítať nastavenia pre každý používateľom zvolený modul

R36: Aplikácia umožňuje uložiť a načítať nastavenia analýzy

Pridávanie novej funkcionality

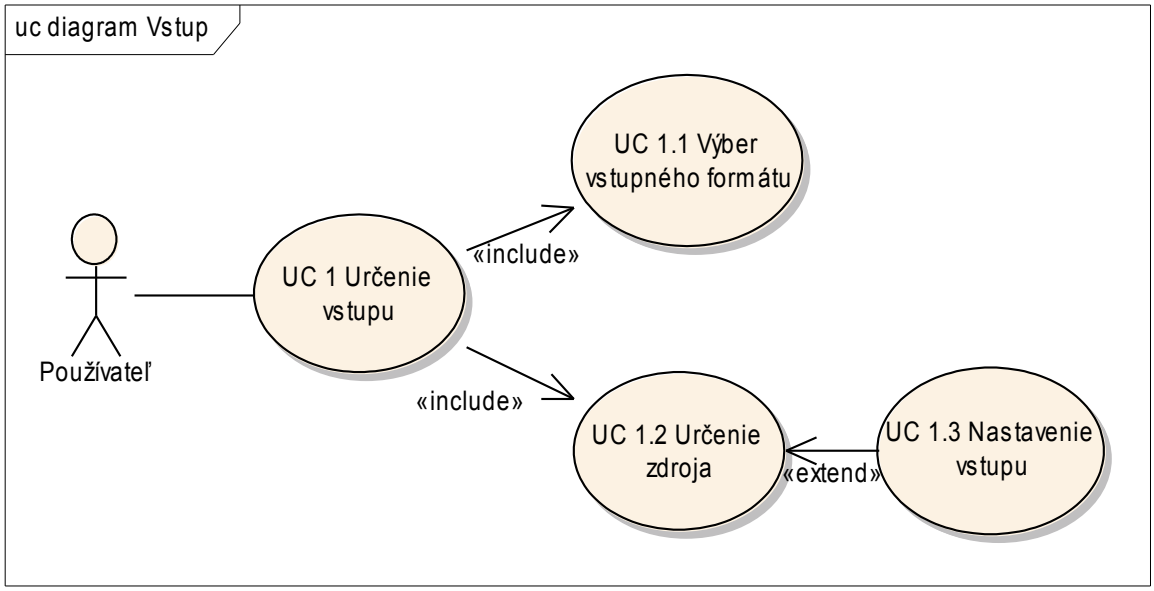
R37: Aplikácia umožňuje pridávanie nových modulov pomocou používateľského rozhrania

3.2. Prípady použitia

Pre funkcionálne požiadavky, špecifikované v predchádzajúcej podkapitole, sme vytvorili prípady použitia, bližšie opísané v tejto podkapitole. Pre kontrolu sme vytvorili maticu sledovateľnosti požiadaviek, ktorá je dôkazom toho, že každá požiadavka je súčasťou niektorého prípadu použitia a tiež, že každý prípad použitia pokrýva nejaké definované požiadavky.

Vstup a načítanie údajov

Používateľ musí vybrať typ údajového vstupu. Tento typ určuje, ktorý modul na spracovanie vstupu sa použije. Po výbere typu vstupu, nastaví konkrétny zdroj. Pri nastavení vstupu používateľ nastaví ďalšie nastavenia, ktoré môžu byť špecifické pre konkrétny zdroj. (napr. nastavenie databázy na konkrétnom serveri alebo výber inej siete v prípade, že v jednom súbore sa ich nachádza viac, nastavenie kódovania súboru, …)

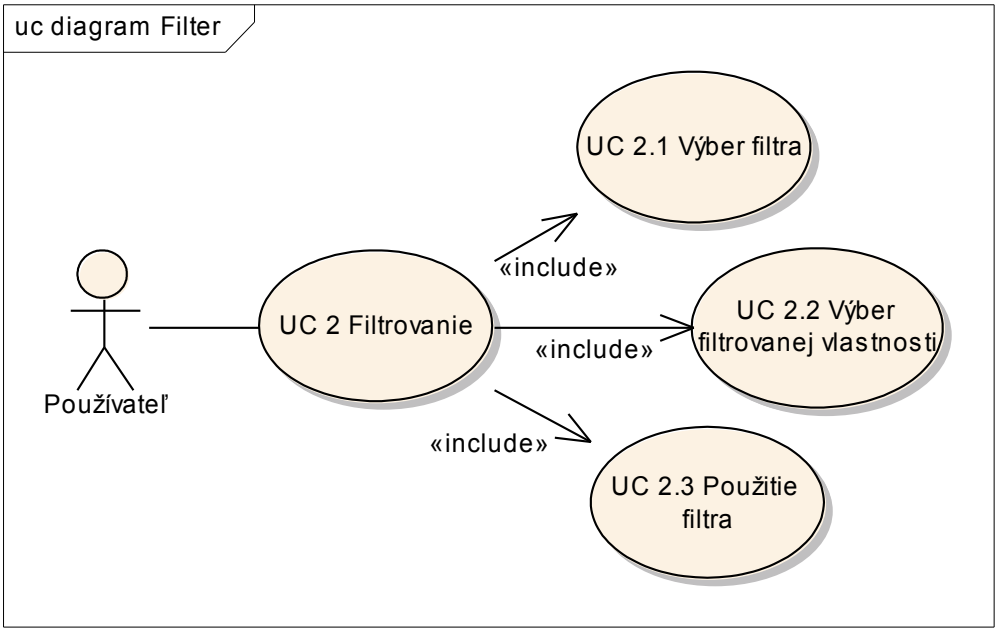


Obrázok 2

Identifikátor	UC 1		
Názov	Určenie vstupu		
Opis	Používateľ vyberie a nastaví vstupný zdroj údajov		
Priorita	1 = vysoká	**Frekvencia**	Pri každej analýze
Vstup. podm.	Používateľovi sa otvoril dialóg na výber a nastavenie vstupu		
Výstup. podm.	Vstupný zdroj údajov je korektne vybraný a nastavený		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ si zo zobrazeného dialógu vyberie typ vstupného formátu	
	2	Po nastavení formátu vyberie konkrétny údajový zdroj (Konkrétna databáza alebo výber súboru)	
	3	Následne nastaví podrobné nastavenia, ktoré sú závislé od konkrétneho vstupu	
	4	Po nastavení potvrdí výber a nastavenie	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Poznámky			

Filtrovanie

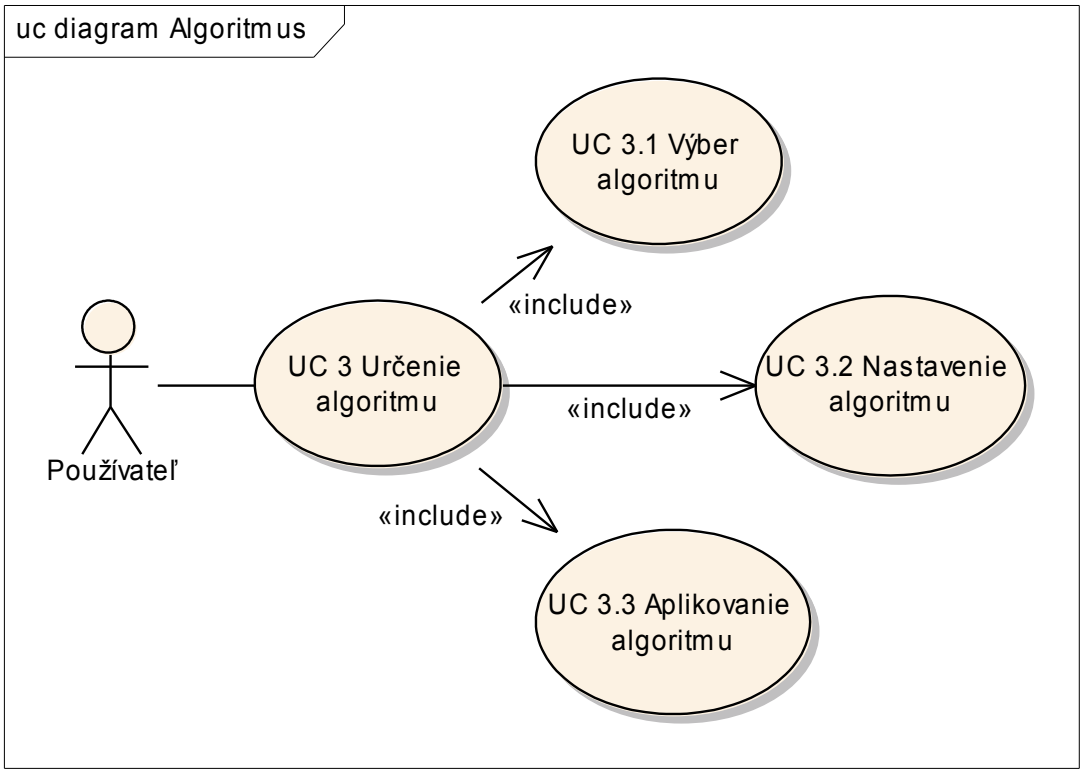
Pri filtrovaní si používateľ najprv zvolí typ filtra, ktorý chce aplikovať na vstupné údaje. Typ filtra je nezávislý na vstupnom zdroji údajov. Typ filtra určuje spôsob akým sa odoberú elementy zo vstupu. Po výbere typu filtra si používateľ vyberie filtrovanú vlastnosť. Táto vlastnosť je nezávislá na vstupnom zdroji údajov, ale je závislá od samotných údajov. Po výbere a nastavení filtra používateľ potvrdí nastavenie. V prípade použitia kartového sprievodcu sa filter použije nakoniec, po potvrdení všetkých kariet a to kvôli možnosti korekcie nastavení filtra v neskoršom kroku.



Obrázok 3

Identifikátor	UC 2		
Názov	Filtrovanie		
Opis	Filtrovanie siete podľa vybraného filtra		
Priorita	4 = nízka	**Frekvencia**	priemerná
Vstup. podm.	Úspešne vybraný a nastavený zdroj vstupných údajov		
Výstup. podm.	Sieť vyhovujúca zvolenému filtru		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	Po nastavení vstupu si používateľ môže vybrať filter	
	2	Zvolenie typu filtra	
	3	Výber vlastnosti, na základe ktorej sa filtruje	
	4	Nastavenie filtra (napr. rozsah)	
	5	Potvrdenie správnosti nastavenia	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Poznámky			

Určenie algoritmu spracovania



Obrázok 4

Po potvrdení zdroja a filtra si používateľ zvolí algoritmus, ktorý chce na sieť použiť. Výberom algoritmu sa určí, ktorý modul bude použitý na analýzu alebo iné operácie (napr. zhlukovanie) nad grafom. Po výbere algoritmu používateľ nastaví

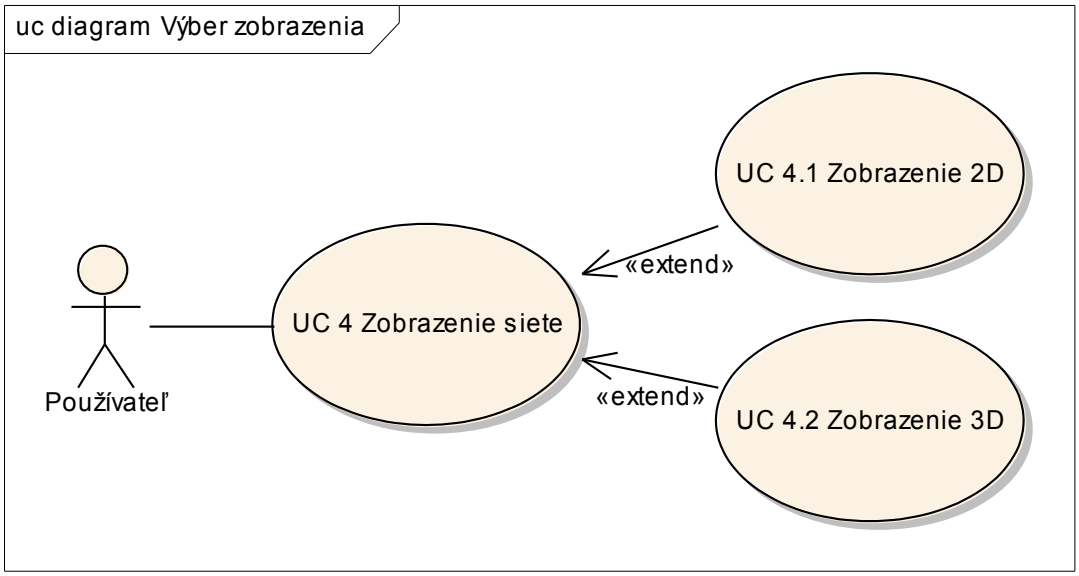
konkrétne vlastnosti pre daný algoritmus. Tieto vlastnosti sú závislé na použitom algoritme a sú pre každý algoritmus iné. Ak používateľ vyplní povinné nastavenia algoritmu, potvrdí ich kliknutím na tlačidlo ďalej. V prípade použitia kartového sprievodcu sa algoritmus spustí až po potvrdení poslednej karty.

Identifikátor	UC 3			
Názov	Algoritmus spracovania			
Opis	Aplikovanie algoritmu spracovania siete			
Priorita	1 = vysoká		**Frekvencia**	Pri každej analýze
Vstup. podm.	Úspešne nastavený vstup			
Výstup. podm.	Výsledok aplikácie algoritmu na sociálnu sieť			
Používatelia				
Základná postupnosť	**Krok**	**Činnosť**		
	1	Po nastavení vstupu a výberu filtra používateľ vyberie algoritmus spracovania		
	2	Používateľ vyberie niektorý(é) z importovaných algoritmov (v ponuke má aj možnosť nepoužiť algoritmus)		
	3	Po výbere sa mu zobrazia nastavenia pre daný algoritmus		
	4	Nastaví všetky povinné parametre a zvolené nepovinné		

	5	Nastavenie potvrdí a prejde na nastavovanie výstupu
Alternatívna postupnosť	**Krok**	**Činnosť**
Poznámky		

Výber zobrazenia

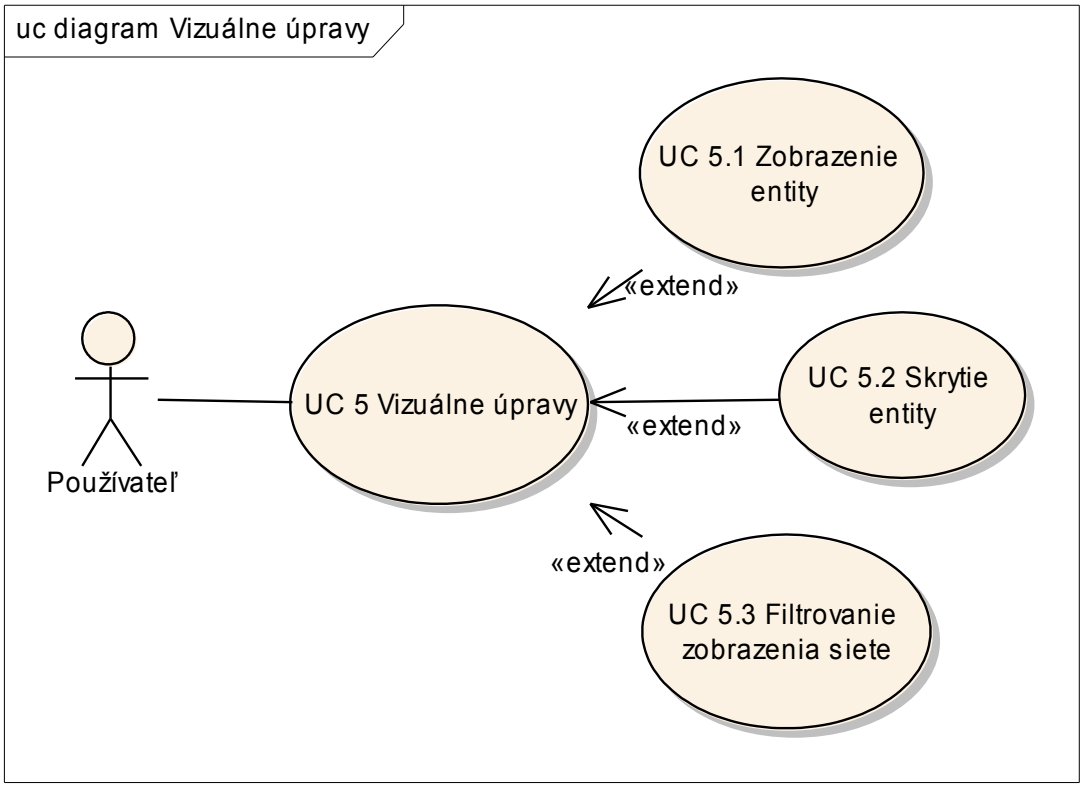
Počas práce so sieťou používateľ má možnosť sieť zobraziť vo forme 2D alebo 3D grafu. Toto zobrazenie si používateľ vyberie ako konkrétny typ výstupu grafu.



Obrázok 5

Identifikátor	UC 4		
Názov	Výber zobrazenia siete		
Opis	Grafické zobrazenie sociálnej siete vo forme 2D alebo 3D grafu		
Priorita	3 = stredná	**Frekvencia**	
Vstup. podm.	Úspešne vybraný vstup a načítané údaje		
Výstup. podm.	Zobrazená sociálna sieť vo forme 2D alebo 3D grafu		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	V časti výstup si používateľ zvolí typ výstupu zobrazenie	
	2	Po zvolení typu výstupu vyberie spôsob zobrazenia – 2D / 3D	
	3	Nastaví rozsah zobrazenia siete	
	4	Potvrdí výstup	
	5	Po potvrdení sa zobrazí sieť vo forme grafu v používateľskom rozhraní aplikácie	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Poznámky			

Vizuálna úprava grafu



Obrázok 6

Používateľ má možnosť meniť vizuálne zobrazenie grafu. Môže zobraziť alebo skryť hranu alebo vrchol. Toto zobrazenie alebo skrytie nezasahuje do štruktúry siete a slúži len na úpravu zobrazenia siete. Algoritmy aplikované na sieť sa aplikujú aj na skryté vrcholy a hrany. Používateľ môže filtrovať zobrazenie siete. Účel vizuálneho filtra je analogický so skrývaním elementov. Slúži len na upravenie zobrazenia pričom elementy, ktoré nevyhovujú filtru sa len skryjú, ale výpočty aplikované na sieť s nimi počítajú.

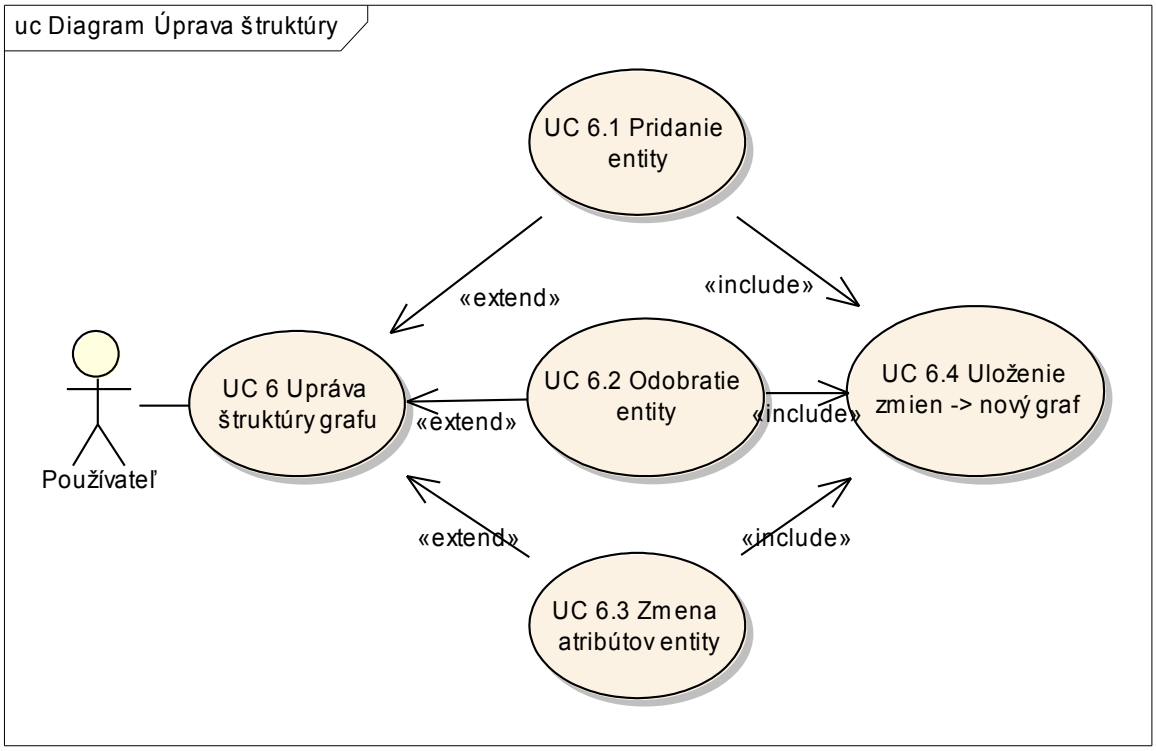
Identifikátor	UC 5		
Názov	Vizuálne úpravy siete		
Opis	Upravenie rozsahu a spôsobu zobrazenia sociálnej siete		
Priorita	1 = vysoká	**Frekvencia**	
Vstup. podm.	Úspešne načítané vstupné údaje a vizualizovanú sociálnu sieť		
Výstup. podm.	Upravená vizualizácia siete		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1 a	Systém zobrazí sieť v používateľskom rozhraní	
	2 a	Používateľ označí entitu siete (uzol / hrana)	
	3 a	Zvolí si možnosť zobrazenia alebo skrytia	

	4 a	Systém zobrazí sieť s/bez zvolenej entity
	5 a	
Alternatívna postupnosť	**Krok**	**Činnosť**
	1 b	Systém zobrazí sieť v používateľskom rozhraní
	2 b	Používateľ si zvolí typ vizuálneho filtra
	3 b	Potvrdí aplikovanie filtra
	4 b	Systém zobrazí sieť bez entít, ktoré nevyhovovali filtru
Poznámky		

Úprava štruktúry grafu vrámci zobrazenia

UC slúži na úpravu grafu cez jeho vizuálnu podobu. Tento UC sa líši od predchádzajúceho tým, že vzniká nový graf. Novovzniknutý graf je možné ďalej používať a aplikovať naň ostatné prípady použitia z procesu analýzy. Zmeny sa týkajú pridania, odobratia uzlov/hrán a zmien ich atribútov.

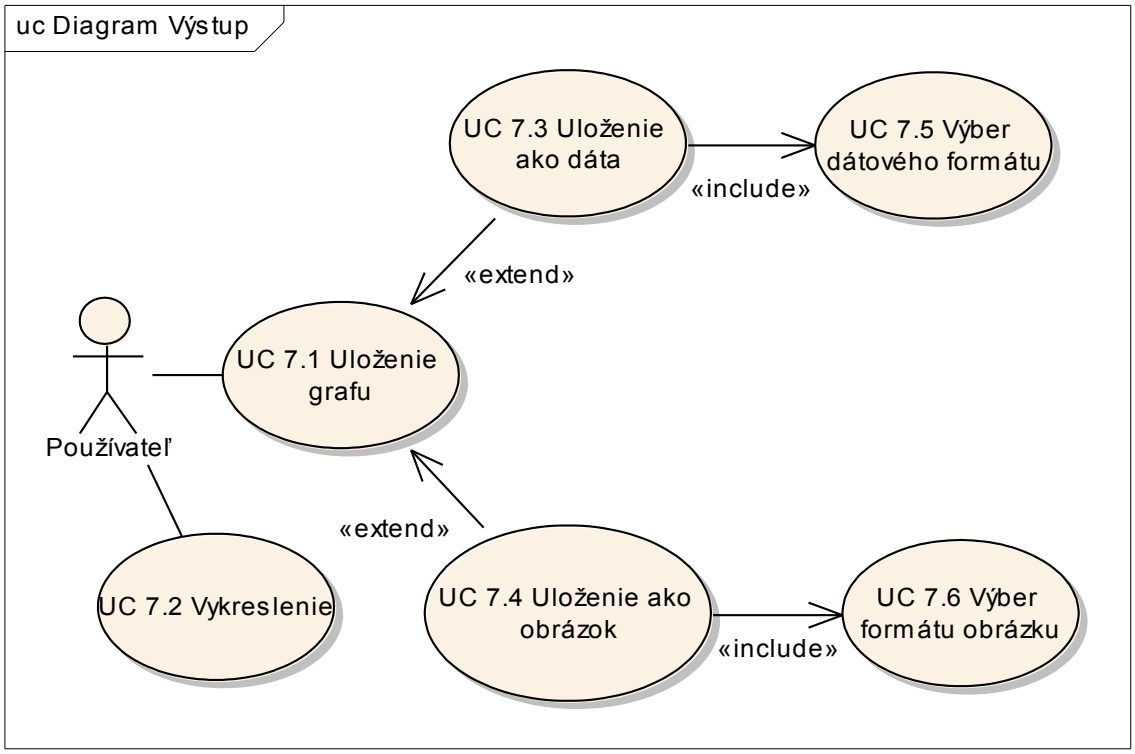
Obrázok 7



Identifikátor	UC6		
Názov	Vizuálna úprava štruktúry grafu		
Opis	Používateľ po vzhliadnutí grafu upravuje jeho štruktúru		
Priorita	4 = nízka	**Frekvencia**	nízka
Vstup. podm.	Vykreslený graf		
Výstup. podm.	Upravená štruktúra grafu		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	Systém vykreslí graf do jeho vizuálnej podoby	
	2	Používateľ označí entitu grafu, čiže uzol alebo hranu	
	3	Systém zobrazí v bočnom paneli atribúty vybranej entity	
	4	Používateľ zmení niektorý z atribútov	
	5	Systém uchová zmeny ako nový graf, ale aj ako kroky zmeny pre možnosť návratu	
	6	Používateľ pokračuje bodom 2 alebo aplikuje na nový graf ďalší UC(2-7)	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	2a	Používateľ vloží novú entitu	
	3a	Systém zobrazí v bočnom paneli atribúty novej entity.	
	4a	Používateľ špecifikuje atribúty	
	4b	Používateľ odoberie entitu	
Poznámky			

Výstup analýzy

UC slúži na uloženie internej reprezentácie grafu v jednom z podporovaných dátových formátov alebo ako obrázok.

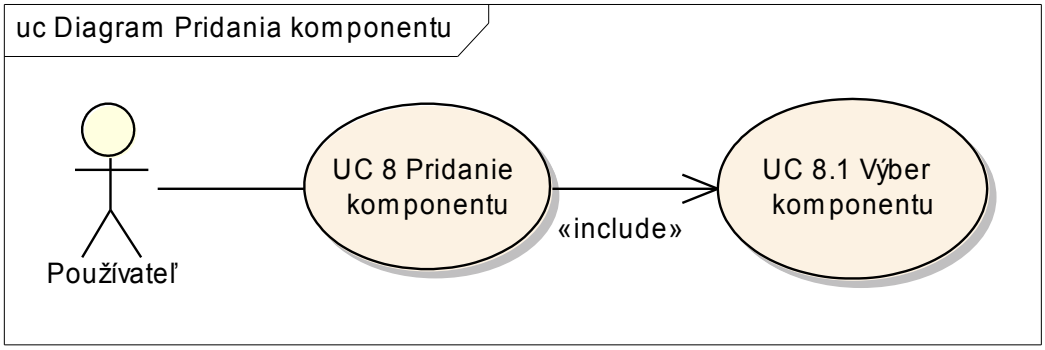


Obrázok 8

Identifikátor	UC7			
Názov	Uloženie výstupu			
Opis	Používateľ uloží výstup vo forme dát alebo obrázku			
Priorita	1 = vysoká		**Frekvencia**	Pri každej analýze
Vstup. podm.	Systém má načítaný graf v internej reprezentácií			
Výstup. podm.	Graf je zobrazený alebo uložený do súboru			
Používatelia				
Základná postupnosť	**Krok**	**Činnosť**		
	1	Používateľ si na 4. karte sprievodcu zvolí typ výstupu		
	2	Systém ponúkne podporované formáty výstupu pre daný typ		
	3	Používateľ si vyberie formát		
	4	Používateľ špecifikuje ďalšie atribúty výstupu		
	5	Systém vykoná sprievodcom definovanú operáciu výstupu ->vykreslí graf		
Alternatívna postupnosť	**Krok**	**Činnosť**		
	1a	Používateľ sa k ukladaniu do súboru dostane priamo a to buď cez menu alebo prostredníctvo tlačítka		
	5b	Systém vykoná sprievodcom definovanú operáciu výstupu ->uloží graf do súboru		
Poznámky				

Registrovanie nového komponentu

UC slúži na pridanie komponentu, či už sa jedná o algoritmus analýzy alebo vstupno-výstupný modul. Používateľ si vyberie cez dialógové okno daný jar súbor obsahujúci modul, systém ho nakopíruje do svojej štruktúry a zaregistruje si ho vo svojej konfigurácii. Následne je potrebný reštart pre úpravu ponuky menu a používateľského rozhrania.

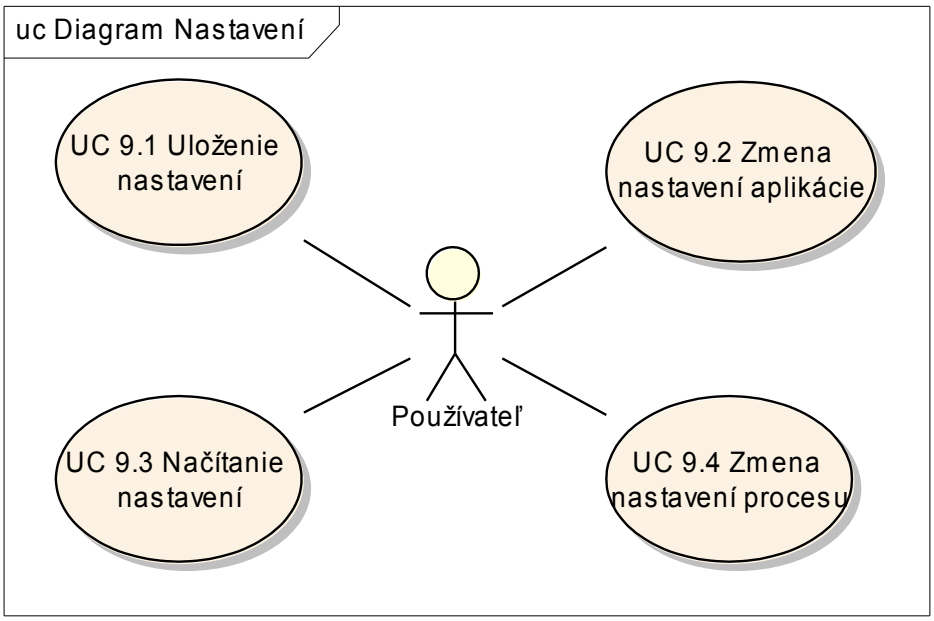


Obrázok 9

Identifikátor	UC8		
Názov	Pridanie komponentu		
Opis	Používateľ pridá do aplikácie nový komponent, rozšíri funkcionalitu		
Priorita	3 = stredná	**Frekvencia**	nízka
Vstup. podm.	V aplikácií nebeží žiaden proces analýzy		
Výstup. podm.	Aplikácia je rozšírená o nový modul		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ zvolí pridanie komponentu cez menu alebo tlačidlom	
	2	Otvorí sa modálne dialógové okno pre nájdenie daného komponentu	
	3	Používateľ si vyberie zvolený komponent	
	4	Systém ho skopíruje, zaregistruje a reštartuje sa	
	5	Používateľ má k dispozícii systém s rozšírenou funkcionalitou	
Poznámky			

Nastavenia

Používateľ má možnosť cez menu alebo cez tlačítko vyvolať dialóg nastavení. Môže v ňom deaktivovať moduly, nastaviť automatické ukladanie, alebo editovať posledné nastavenie procesu analýzy teda hlavne filter a použitý algoritmus. Tieto nastavenia môže uložiť do špecifikovaného súboru, alebo ich načítať.

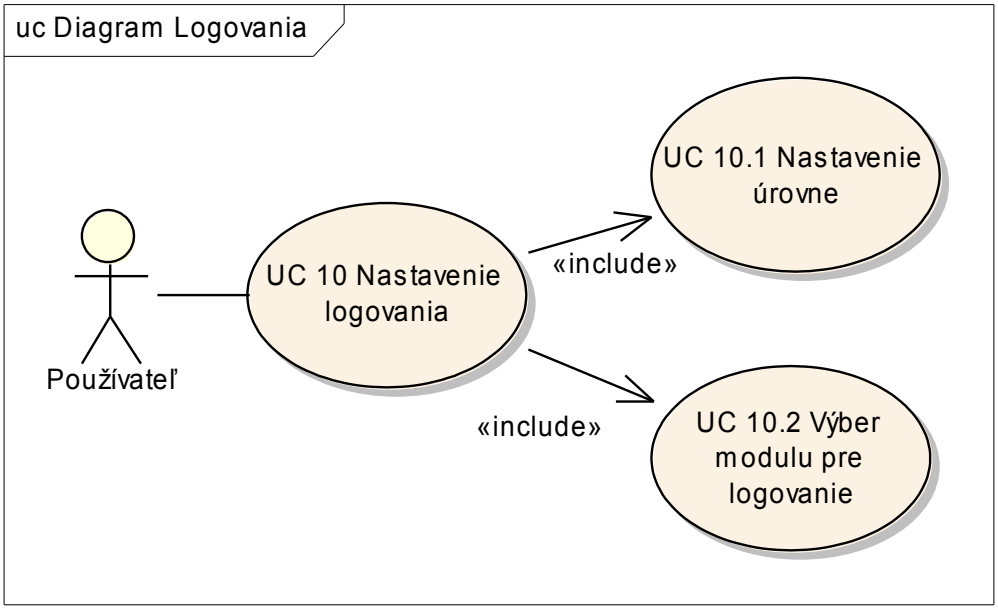


Obrázok 10

Identifikátor	UC9		
Názov	Zmena konfigurácií		
Opis	Používateľ má možnosť manuálne konfigurovať aplikáciu ako aj proces analýzy		
Priorita	3 = stredná	**Frekvencia**	nízka
Vstup. podm.	Nie sú		
Výstup. podm.	Upravené nastavenia.		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ zvolí cez menu alebo cez tlačítko dialóg nastavení	
	2	Používateľ si zvolí systémové nastavenia	
	3	Systém zobrazí zoznam modulov a možnosť automatického ukladania so zaškrtávacími poľami	
	4	Používateľ editáciou zaškrtávacích polí povoľuje alebo zakazuje jednotlivé funkcionality	
	5	Používateľ potvrdí nastavenia	
	6	Systém zobrazí dialóg na uloženie nastavení	
	7	Používateľ definuje konfiguračný súbor	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	2b	Používateľ si zvolí nastavenia procesu analýzy	
	3b	Systém zobrazí vybraný vstup, filter, algoritmus, výstup a ich parametre	
	4b	Používateľ edituje nastavenia	
	2c	Používateľ si zvolí načítanie nastavení zo súboru	
	3c	Systém zobrazí dialóg na uloženie nastavení	
	4c	Používateľ definuje vstupný konfiguračný súbor	
Poznámky			

Logovanie

Používateľ má možnosť nastaviť pre niektorý z dostupných modulov logovanie. Taktiež má možnosť špecifikovať pre vybraný modul aplikácie úroveň logovania, napríklad iba výsledok algoritmu alebo aj postup výpočtu a výstupný log súbor na ďalšej záložke



Obrázok 11

Identifikátor	UC10		
Názov	Natavenie logovania		
Opis	Používateľ má možnosť definovať modul a úroveň pre logovanie		
Priorita	5 = nízka	**Frekvencia**	nízka
Vstup. podm.	Dostupný aspoň jeden logovací modul		
Výstup. podm.	Logovanie je špecifikované		
Používatelia			
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ zvolí cez menu alebo cez tlačítko dialóg logovania	
	2	Systém zobrazí dostupné moduly pre logovanie	
	3	Požívateľ si vyberie jeden z dostupných modulov	
	4	Používateľ na ďalšej záložke vyberie úroveň logovania a názov pre výstupný log súbor.	
	5	Používateľ potvrdí nastavenia	
Poznámky			

3.3. Nefunkcionálne požiadavky

Keďže výsledný produkt je určený pre ľudí, ktorý sa vyznajú v problémovej oblasti analýzy sociálnych sietí, zároveň má poskytovať možnosti rozšírenia jeho častí a poskytovať základné API pre používateľa, vznikli aj konkrétne nefunkcionálne požiadavky. Tieto požiadavky sú:

Modulárnosť – výsledný produkt je modulárny. Oddelené od seba sú všetky časti, ktoré spolu funkcionálne nesúvisia. Modularizovanie je vykonané najmä v nasledujúcich aspektoch:

- oddelenie zobrazovacej časti od výkonnej
- oddelenie časti predspracovania vstupných údajov
- oddelenie časti výstupných údajov
- oddelenie časti algoritmov analýz
- oddelenie časti pre prácu so sociálnou sieťou (neanalyzačné algoritmy, zhlukovanie)
- oddelenie časti používateľského rozhrania a vizualizácie

Dynamické pridávanie modulov – aplikácia podporuje dynamické pridávanie modulov bez potreby opätovného kompilovania.

Pridanie nového modulu je možné vykonať používateľskou akciou v aplikácii.

Dynamicky je možné pridať tieto moduly:

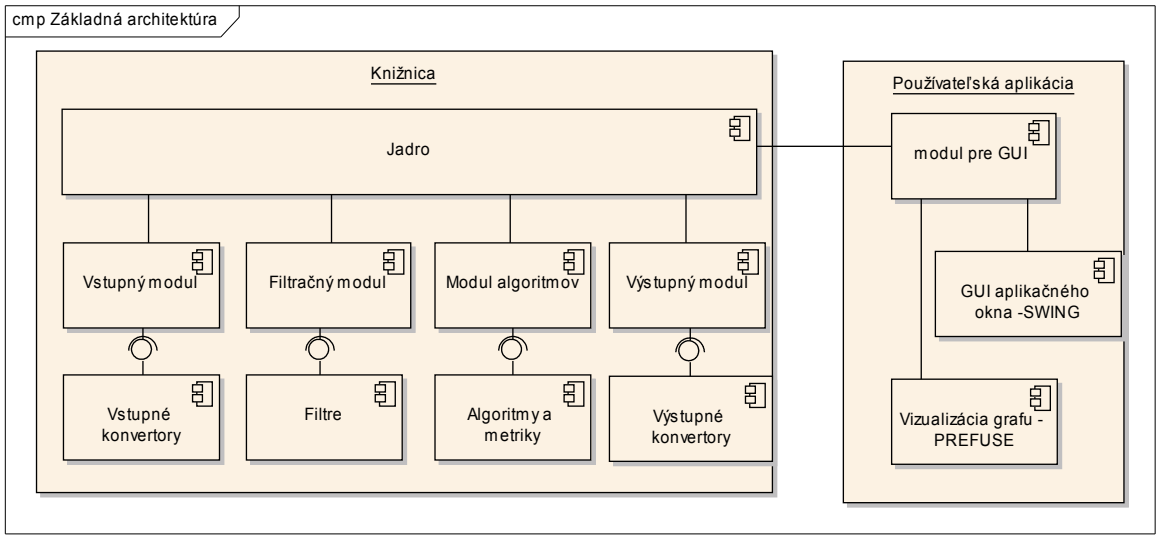
- analyzátor sociálnych sietí
- modul načítania a predspracovania údajov. (Tým je dosiahnutá podpora viacerých vstupných formátov a možnosť pridať kompatibilitu s novými formátmi)
- modul exportovania výstupných údajov. (Tým je dosiahnutá podpora viacerých výstupných formátov údajov a možnosť vytvoriť kompatibilitu s novými formátmi)

Jadro aplikácie – slúži ako funkcionálny základ a súbor nástrojov pre analyzovanie sociálnych sietí.

- je vytvorené vo forme JAVA knižnice
- dynamické pridávanie modulov a prácu s modulmi realizuje práve táto knižnica.
- neobsahuje prvky zobrazovania
- rozhranie knižnice je dokumentované samostatnou používateľskou príručkou alebo samostatnou časťou používateľskej príručky

4. Návrh

4.1. Návrh architektúry



Obrázok 12: Základná schéma architektúry

Základným cieľom navrhnutej architektúry je zabezpečiť znovupoužiteľnosť častí nášho systému v podobe knižnice. Knižnica obsahuje hlavnú funkcionalitu pre analyzovanie sociálnych sietí ako aj možnosť rozšíriť ju pridaním ďalších modulov. Z tohto dôvodu je nutné striktné oddelenie používateľského rozhrania od zvyšku systému, kde predpokladáme, že jadro nášho systému bude možné použiť aj v iných aplikáciách a systémoch, ktoré budú disponovať vlastným riešením používateľského rozhrania.

Vrámci knižnice sme identifikovali nasledujúce komponenty:

Jadro knižnice

- **Vykonáva logovanie** – obsahuje nastaviteľnú úroveň, po ktorú sa má aplikácia logovať. Loguje sa vnútorná činnosť (spustenie analýzy, načítanie siete), ale aj práca s modulmi a ich činnosť.
- **Sprostredkováva komunikáciu medzi modulmi** – aplikácia tvorí komunikačnú kostru pre jednotlivé moduly, čiže vymieňa potrebné údaje a sprostredkováva prenos správ medzi modulmi.
- **Určuje postupnosť vykonávania analýzy a spúšťa jednotlivé moduly** – Jadro určuje postupnosť jednotlivých modulov v akom poradí sa budú spúšťať pri konkrétnej práci so sieťou, sprostredkuje nastavenia modulov a stará sa o ich chod.
- **Ošetruje chyby modulov a chyby vnútornej funkcionality** – jadro zabezpečuje bezpečný chod aplikácie a odchytáva chyby vznikajúce v moduloch. V prípade

chyby informuje používateľskú aplikáciu o vzniknutej chybe a zabezpečí stabilný chod aplikácie aj po vzniknutí tejto chyby.

- **Poskytuje rozhranie pre vykonanie analýzy** – jadro knižnice obaľuje funkcionalitu všetkých modulov a poskytuje aplikačné rozhranie vo forme knižnice pre nastavenie a vykonanie analýzy sociálnych sietí.
- **Obsahuje vnútornú reprezentáciu načítanej sociálnej siete** – v jadre sa nachádza aj interná reprezentácia analyzovaných údajov v podobe grafu. Nad touto štruktúrou sa vykonávajú algoritmy a slúži len ako dočasný formát počas spracovania. Konverziou externých formátov do formátu internej reprezentácie sa zaoberá modul vstupu.

Vstupný modul

- zabezpečuje prenos vstupných údajov z externých zdrojov. V prípade databázy stiahne údaje a ponúkne ich na spracovanie.
- konvertuje externé formáty do formátu internej reprezentácie grafu. Po načítaní údajov ich konvertuje do podoby v akej sa spracovávajú.

Filtračný modul

- ponúka filtre na pretriedenie vstupných údajov pred použitím konkrétneho algoritmu spracovania. Po aplikovaní filtra na vstupné údaje sa zredukujú entity v internej reprezentácii a zo štruktúry sa odoberú tie, ktoré nevyhovujú filtru. Toto pretriedenie je možné definovať na základe vlastností vstupných údajov.

Modul algoritmov

- obsahuje algoritmy spracovania, ktoré je možné aplikovať na graf. Všetky algoritmy pracujú nad formátom internej reprezentácie a výsledok ukladajú ako vlastnosti jednotlivých entít grafu.

Výstupný modul

- zabezpečuje konverziu interného formátu reprezentácie grafu do externých formátov alebo do vizuálnej podoby. Po konverzii týchto formátov ich uloží podľa používateľom nastavených preferencií.

Používateľská aplikácia

- ponúka používateľské rozhranie na prácu s analyzátorom.
- sprostredkúva používateľovi manažment modulov
- umožňuje obsluhovať nastavenie jadra analyzátora
- umožňuje obsluhovať nastavenie jednotlivých modulov analyzátora
- umožňuje spúšťať nastavenú analýzu siete
- vizualizuje výsledky spracovania siete

4.2. Návrh používateľského rozhrania

Hoci hlavným cieľom pri tvorbe aplikácie je vytvoriť univerzálny základ pre analýzu sociálnych sietí vo forme knižnice, snažíme sa, aby bola využiteľná aj ako samostatná aplikácia. Pre naplnenie tohoto cieľa sme vytvorili grafické používateľské

rozhranie. Popis jeho najdôležitejší ch častí bude ďalej nasledovať v tejto kapitole.

Popis sprievodcu

Keďže proces analýzy zahŕňa niekoľko krokov, rozhodli sme sa vytvoriť sprievodcu, ktorý používateľa týmto procesom prevedie.

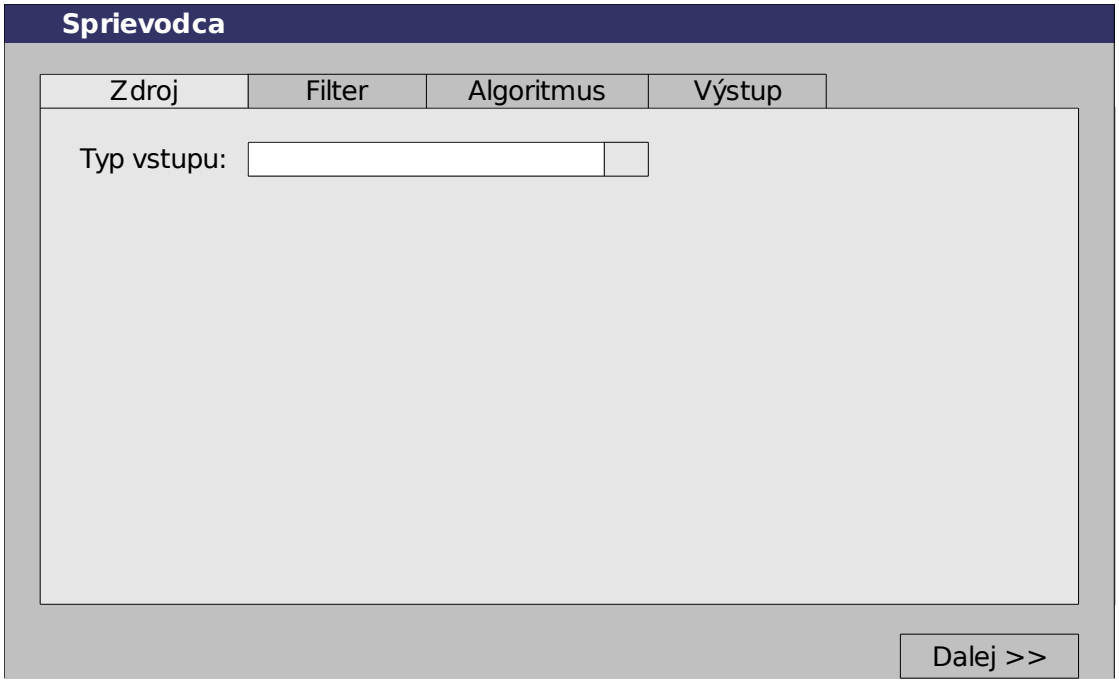
Sprievodca má štyri základné časti, každá z nich je zobrazená v samostatnej záložke.

- Zdroj
- Filter
- Algoritmus
- Výstup

Sprievodca poskytuje aj tlačidlá ďalej a späť, ale aplikáciu je možné ovládať aj na nich nezávisle, iba pomocou záložiek. V ďalšej časti sú popísané jednotlivé záložky.

Zdroj

Po spustení sprievodcu sa zobrazí obrazovka, ktorú je vidieť na obrázku 13. V položke typ súboru ešte nie je zvolená žiadna možnosť.

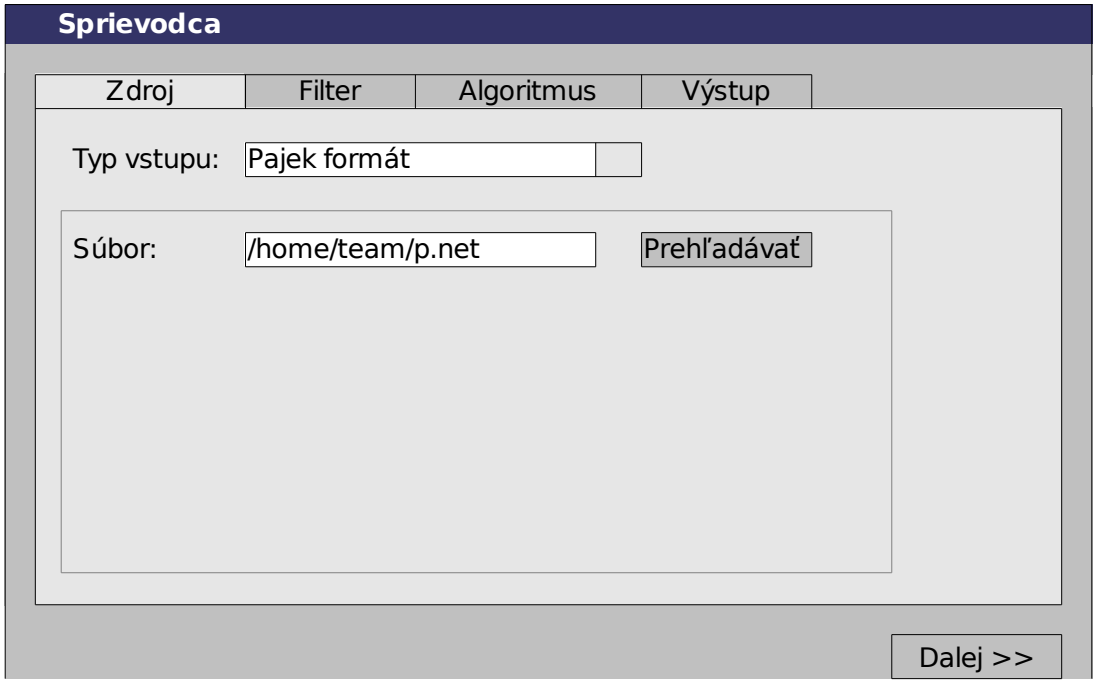


Obrázok 13: obrazovka po spustení sprievodcu

Tu si používateľ môže vybrať požadovaný typ vstupných dát.

Na základe tohto výberu sa mu v dolnej časti obrazovky zobrazia dodatočné nastavenia pre jednotlivé typy dát.

Na obrázku 14 je vidieť obrazovka po výbere možnosti pajek formátu. Tu je možné nastaviť cestu ku vstupnému súboru.

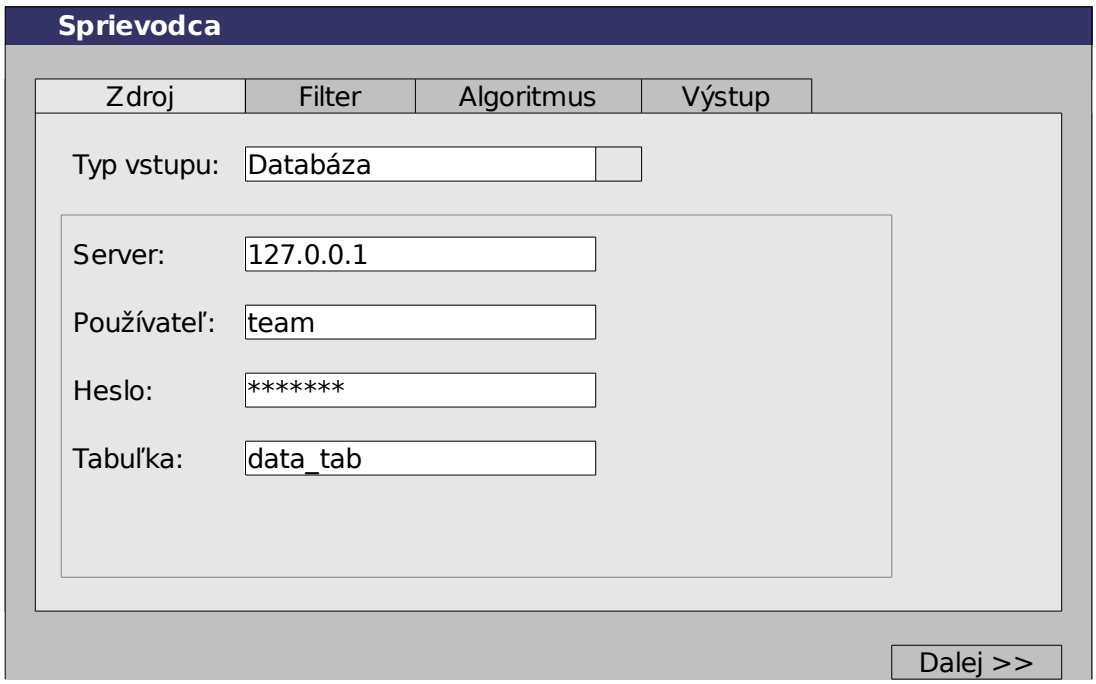


Obrázok 14: po výbere formátu

Na obrázku 15 je zobrazená obrazovka po výbere možnosti načítania dát z databázy. Tu je možnosť nastaviť:

- **Server:** adresa databázového servera
- **Používateľ:** používateľské meno
- **Heslo:** heslo používateľa
- **Tabuľka:** tabuľka na danom databázovom serveri, ktorá obsahuje vstupné dáta

Po kliknutí na tlačidlo *Ďalej* sa prepne pohľad na druhú záložku, na záložku **Filter**.

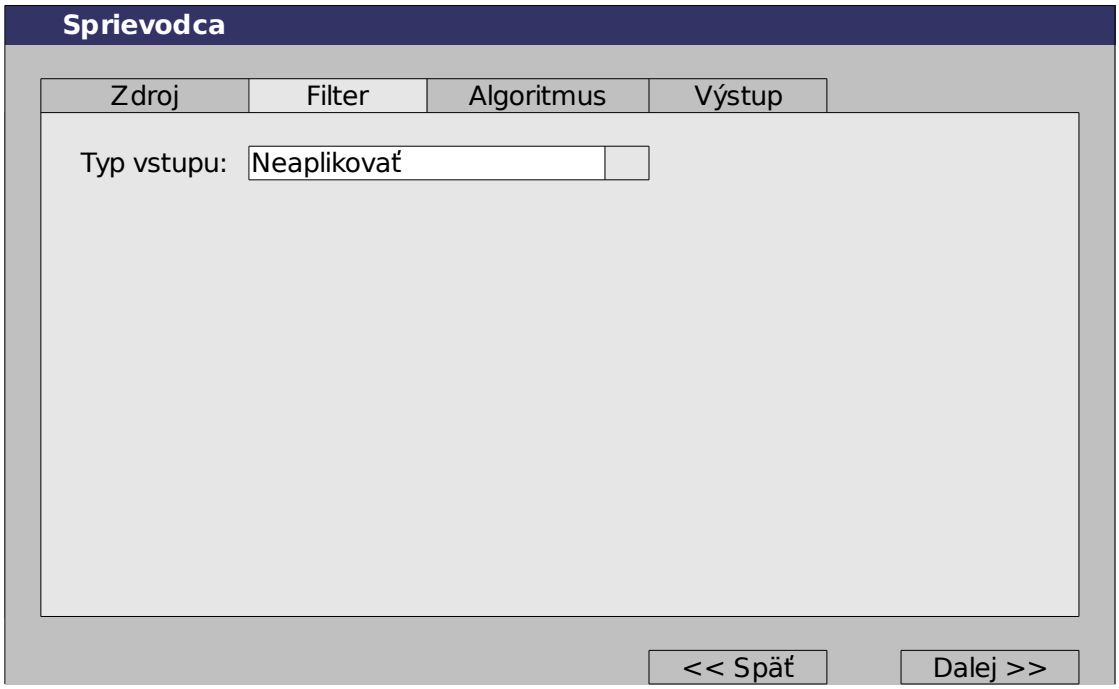


Obrázok 15: po výbere možnosti databáza

Filter

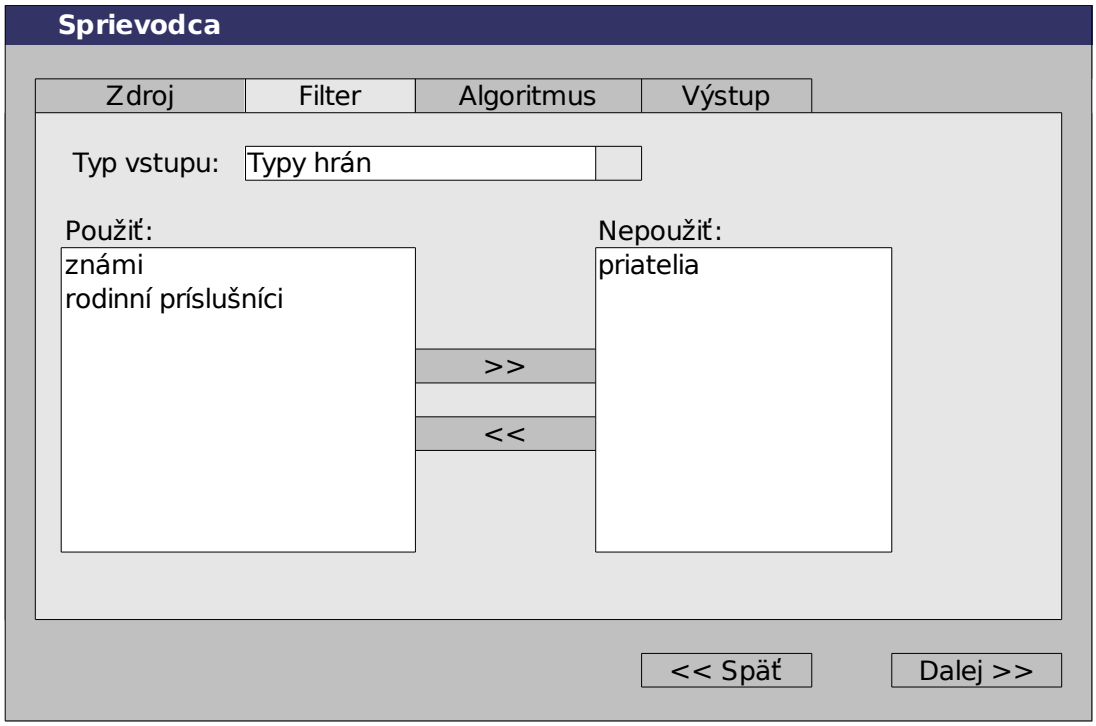
V tejto záložke je možné nastaviť filtre pre načítané dáta. Ide o nastavenie atribútov, ktoré majú byť zahrnuté do analýzy.

Pri prvom prechode na túto záložku (Obrázok 16) je prednastavený filter *Neaplikovať*. Túto voľbu je možné ponechať, v takom prípade sa do analýzy zahrnú všetky dáta.



Obrázok 16: záložka filter

Po výbere jedného z možných filtrov sa zobrazia jeho ďalšie nastavenia. Na obrázku 17 je uvedený iba jeden príklad, nakoľko v súčasnej dobe nie sú známe presné popisy podporovaných filtrov.



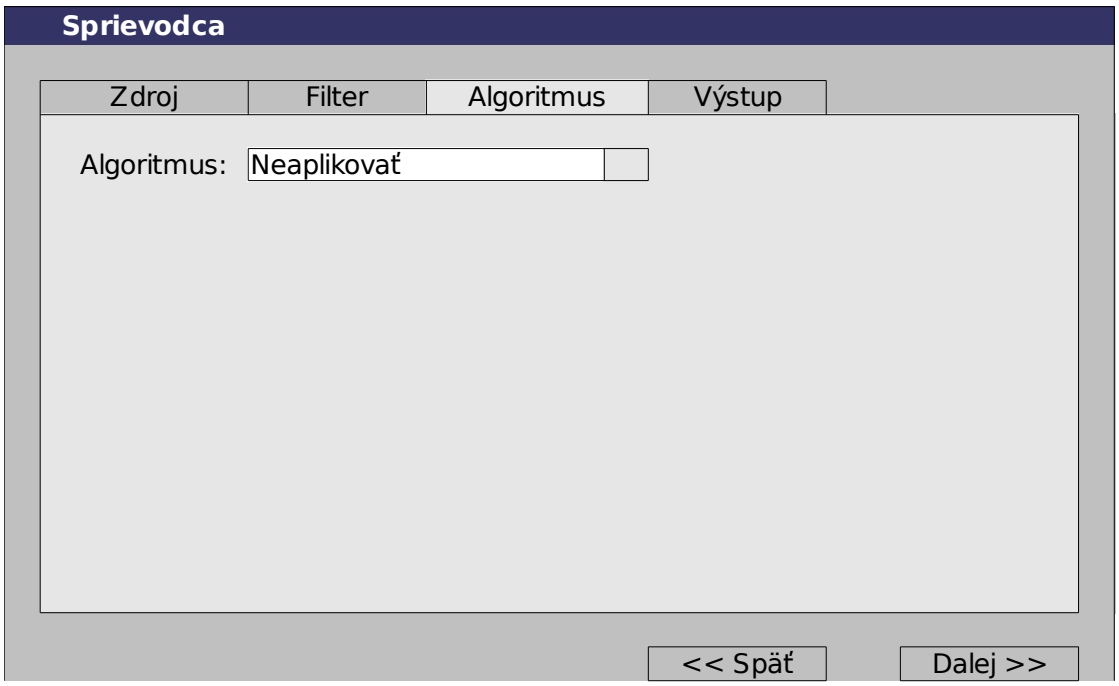
Obrázok 17: po vybratí jedného z filtrov

Po kliknutí na tlačidlo ďalej sa zobrazenie prepne na ďalšiu záložku, na záložku **Algoritmus.**

Algoritmus

V tejto záložke je možné nastaviť algoritmy, ktoré sa aplikujú na načítané a filtrované dáta.

Pri prvom prechode na túto záložku (Obrázok 18) je prednastavená možnosť N*epoužiť algoritmus*. Túto voľbu je možné ponechať, v takom prípade sa algoritmus neaplikuje. Po výbere jedného z podporovaných algoritmov sa zobrazia jeho ďalšie možné nastavenia. Tie na tomto mieste neuvádzame, nakoľko sú pre každý z algoritmov špecifické.



Obrázok 18: algoritmus

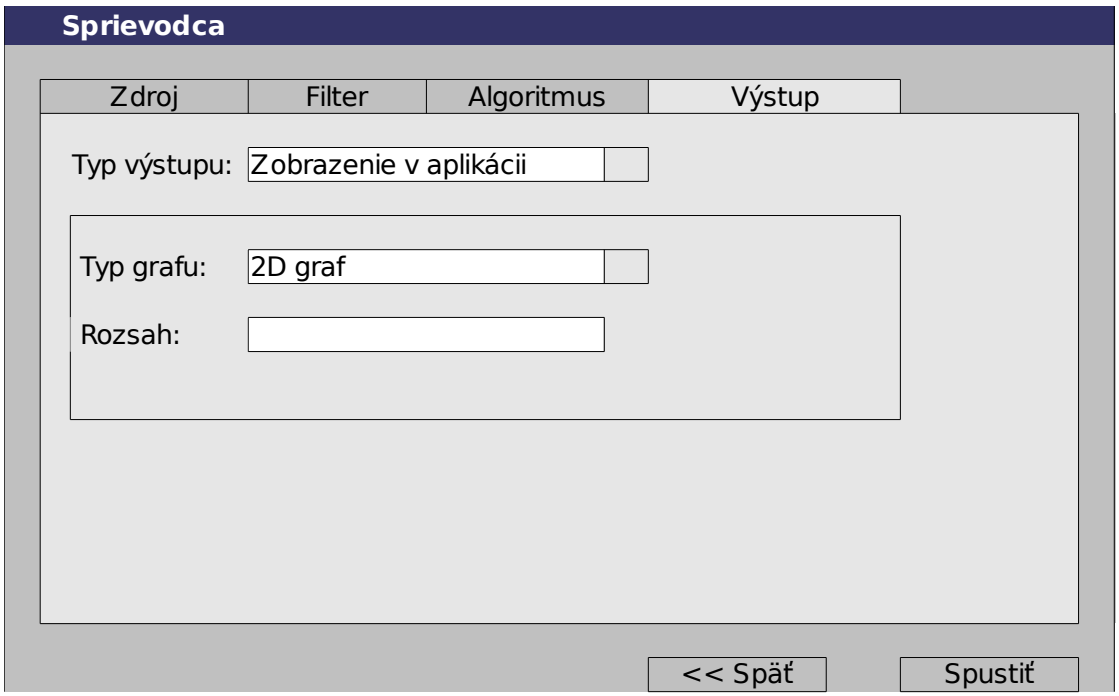
Po kliknutí na tlačidlo ďalej sa zobrazenie prepne na ďalšiu záložku, na záložku **Výstup.**

Výstup

V tejto záložke je možné si nastaviť typ výstupu. Po načítaní tejto záložky je prednastavená hodnota Zobrazenie v aplikácii (Obrázok 19). Ku tejto možnosti sa zobrazia doplňujúce možnosti

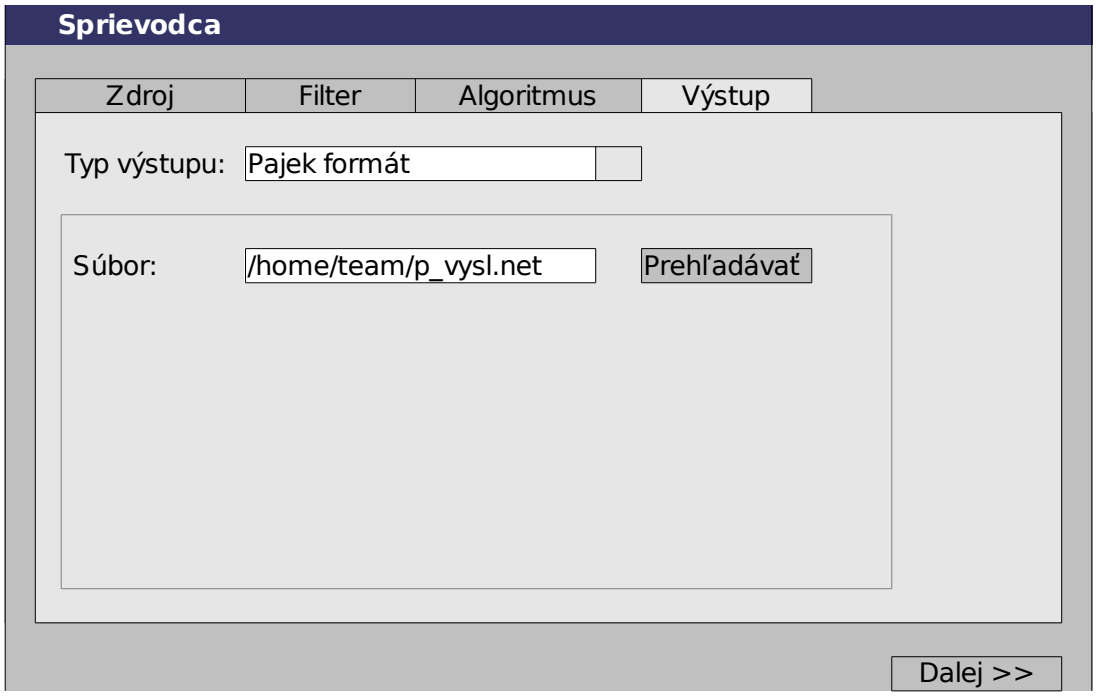
- výstup ako 2D graf
- výstup ako 3D graf

a rozsah zobrazenia grafu.



Obrázok 19: výstup zvolený v aplikácii

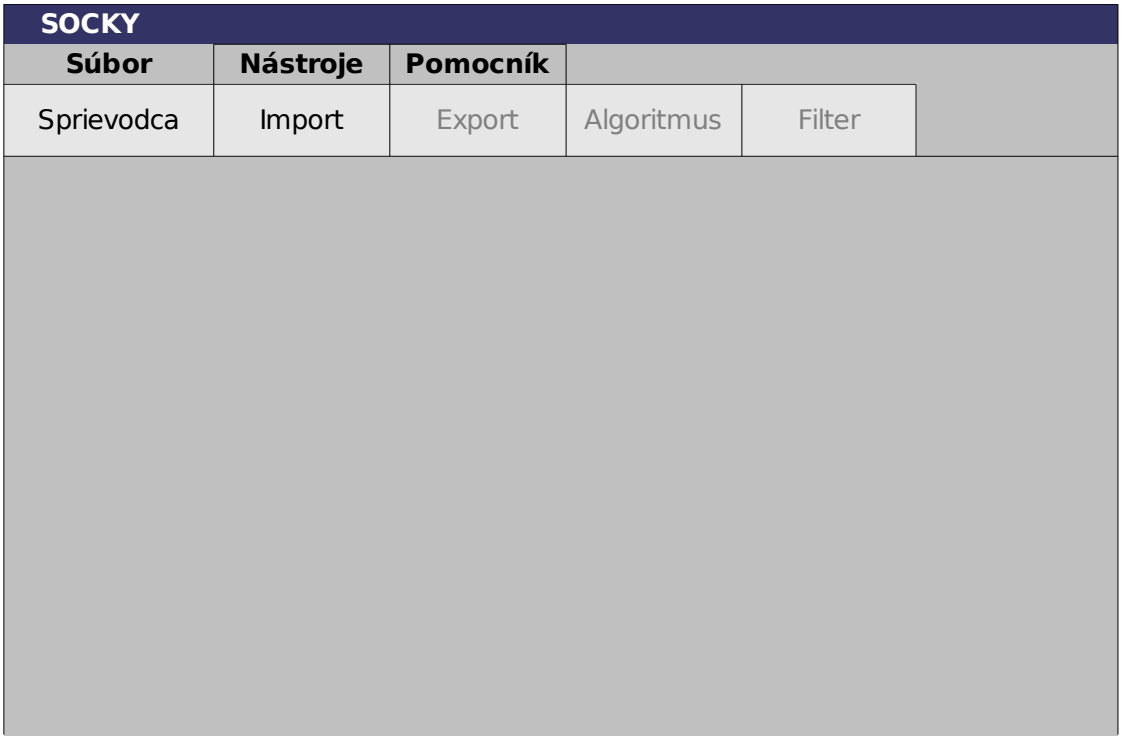
Druhou možnosťou je výstup vo formáte pajek (Obrázok 20). V tomto prípade sa zobrazí ako doplňujúca možnosť zadať cestu k súboru, do ktorého sa má výstup vygenerovať.



Obrázok 20: výstup zvolený pajek formát

Popis hlavného okna aplikácie

Hlavné okno aplikácie tak, ako vyzerá po spustení je vidieť na obrázku 21.



Obrázok 21: hlavné okno aplikácie

Sú tu možnosti

- Súbor
- Nástroje
- Pomocník

Súbor obsahuje

- Uložiť výstup
- Načítať vstup
- Vyvolať sprievodcu
- Ukončiť

Nástroje obsahuje

- Aplikovať filter
- Aplikovať analýzu
- Uložiť nastavenia aplikácie

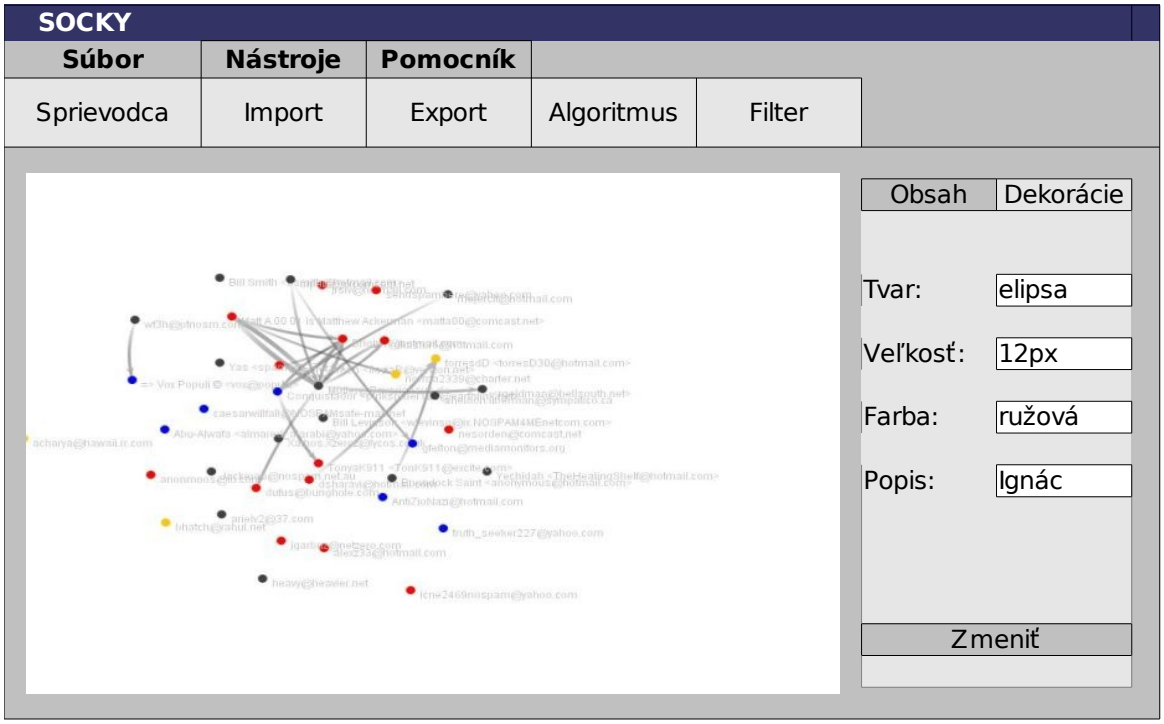
Pomocník obsahuje

- O programe
- Používateľská príručka

Hlavné okno aplikácie tiež obsahuje zrýchlené voľby na

- Sprievodca
- Export
- Import
- Algoritmus
- Filter

Ak sa skončí analýza a výsledok sa vizualizuje, tak sa zobrazí v tomto hlavnom okne. Príklad takéhoto zobrazenia 2D grafu je vidieť na obrázku 22.



Obrázok 22: Okno hlavnej aplikácie po zobrazení grafu

Do okna sa pridala bočná lišta s dvoma záložkami, kde je možné nastavovať vlastnosti grafu.

Na obrázku 22 je zvolená záložka Dekorácie a zvolený nejaký vrchol. V tejto je možné nastaviť

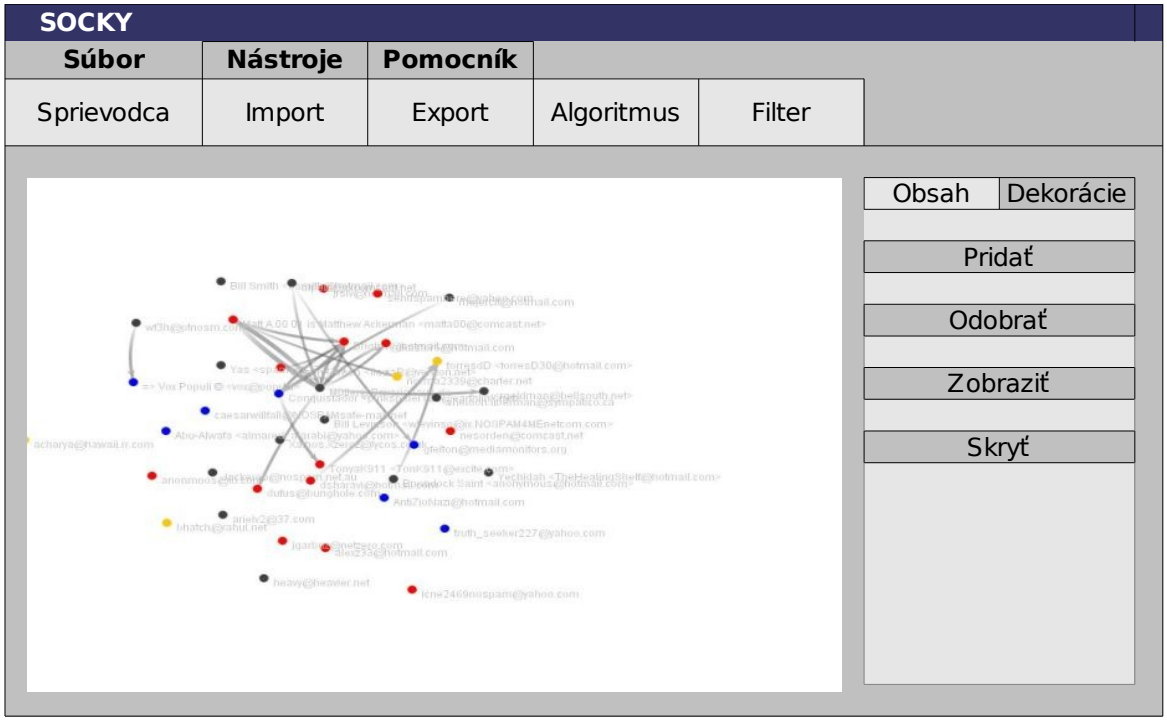
- tvar
- veľkosť
- farbu

Za predpokladu, že je zvolená nejaká hrana sa v záložke Dekorácie zobrazia nasledujúce možnosti:

- farba
- hrúbka

Na obrázku 23 je zobrazené používateľské rozhranie so zvolenou záložkou Obsah. Na tejto záložke sú nasledujúce možnosti:

- pridať (po stlačení sa zobrazí okno s možnosťou nastavenia novej hrany alebo vrcholu)
- odobrať (odoberie sa zvolená hrana alebo vrchol zo štruktúry grafu. Ak nie je nič zvolené, tak je zakázaná)
- zobraziť (zobrazia sa všetky skryté entity)
- skryť (skryje sa zvolená hrana alebo vrchol zo štruktúry grafu. Ak nie je nič zvolené, tak je zakázaná)



Obrázok 23: Hlavné okno aplikácie so zvolenou možnosťou Obsah

4.3. Použité technológie

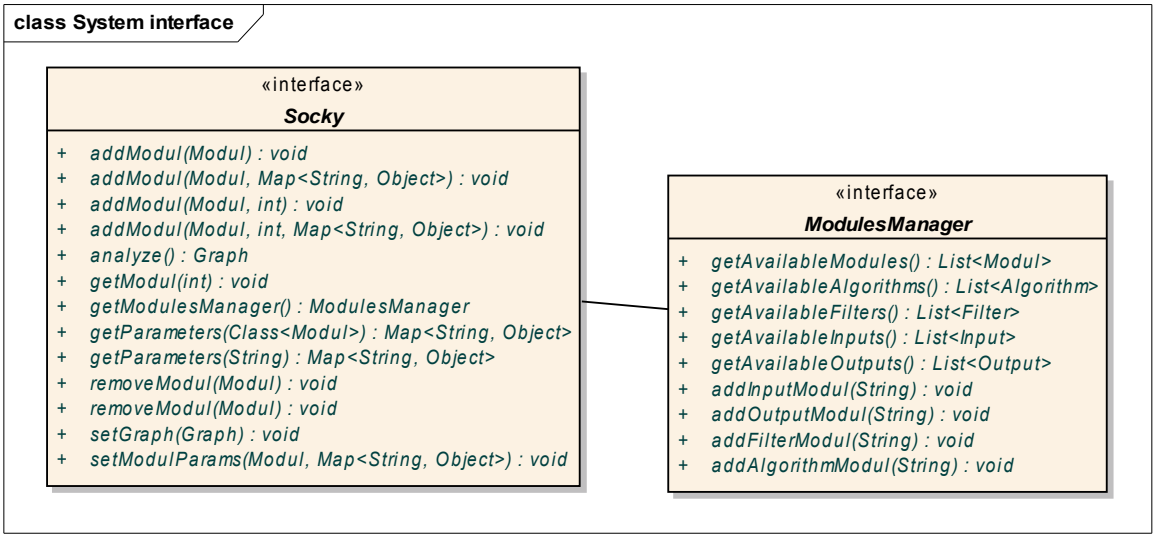
- Java 1.6
- SWING
- PREFUSE
- JUNG
- Log4j

5. Podrobný návrh

Návrh v kapitole 4.1. bol len hrubým návrhom architektúry systému. Keďže systém má byť modulárny a používateľ musí mať umožnené pridávanie vlastných modulov, musí mať systém jednoznačne definované rozhrania. Pri zjemňovaní návrhu sme sa teda sústredili najmä na dôkladný popis rozhraní systému. Tento krok považujeme za veľmi dôležitý najmä z toho dôvodu, aby bol systém pohodlne použiteľný rovnako pre tvorcu grafického používateľského rozhrania, tak aj pre programátora vytvárajúceho externú aplikáciu. Zároveň sme sa však snažili zachovať zapuzdrenie celého systému, aby používateľ týchto rozhraní nemohol vlastnou chybou spôsobiť jeho nestabilitu.

Rozhrania systému

Pri tvorbe rozhrania k systému sme vzali do úvahy, že v budúcnosti môžu k nemu pribúdať ďalšie funkcionality, preto sme sa rozhodli do hlavného rozhrania umiestniť iba metódy na prácu so samotným pracovným tokom (angl. work flow) a všetky ďalšie funkcionality (napríklad práca s modulmi) budú riešené formou samostatných rozhraní, ktoré bude hlavné rozhranie iba sprístupňovať. Rozhrania k systému sú zobrazené na obrázku 24.



Obrázok 24: Rozhrania k systému

Hlavné rozhranie

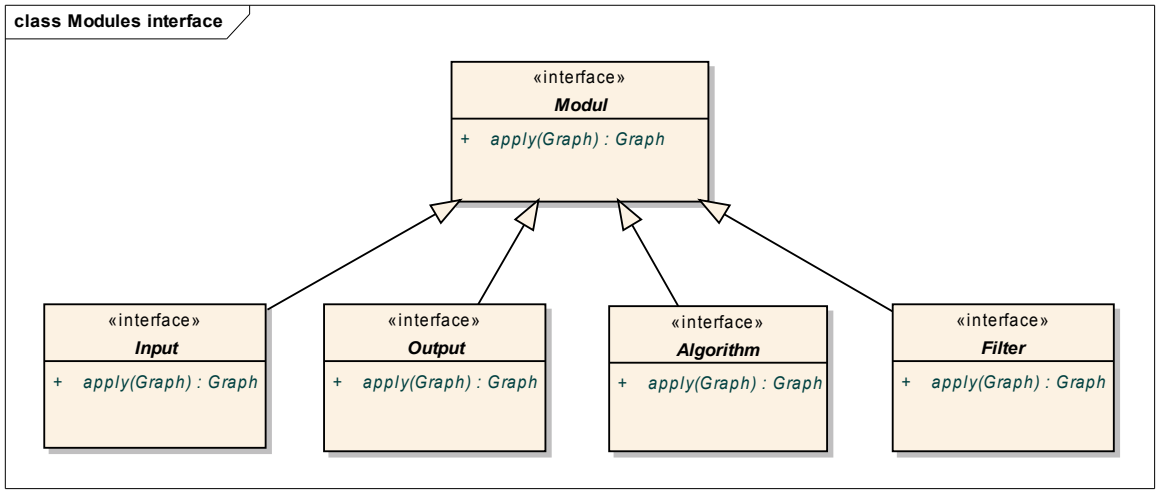
Na obrázku 24 vľavo je zobrazené hlavné rozhranie systému. Obsahuje operácie na pridanie modulu, jeho odobratie i nastavenie parametrov jednotlivým modulom. Toto rozhranie tiež umožňuje priame nastavenie grafu systému.

Rozhranie práce s modulmi

Na obrázku 24 vpravo je zobrazené rozhranie na prácu s modulmi. Toto rozhranie poskytuje informácie o všetkých moduloch, ktoré sú práve k dispozícii, zároveň umožňuje pridať nový modul špecifikovaním cesty k jeho JAR súboru.

Rozhrania modulov

V systéme rozlišujeme štyri rôzne moduly: vstupný modul, výstupný modul, modul algoritmov a modul filtrov. Napriek tomu, že ponúkajú rôznu funkcionalitu, snažili sme sa vytvoriť spoločné rozhranie, z ktorého budú rozhrania pre jednotlivé moduly dediť (obrázok 25). To nám umožní využiť rovnaké triedy na načítanie modulov, načítanie ich parametrov i na vytvorenie pracovného toku.



Obrázok 25: Rozhranie modulov

6. Prototypovanie

V rámci prototypovania sme sa rozhodli ujasniť si niektoré nezrovnalosti, prípadne si overiť funkčnosť a efektívnosť navrhovaných riešení. Keďže sme mali viacero otázok a nejasností rozhodli sme sa prototypovať hneď niekoľko častí systému.

1. Vykresľovanie knižnicami JUNG a Prefuse
2. Dynamické načítavania modulov
3. Dynamické nastavovanie parametrov a generovanie obrazoviek
4. Používateľské rozhranie

Niektoré prototypované časti sme spojili, niektoré slúžili len na otestovanie ich možností a sú prototypom na zahodenie.

6.1. Vykresľovanie knižnicami JUNG a Prefuse

Prototyp vizualizačných knižníc pozostáva z dvoch aplikácií, ktoré zobrazujú 3 typy grafov s variabilným počtom vrcholov a hrán. Typy grafov, ktoré aplikácie zobrazujú:

- **Vrcholy** – graf s N vrcholmi, ktoré navzájom nie sú spojené hranami
- **Lineárny graf –** graf s N vrcholmi navzájom spojenými pomocou N hrán
- **N-rozmerný graf –** graf s N vrcholmi, každý vrchol je spojený so všetkými ostanými vrcholmi hranou.

Pretože prototyp má rozhodnúť o použití určitej knižnice vytvorili sme prototyp na zahodenie.

Účel

Prototyp vykresľovania rôznych typov grafov sme vytvorili preto, aby sme si overili efektívnosť použitia vizualizačných knižníc na zobrazenie sociálnej siete. Pri testovaní sme zohľadňovali tieto vlastnosti:

- počet zobrazených vrcholov
- počet zobrazených hrán

V prototype sme sledovali plynulosť manipulácie s grafom pri zmene jednotlivých vlastností. Keďže plynulosť práce s grafom je vágny pojem, testovanie sme vykonávali empiricky, praktickou manipuláciou s časťami grafu.

Prototyp sme vytvorili aj za účelom testovania konverzie grafov medzi vnútornými reprezentáciami použitých vo vizualizačných knižniciach.

Konverzie podporované prototypom:

JUNG -> Prefuse

Prefuse -> JUNG

Záver

1. Pri testovaní prototypu sme zistili, že knižnica Prefuse podporuje hardvérovo urýchľované vykresľovanie a knižnica JUNG podporuje iba softvérové vykresľovanie. Tento fakt ovplyvnil aj výsledky v prospech knižnice Prefuse.
2. Schopnosť zobrazovať samostatné vrcholy začínala klesať, pri použití oboch knižníc, pri (približne) rovnakom počte vrcholov. Pridávaním hrán do grafov nároky na ich vykreslenie rýchlo stúpali.
3. Z výsledkov testovania možno pozorovať, že knižnica Prefuse je schopná zobrazovať a plynulo manipulovať s väčším počtom vrcholov ako knižnica JUNG. Na základe výsledkov sme sa rozhodli na vizualizáciu použiť knižnicu Prefuse a na vnútornú reprezentáciu knižnicu JUNG.

Počet vrcholov a hrán			Popis stavu
N vrcholov 0 hrán	N vrcholov N hrán	N vrcholov N*N hrán	
160 000	100 000	700	Nie je možné určiť (nezobrazí sa)
100 000	50 000	500	Neovládateľné
40 000	10 000	250	Čiastočne plynulé ovládanie
10 000	5 000	180	Plynulé ovládanie

Tab. 1 Štatistiky použitia knižnice Prefuse

Počet vrcholov a hrán			Popis stavu
N vrcholov 0 hrán	N vrcholov N hrán	N vrcholov N*N hrán	
150 000	60 000	500	Nie je možné určiť (nezobrazí sa)
100 000	40 000	300	Neovládateľné
5 000	600	50	Čiastočne plynulé ovládanie
1 000	300	30	Plynulé ovládanie

Tab. 2 Štatistiky použitia knižnice JUNG

Keďže na vnútornú reprezentáciu sociálnej siete budeme používať formát používaný v knižnici JUNG bolo potrebné otestovať konverziu medzi internými formátmi

používaných v knižniciach JUNG a Prefuse.

6.2. Dynamické načítavania modulov

Zo špecifikácie nášho projektu vyplýva, že chceme vytvoriť rozširovateľnú aplikáciu, kde pre každý z hlavných UC systému (načítanie, spracovanie, vykreslenie, ..) možno pomocou prídavného modulu dodať novú funkcionalitu podobne ako napríklad v aplikácii Firefox. Keďže sa jedná o jednu z hlavných vlastností, bolo prirodzené vyskúšať možnosti, ktoré Java v tomto smere poskytuje počas prototypovania.

Účel

Proces načítavania modulov by mal prebiehať bez nutnosti modifikácie základného kódu, iba pridaním nového java archívu (JAR) do adresárovej štruktúry aplikácie, čím by sa rozšírila špecifická časť aplikácie. V rámci prototypu sme danú vlastnosť overili na ukážke pridávania modulov pre načítanie z rôznych zdrojov.

Priebeh

Definovanie pojmov

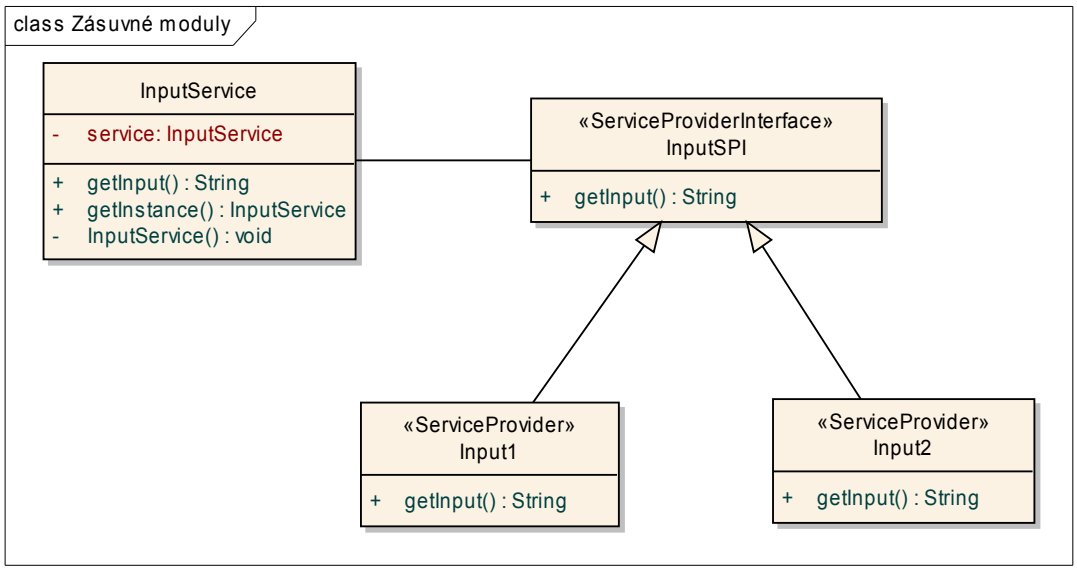
Service (služba) je množina programových rozhraní a tried, ktoré poskytujú prístup k nejakej špecifikovanej funkčnosti aplikácie. Služba definuje rozhrania a spôsob získanie ich implementácie. Napríklad služba pre vstupný modul definuje spôsob načítania, ale už sa nezaoberá priamo jeho implementáciou. To ponechá na *service provider* (poskytovateľa služby).

Service provider interface SPI (interfejs poskytovateľa služby) - je skupina verejných (public) rozhraní definujúcich metódy dostupné pre aplikáciu, ktoré musia poskytovatelia implementovať.

Service provider (poskytovateľ služby). Implementuje *SPI*. Predstavuje tú časť, ktorá bude do aplikácie pridávaná ako zásuvný modul.

Pre lepšie porozumenie pridávame obrázok 26, zobrazujúci vzťahy medzi jednotlivými triedami.

Obrázok 26: Class diagram implementácie zásuvných modulov



Každy *SP* musí zaregistrovať svoju prítomnosť u služby, to možno docieliť umiestnením *SP* do rovnakého adresára ako je uložená implementácia zdrojového kódu služby.

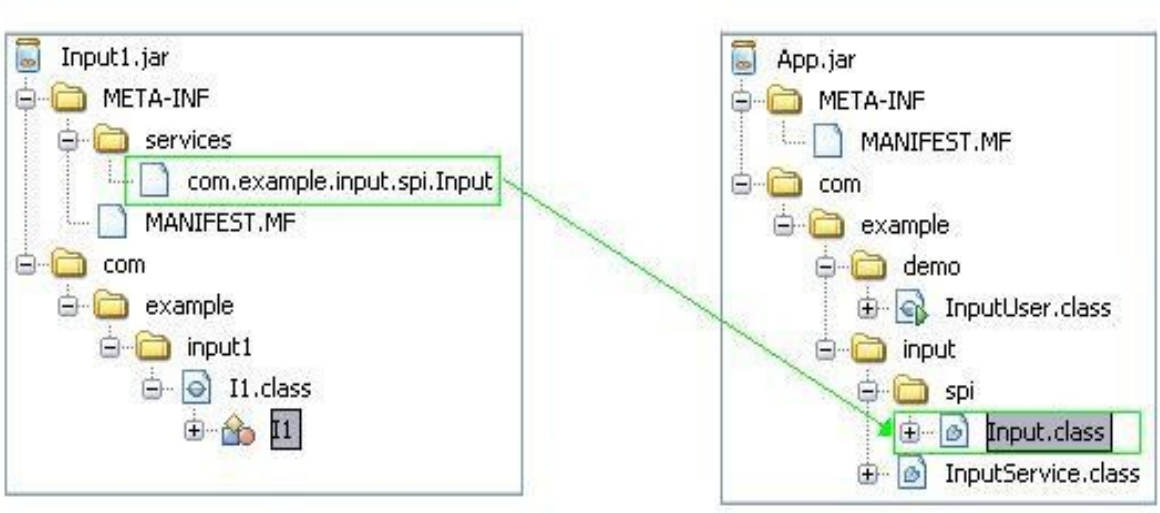
Java SE 6 platforma poskytuje nové API, ktoré umožňuje nájsť, zaviesť a používať *SP*. Trieda *java.util.ServiceLoader* existovala a na pozadí vykonávala svoju prácu od verzie 1.3, ale od Java SE 6 sa stala verejným API.

Táto trieda hľadá *SP* na koreni adresárovej štruktúry aplikácie alebo v zadefinovanom adresári rozšírení pre JRE. *ClassLoader* nájde všetkých *SP*, ktorý implementujú známe rozhranie a umožní prácu s nimi.

ServiceLoader je trieda typu *final* čo znamená, že sa z nej nedá odvodiť nová trieda dedením ani prepísať niektorý z jej algoritmov.

Z pohľadu triedy *ServiceLoader* všetky služby majú jednotný typ, ktorým je obyčajne interfejs alebo abstraktná trieda. *SP* obsahuje jednu alebo viacero tried, ktoré rozširujú službu o implementáciu špecifickú pre ich zámer. *ServiceLoader* vyžaduje, aby *SP* mali základný konštruktor bez parametrov, čo mu umožní rýchle vytváranie inštancií nájdených *SP*.

Na registrovanie poskytovateľa služby (*SP*) musíme vytvoriť konfiguračný meta súbor v adresári *META-INF/services*. Meno súboru by malo byť plne kvalifikované meno služby, ktorú daný *SP* implementuje (obrázok 27).



Obrázok 27: Meta súbor identifikujúci SPI

Záver

Implementáciou prototypu sme vytvorili funkčné demo, ktoré ilustruje registrovanie jednotlivých komponentov pre vstupné moduly na základe ich prítomnosti v adresári aplikácie. Tento spôsob budeme využívať pri realizácii celkovej implementácie. [10]

6.3. Dynamické nastavovanie parametrov a generovanie obrazoviek

Prototypovali sme dynamické načítavanie parametrov na základe anotácií, ich dynamické nastavovanie a generovania obrazoviek na ich základe. Jedná sa o prototyp, ktorý nemá byť na zahodenie.

Účel

Hlavným cieľom bolo zistiť:

1. nakoľko obtiažné by bolo vyhotoviť knižnicu, schopnú dynamicky načítať parametre zo zdrojového kódu, poskytnúť tieto parametre na spracovanie a hodnoty parametrom nastaviť.
2. nakoľko obtiažne by sa s takouto knižnicou pracovalo v klientskom kóde.
3. nakoľko obtiažné by bolo vytvoriť generátor obrazoviek zo zdrojového kódu.

Priebeh

Vytvorili sme triedu, ktorá obsahovala všetky metódy potrebné na načítavanie zoznamu parametrov zo zadefinovaného súboru na základe Java anotácií. Vytvorili sme tiež príklad algoritmu, ktorý obsahoval 2 polia označené anotáciami a metódy, ktoré im nastavovali hodnoty. Rovnako sme vytvorili aj triedu, ktorá na základe zoznamu polí označených anotáciami a ich typov vytvorila panel, s dvojicami komponentov: *JLabel*, ktorý obsahoval meno poľa a *JTextField*, ktorý predstavoval miesto na nastavovanie hodnoty parametra. Vytvorili sme tiež jednoduchú aplikáciu, ktorá s týmito triedami pracovala.

Záver

Pri prototypovaní sme dospeli k nasledovným záverom:

1. knižnica bola vytvorená za približne 2 hodiny práce. Znamená to, že obtiažnosť nie je prehnaná a takýto spôsob nastavovania parametrov zahrnieme do výsledného projektu.
2. práca s touto knižnicou je v klientskom kóde nezvyklá, ale nie obtiažna. Nie je časté, aby sa parametre nastavovali v mapách. Práca je však rýchla a preto sme sa rozhodli, že takéto nastavovanie do výsledného projektu zahrnieme.
3. generátor obrazoviek bol vytvorený za približne jednu hodinu práce. Znamená to, že obtiažnosť nie je prehnaná a takýto spôsob generovania obrazoviek zahrnieme do výsledného projektu.

6.4. Používateľské rozhranie

V rámci prototypovania sme sa rozhodli vytvoriť aj prvú verziu používateľského rozhrania, hoci v tejto fáze má len prezentačnú funkciu, keďže všetky funkcie aplikácie nie sú zrealizované.

Účel

Prototypovaním používateľského rozhrania sme si chceli overiť, či je pre používateľa prijateľné, či je nenáročné na ovládanie a dostatočne intuitívne. V návrhu používateľského rozhrania (kapitola 4.2.) sme popísali sprievodcu, ktorý sa nám zdal dobrým riešením.

Priebeh

Pri tvorbe používateľského rozhrania sme postupovali podľa návrhu používateľského rozhrania, vytvorili sme hlavné okno aplikácie a sprievodcu.

Nakoľko sme sa venovali prototypovaniu viacerých častí naraz, prvé vytvorené používateľské rozhranie sa od výsledného odlišovalo. Najprv sme jednotlivé záložky sprievodcu naprogramovali samostatne a boli napevno začlenené v kóde. Neskôr, keď

sme už mali vytvorený prototyp dynamického nastavovania parametrov a generovania obrazoviek, spojili sme tieto dva prototypy a to tak, že niektoré záložky sprievodcu boli generované dynamicky, načítaním parametrov a následným vygenerovaním obrazovky.

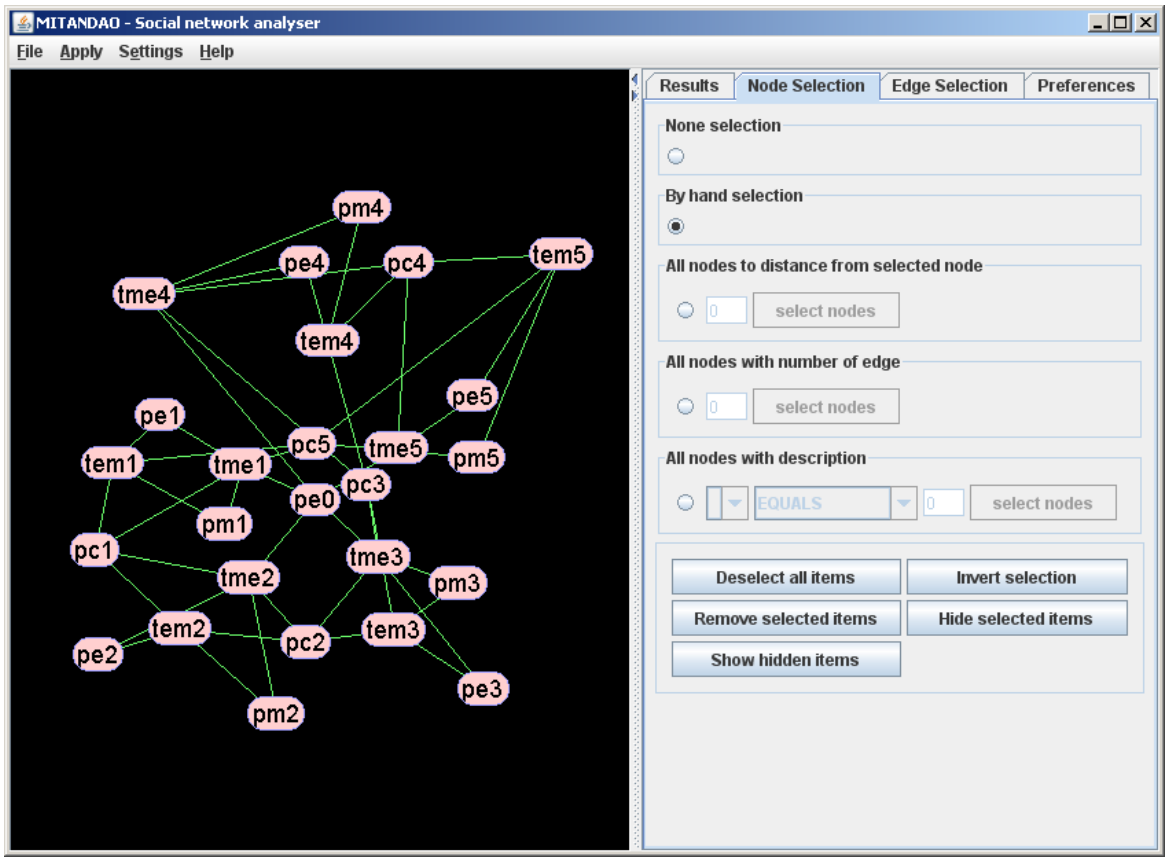
Záver

Prototypovaním sme si potvrdili správnosť myšlienky zavedenia sprievodcu. Sprievodca je jednak prehľadný, každý krok, záložka sa venuje samostatnej časti. Poskytuje tiež používateľom nováčikom pomoc tým, že sprievodca má určenú postupnosť krokov a ich postupným vykonávaním sa používateľ dostane od vstupných dát cez aplikovanie, filtrov alebo algoritmov až k výstupným, spracovaným dátam.

7. Produkt

Výsledný produkt, analyzátor sociálnych sietí, sme sa rozhodli nazvať Mitandao. Toto slovo pochádza zo svahilčiny a v preklade znamená siete, ktoré sú predmetom spracovania a skúmania v našom projekte.

Výsledkom projektu sú dve časti – knižnica Mitandao použiteľná v iných projektoch na analýzu sociálnych sietí a grafov a aplikácia, s ktorou môže používateľ pracovať cez grafické používateľské rozhranie (obrázok 28) a ktorá túto knižnicu využíva. Používateľské príručky pre rôznych používateľov analyzátora sme zaradili vo forme príloh (Prílohy C, D a E).



Obrázok 28: Hlavné okno aplikácie

Už pri návrhu systému sme si určili podmienku, že systém musí byť modulárny. Od tejto základnej požiadavky sa odvíjali všetky ostatné ako aj návrh architektúry a samotná implementácia systému. Bolo pre nás dôležité, aby bol systém modulárny, pretože problematika analýzy sociálnych sietí je veľmi rozsiahla a nie je možné pokryť ju celú v čase, ktorý bol vyčlenený pre tento projekt. Tým, že sme si dali požiadavku modulárnosti, ktorú sa nám podarilo splniť, sme umožnili budúcim používateľom nášho systému doplniť ho o funkcionalitu, ktorú potrebujú.

Niektoré požiadavky na funkcionalitu, ktoré sme si stanovili pri špecifikácii sa

nám žiaľ nepodarilo splniť, v nasledujúcej podkapitole spomenieme, o aké požiadavky išlo a z akých dôvodov sme ich neuskutočnili. V podkapitole 7.2 hovoríme o zaujímavých implementačných detailoch systému a v poslednej podkapitole tejto časti opíšeme možné rozšírenia systému.

7.1. Zmeny oproti špecifikácii a návrhu

Počas prác na projekte sa neobjavili závažnejšie problémy. Napriek tomu sa nám nepodarilo niektoré časti systému vytvoriť, takže niektoré požiadavky na systém, ktoré sme si určili v špecifikácii nie sú implementované. Do určitej miery je to spôsobené tým, že sme pri vytváraní analyzátora použili existujúce grafické knižnice na vizualizáciu a prácu s grafom, knižnice JUNG a Prefuse.

Problémom, ktorý sa nám nepodarilo vyriešiť, je zobrazenie 3D grafu. Pri prvotnej analýze bez prototypovania sme sa chybne domnievali, že knižnica Prefuse dokáže zobraziť aj graf v trojrozmernom priestore. Pri implementácii a študovaní dokumentácie sme zistili, že takéto zobrazenie knižnica nepodporuje. Na trojrozmerné zobrazenie by sme potrebovali využiť ďalšiu externú knižnicu a po zvážení dopadov na projekt sme sa rozhodli túto funkcionalitu neimplementovať.

Ďalší problém sa vyskytol pri našej snahe vytvoriť modul na ukladanie kontrolných bodov, pri snahe o serializáciu grafu sme zistili, že ani knižnica JUNG ani knižnica Prefuse serializáciu nepodporujú. Bola možnosť upraviť zdrojové kódy knižnice JUNG tak, aby bola serializácia možná, rozhodli sme sa však neísť touto cestou, pretože by náš graf a aplikácia nebola kompatibilná s ďalšími verziami knižnice JUNG.

Nakoniec sa nám tento problém podarilo vyriešiť a vytvorili sme modul na ukladanie kontrolných bodov iným spôsobom. Aplikácia teraz ponúka používateľovi možnosť neobmedzeného kroku späť. Tento krok späť je možné uložiť ručne výberom z menu, alebo pomocou špeciálneho modulu `CheckpointWriter`. Jednotlivé kontrolné body, sú ukladané do samostatných XML súborov, ktoré existujú len počas behu programu. Po vypnutí programu sú automaticky vymazané.

Významnou zmenou oproti špecifikácii bola zmena spôsobu načítavania modulov. Hoci sa nám podarilo vytvoriť funkčný prototyp na načítavanie modulov uložených v JAR súboroch, nastal problém s tým, že použitá technológia vracala zakaždým rovnakú inštanciu daného modulu. To by však znamenalo, že by nebolo možné použiť v jednej analýze dvakrát ten istý modul (napríklad načítanie údajov z 2 súborov vo formáte Pajek). Preto sme sa rozhodli vytvoriť vlastnú technológiu na načítavanie modulov.

Rozhodli sme sa vytvoriť len zopár modulov, s ktorými môže používateľ pracovať, nezamerali sme sa na implementovanie všetkých známych algoritmov, ktoré sme opisovali pri analýze v prvej kapitole, pretože systém sme vytvorili tak, aby si tam používatelia mohli kedykoľvek ďalšie moduly doplniť. Z rovnakého dôvodu sme nepoužili ani všetky algoritmy implementované v knižnici JUNG. Prioritne sme vytvárali to, čo sme si stanovili v špecifikácii, tak aby sme pokryli všetky funkcionality, ktoré sme systému predpovedali a prípadné ďalšie nápady a vylepšenia, na ktoré sme prišli počas implementácie, sme sa rozhodli v danom momente neimplementovať.

Rovnako sme sa rozhodli neimplementovať požiadavky, ktoré pre bežnú prácu s aplikáciou nie sú kľúčové. Konkrétne sa jednalo o nastavovanie vlastností logovania a o pridávanie nových modulov cez aplikáciu.

Ďalšou požiadavkou, ktorú sme sa rozhodli neimplementovať, je definovanie atribútov, na základe ktorých sa majú prepojiť dva rôzne grafy. Dôvodom bola prílišná náročnosť takéhoto algoritmu, my sme sa rozhodli zamerať najmä na vytváranie takého prostredia, do ktorého si používateľ môže takýto modul jednoducho doprogramovať.

Ostatné funkcionálne požiadavky sa nám podarilo splniť podľa plánu.

Prehľad neimplementovaných funkcionálnych požiadaviek

R07: Aplikácia umožňuje nastaviť rôzne úrovne logovania

R08: Aplikácia umožňuje nastaviť logovanie pre každý modul zvlášť

R09: Aplikácia podporuje algoritmy implementované v knižnici JUNG 1.7.6

R12: Aplikácia podporuje možnosť vizualizovať údaje vo forme 3D grafu

R37: Aplikácia umožňuje pridávanie nových modulov pomocou používateľského rozhrania

R40: Možnosť definovať atribút na základe, ktorého sa vytvorí prepojenie grafov z rôznych vstupov. Toto prepojenie sa vykoná na základe zhody definovaných atribútov vrcholov v dvoch rôznych grafoch.

7.2. Architektúra

7.2.1. Architektúra knižnice Mitandao

Knižnica Mitandao poskytuje tri základné služby – načítanie modulov, nastavovanie parametrov a zabezpečenie priebehu analýzy. Okrem toho poskytuje nástroje na spracovanie grafov a uchovávanie dát z rôznych analýz.

Dynamické načítavanie modulov

Jednou zo služieb jadra projektu Mitandao je dynamické načítavanie modulov. Moduly sú vyhľadávané podľa záznamov v konfiguračnom súbore modules_paths.xml, ktorý musí byť umiestnený niekde na CLASSPATH. Konfiguračný súbor modules_paths.xml má nasledujúcu štruktúru:

```
<modulePaths>
        ...
   <modulePathToClass>cesta_ku_korenovemu_adresaru_modulov</modulePathTo
Class>
        ...
</modulePaths>
```

V tomto konfiguračnom súbore môže byť nastavený ľubovoľný počet adresárov. Každý z

týchto záznamov môže byť buď absolútna cesta alebo relatívna. Ak sa jedná o absolútnu cestu, táto musí začínať predponou file://. Ak sa jedná o relatívnu cestu, tá nesmie začínať predponou file:// a musí byť relatívna vzhľadom na umiestnenie adresára, odkiaľ sa spúšťa aplikácia používajúca knižnicu Mitandao.
V konfiguračnom súbore nie je nutné nastaviť cesty pre vstupné, výstupné moduly, moduly algoritmov a moduly filtrov zvlášť, pretože knižnica si konkrétne typy dokáže sama zistiť.

Nastavenie parametrov modulu

Knižnica poskytuje automatické nastavovanie parametrov jednotlivým modulom. Toto nastavovanie využíva subsystém Mitandao UI Framework. Ku tomu, aby bolo možné parametre automaticky nastavovať, je nutné členské premenné modulu anotovať anotáciou `Parameter`. Pre technické detaily viď príloha F: Technická dokumentácia, časť Mitandao UI Framework.

Analýza sociálnej siete

Analýza sociálnej siete spočíva vo vytvorení lineárneho reťazca rôznych modulov (angl. workflow) a zavolaní metódy `apply()` na každom z nich. Poradie jednotlivých modulov nie je ohraničené tým, či sa jedná o vstupný modul, modul algoritmu atď, to znamená, že správne nastavenie poradia je na používateľovi knižnice. Knižnici sa dá nastaviť graf, ktorý bude vstupom do takéhoto spracovania, alebo knižnica sama vytvorí novú (prázdnu) inštanciu grafu.

Pri vykonávaní analýzy môže dôjsť k výnimke v ktoromkoľvek module, pričom nie všetky výnimky musia byť fatálne. Knižnica preto všetky výnimky odchytáva a analýzu nechá prebehnúť dokonca. Všetky výnimky, ktoré sa vyskytli, sa ukladajú do jednej výnimky, ktorá sa na konci analýzy vyhodí. Tá obsahuje aj inštanciu analyzovaného grafu, takže ten je prístupný aj v prípade chybového priebehu analýzy.

Pre technické detaily viď príloha F: Technická dokumentácia, časť Priebeh analýzy.

Práca s grafom a uchovávanie údajov

Knižnica JUNG ponúka možnosť uchovávania údajov priamo v grafe vo forme kľúč (`Object`) – hodnota (`Object`). Takéto úložisko je veľmi všeobecné, no na druhej strane automatické spracovanie takýchto dát je problematické. Preto sme sa rozhodli definovať iba dve podporované úložiská dát – úložisko pre dáta vrcholov a úložisko pre dáta hrán. Všetky nástroje na prácu s grafom implementované v knižnici pracujú iba s týmito úložiskami, ostatné ignorujú. Programátori modulov preto musia všetky dáta, ktoré chcú uchovať, prekopírovať do týchto úložísk.

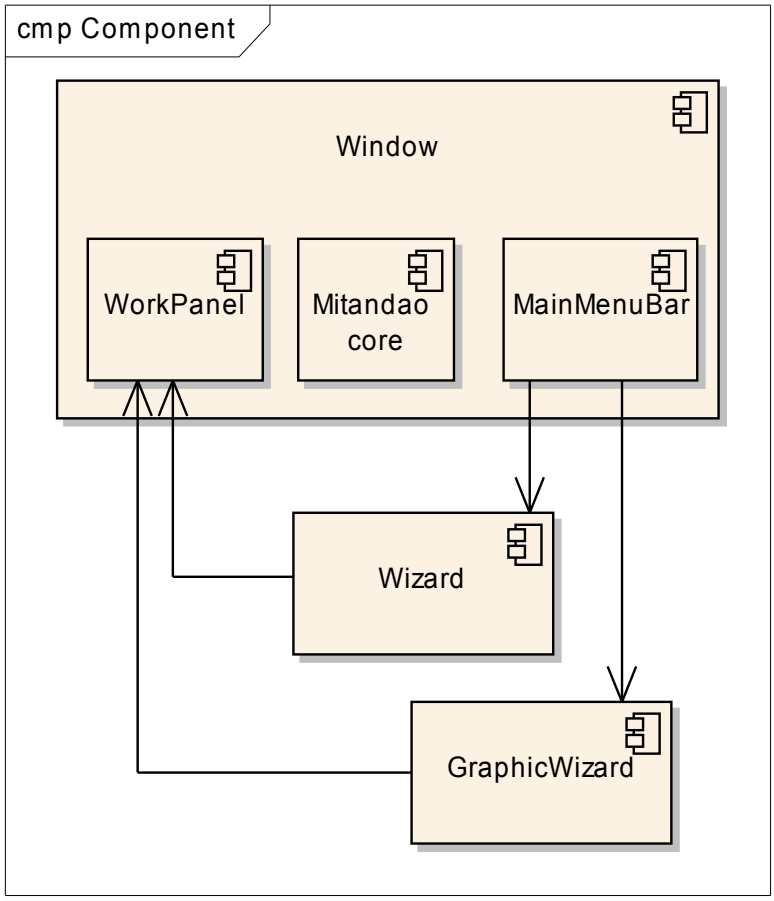
Každá inštancia takéhoto úložiska uchováva dáta jedného atribútu (napríklad jednej analýzy) a je uložená v grafe pod určitým kľúčom (najčastejšie plné meno modulu, ktorý tie dáta vytvoril). Dáta sú v ňom uložené v tvare kľúč (uzol) – hodnota (reťazec), resp. kľúč (hrana) – hodnota (reťazec).

Knižnica ponúka svojim používateľom aj niekoľko nástrojov na prácu s grafom – vytváranie kópie grafu, zjednotenie grafov i kopírovanie dát z jedného úložiska dát implementujúceho rozhranie knižnice JUNG `VertexStringer` (resp. `EdgeStringer`) do

druhého, nášho úložiska dát.

Pre technické detaily viď príloha F: Technická dokumentácia, časť Uchovávanie dát v grafe.

7.2.2. Architektúra aplikácie Mitandao



Obrázok 29: Komponenty aplikácie Mitandao

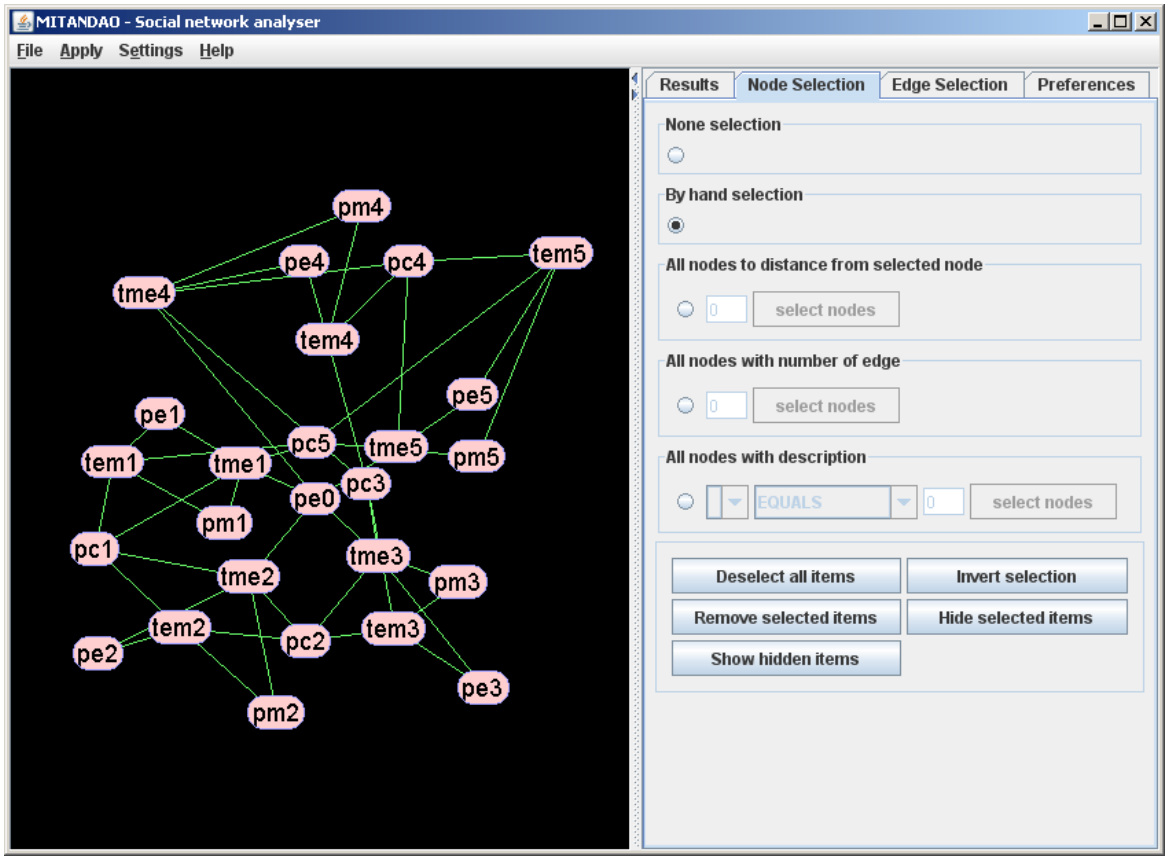
Základnou časťou aplikácie je komponenta, knižnica Mitandao core (obrázok 29). Aplikácia tvorí robustné grafické rozhranie na využívanie funkcionality, ktorú táto knižnica poskytuje. Hlavnou časťou grafického rozhrania je hlavné okno (obrázok 30), ktoré sa otvorí hneď po štarte aplikácie. Toto okno sprostredkúva všetku komunikáciu medzi komponentami aplikácie. V otvorenom hlavnom okne si používateľ môže zvoliť spôsob vytvorenia novej analýzy. Má na výber z dvoch možností.

Prvú možnosť sprostredkováva **sprievodca založený na záložkách**. Tento sprievodca obsahuje sekvenciu štyroch záložiek v postupnosti vstupný modul, filtračný modul, modul analyzátora a výstupný modul.

Druhá možnosť zadávania je pomocou **grafického sprievodcu** (komponenta GraphicWizard). Tento sprievodca umožňuje vizuálne nastaviť poradie jednotlivých modulov a dátové toky medzi nimi. Na nastavenie jednotlivých modulov využíva sprievodcu založeného na záložkách, ktorý je spustený v špeciálnom móde s jednou

záložkou pre daný typ modulu.

V prípade, že už bola vykonaná analýza, alebo načítané vstupné údaje, pošlú sa výstupné údaje zo zvoleného sprievodcu do hlavného okna. V prípade, že používateľ zvolil možnosť zobrazenia grafu aktivuje sa komponent WorkPanel, ktorý pomocou knižnice Prefuse zobrazí analyzovaný graf.



Obrázok 30: Hlavné okno aplikácie Mitandao so zobrazeným grafom.

Po zobrazení výstupného grafu aplikáciou, používateľ má možnosť si priamo v tejto aplikácii prezrieť výsledky analýzy. Pri zobrazení grafu aplikácia taktiež podporuje vizuálnu prácu s týmto grafom. Používateľ môže upraviť vizuálny vzhľad grafu (napr. farba vrchola, farba hrán atď.) , ale taktiež môže upravovať aj jeho štruktúru. Bližší popis architektúry je popísaný v časti Príloha F: Technická dokumentácia – Architektúra aplikácie Mitandao a bližší popis práce s touto aplikáciou je v časti Príloha C: Používateľská príručka.

7.3. Implementácia systému

Implementovať systém sme sa rozhodli v jazyka Java, najnovšia verzia 1.6, pracovali sme v prostredí Eclipse s použitím SVN na zdieľanie zdrojových kódov. Aby bolo možné pridávať a získavať zdrojové kódy z SVN, použili sme zásuvný modul Subclipse. Z ďalších technológií stojí za zmienku použitie JUnit testov, logovania pomocou log4j a použitie nástroja Ant.

Ako sme spomínali už v podkapitole 4.1 návrh architektúry, rozdelili sme systém na dve časti, jadro a grafické rozhranie. Každá z týchto častí tvorí samostatný balík.

V balíku jadra sú zahrnuté nástroje na definovanie lineárneho toku spracovania, nachádza sa tu tiež rozhranie na prácu s modulmi, ako aj samotné moduly.

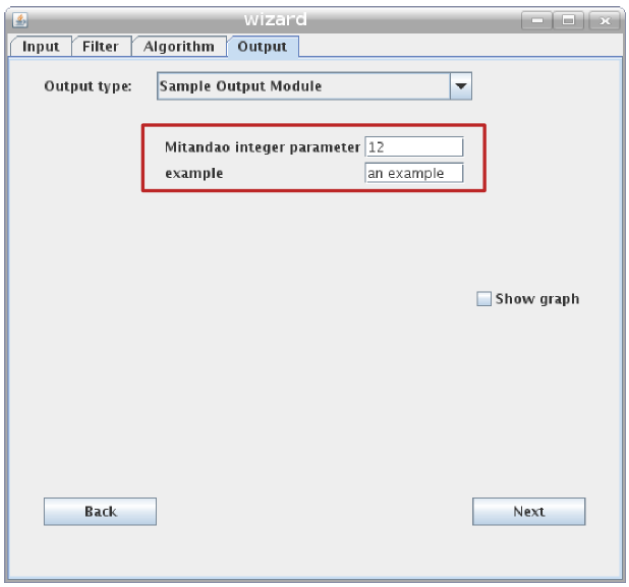
V balíku grafického rozhrania sú zahrnuté zdrojové kódy hlavného okna, sprievodcu a grafického sprievodcu, tiež triedy na prácu so zobrazeným grafom, či triedy na konvertovanie grafu z reprezentácie knižnice JUNG na Prefuse a naopak.

V nasledujúcich podkapitolách sa bližšie pozrieme na hlavné, významné časti produktu, ktoré sme vytvorili, či už v balíku jadra alebo grafického rozhrania.

7.3.1. Mitandao UIFramework

Mitandao UI Framework je časť projektu Mitandao, ktorá slúži na uľahčenie vytvárania grafických používateľských rozhraní pre zásuvné moduly do projektu Mitandao. Existujú tri spôsoby vytvárania grafických používateľských rozhraní pomocou popisovaného rámca:

- Nechať vygenerovať grafické používateľské rozhranie pomocou Mitandao UI Framework (príklad takto vygenerovaného používateľského rozhrania viď. obrázok 31, kde červeným obdĺžnikom je označená časť vygenerovaná pomocou popisovaného rámca)
- Vytvoriť vlastný panel pre potreby daného modulu a prepojiť ho s konkrétnym zásuvným modulom
- Vytvoriť vlastný panel, zaregistrovať ho ako znovupoužiteľnú komponentu a využívať vo viacerých zásuvných moduloch



Obrázok 31:Vygenerované používateľské rozhranie

Vo všetkých prípadoch sa o synchronizáciu hodnôt medzi grafickým panelom a zásuvným modulom stará Mitandao UI Framework. Technické detaily synchronizovania možno nájsť v prílohe F: Technická dokumentácia.

Mitandao UI Framework generuje grafické používateľské rozhranie podľa definovaných anotácií (technológia Java Annotations). Definované anotácie sú:

- Parameter
- Panel
- RemoteParameter
- SpecialParameter

Presný popis účelu a použitia jednotlivých anotácií sa nachádza v prílohe F Technická dokumentácia, v časti Mitando UI Framework.

7.3.2. Vizuálne definovanie procesu analýzy

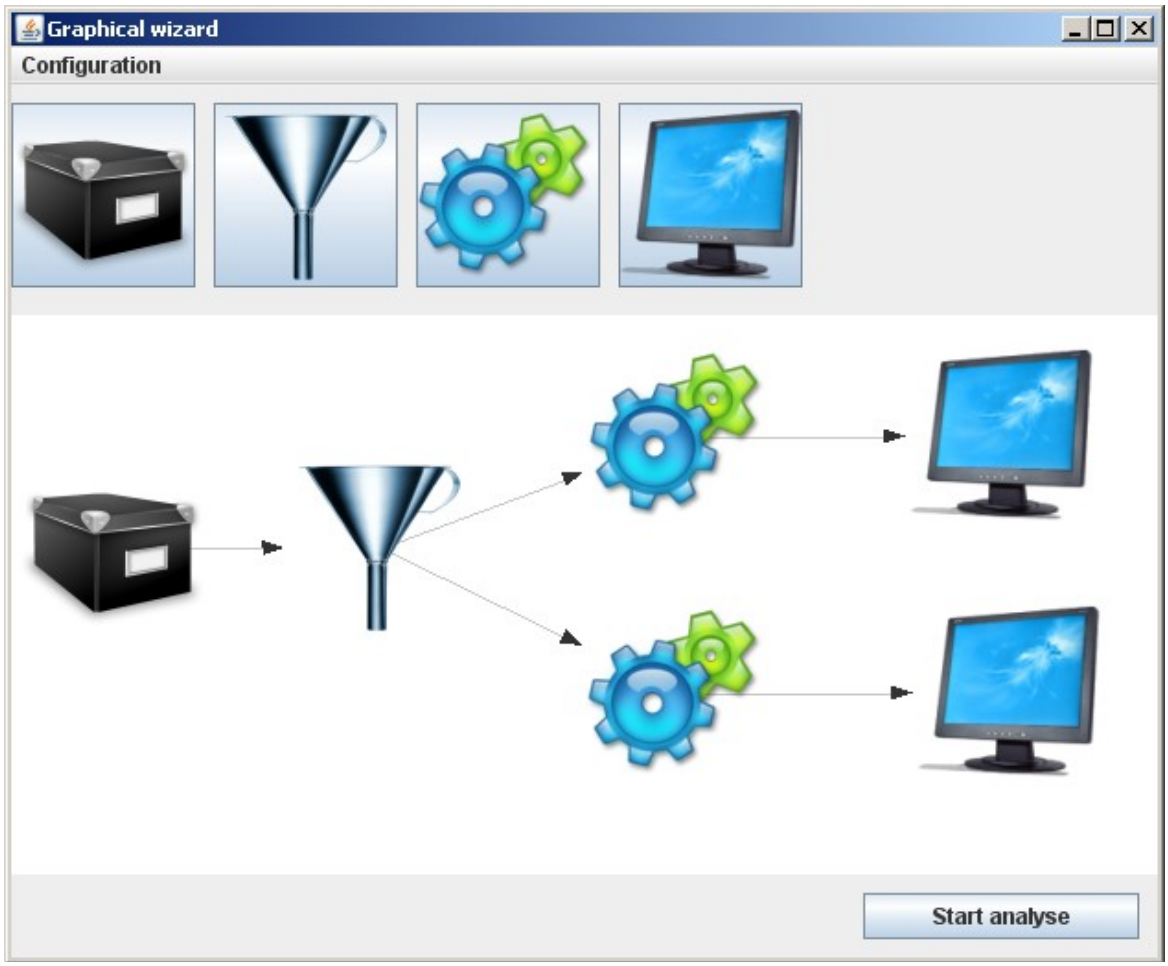
Na základe dodatku k špecifikácii sme sa rozhodli navrhnúť a implementovať rozhranie, v ktorom je možné vizuálne definovať postupnosť procesu analýzy. Vizuálne definovanie podporuje možnosť vytvorenia paralelných tokov údajov, ich ďalšie rozvetvenie do viacerých tokov. Tento prístup výrazne zmenil princípy toku spracovania, ktoré boli navrhnuté pre knižnicu Mitandao. Táto knižnica podporuje definovanie len lineárneho toku spracovania a jeho sekvenčné vykonávanie.

Na to, aby bolo možné vykonávať aj paralelné vetvy toku spracovania sme upravili grafickú aplikáciu takým spôsobom, aby vedela rozdeliť definovanú štruktúru spracovania na lineárne časti (vetvy). Tieto vetvy sa následne vykonajú sekvenčne v poradí v akom sa tok rozvetvuje alebo spája. Dôsledkom toho je fakt, že aj napriek vizuálnemu paralelnému definovaniu sa analýza vykonáva sekvenčne.

Grafický sprievodca analýzou

Vizuálne definovanie postupnosti spracovania je možné vykonať pomocou grafického sprievodcu. Spôsob práce s grafickými prvkami je analogický ako pri vizuálnej práci s grafom sociálnej siete v hlavnom okne aplikácie. Jednotlivé uzly predstavujú moduly a hrany medzi nimi predstavujú tok údajov. Princípy práce a zloženie tohto sprievodcu je bližšie popísané v prílohe C: Používateľská príručka.

Na implementáciu grafického rozhrania tohto sprievodcu sme použili komponenty z knižnice Prefuse, ktorá už bola v projekte použitá.



Obrázok 32: Okno grafického sprievodcu s rozvetveným tokom spracovania.

Grafický sprievodca (obrázok 32) je určený najmä pre používateľov s vyššími nárokmi a umožňuje vytvárať aj zložité štruktúry spracovania. Na to, aby používateľ nemusel toto spracovanie a nastavenia jednotlivých modulov definovať pri každej novej analýze, má možnosť jednotlivé nastavenia uložiť do externého XML súboru.

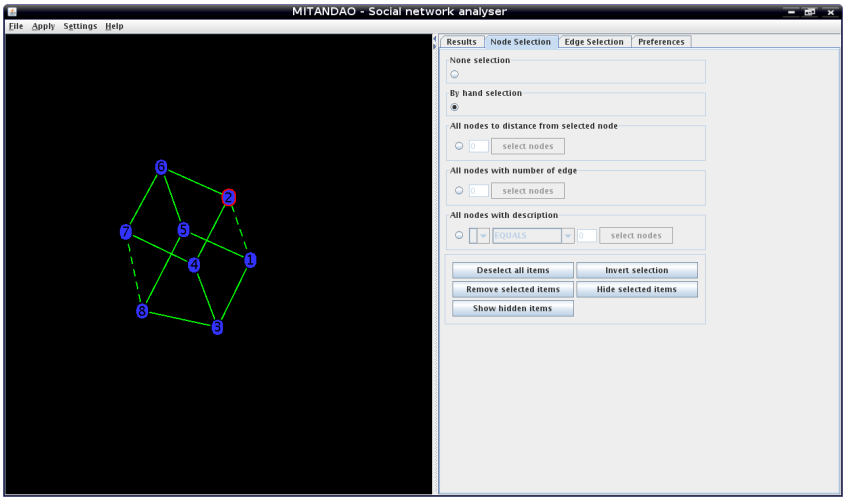
Ukladanie a načítanie nastavení pre každý modul – ukladajú sa nastavenia pre každý modul zvlášť. Ukladajú sa iba nastavenia, čiže pred načítaním modulu treba najskôr nastaviť typ modulu a následne načítať nastavenia pre daný modul.

Ukladanie a načítanie celého štruktúry spracovania (konfigurácie) – ukladá sa kompletná štruktúra definovaného spracovania aj s nastaveniami jednotlivých modulov. V prípade, že určitý modul pri načítaní nie je nainštalovaný v knižnici načíta sa prázdny modul, ktorý je potrebné neskôr nastaviť.

7.3.3. Grafické vyberanie vrcholov a hrán

V prípade, že je v hlavnom okne aplikácie zobrazený graf (ako je vidieť na obrázku 33), je možné s daným grafom ďalej interaktívne pracovať. Podsystém vyberania vrcholov a hrán slúži na to, aby bolo možné jednoducho vyberať hrany resp. vrcholy buď

klikaním na hrany resp. vrcholy, alebo hromadne, podľa toho, či daná skupina vrcholov alebo hrán vyhovuje špecifikovanej podmienke. O tom, ktoré vrcholy resp. hrany sú aktuálne vybraté sú notifikované iné podsystémy aplikovaním štandardného návrhového vzoru Observer (technické detaily toho, ako sa na tento subsystém registrovať a ako sú klienti notifikovaní, viď prílohu F: Technická dokumentácia, časť Grafické vyberanie vrcholov a hrán). Takýmito podsystémami sú napríklad podsystémy na zmenu farby zvolených vrcholov alebo vypisovanie výsledkov analýzy pre vybraté vrcholy resp. hrany.



Obrázok 33: Zobrazenie grafu v hlavnom okne

Možnosti, ako vyberať vrcholy sú nasledujúce:

- ručne (pomocou klikania na vrcholy na zobrazenom grafe)
- vybrať všetky vrcholy do istej vzdialenosti od zvoleného vrcholu
- vybrať všetky vrcholy, ktoré majú zvolený počet hrán
- vybrať všetky vrcholy, ktoré vyhovujú istému popisu (napr. výsledok nejakej analýzy je menší ako N)
- invertovať zoznam vybratých vrcholov

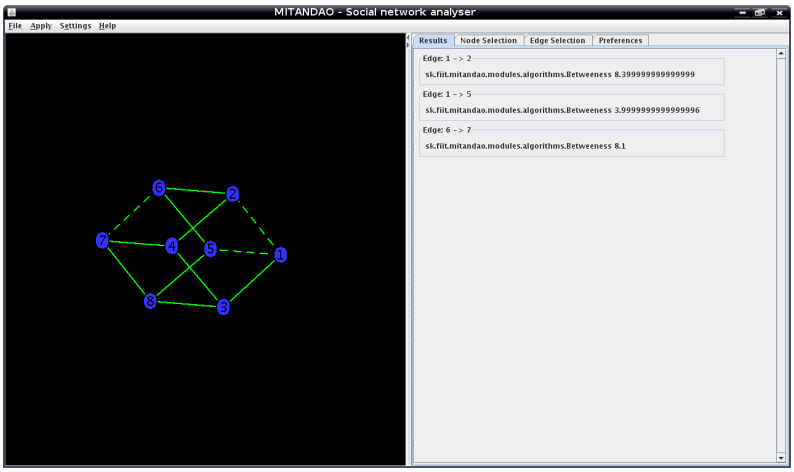
Možnosti, ako vyberať hrany sú nasledovné:

- ručne (pomocou klikania na hrany na zobrazenom grafe)
- vybrať všetky hrany, ktoré vyhovujú istému popisu (napr. výsledok nejakej analýzy je menší ako N)
- invertovať zoznam vybratých hrán

Technické detaily architektúry tohto subsystému F: Technická dokumentácia, časť Grafické vyberanie vrcholov a hrán.

7.3.4. Zobrazovanie výsledkov analýzy

Výsledky všetkých analýz sa ukladajú do grafu ku vrcholom alebo hranám (v závislosti od použitého algoritmu). Výsledky je možné prehliadať v záložke Results v pravej časti obrazovky. Toto zobrazovanie je notifikované o aktuálnom zozname vybratých vrcholov a hrán (viď kapitola Grafické vyberanie vrcholov a hrán). Hodnoty atribútov zvolených vrcholov a hrán sa zobrazia pod sebou (obrázok 34).



Obrázok 34: Zobrazenie výsledkov analýzy v záložke Results

7.3.5. Nastavenie zobrazenia vrcholov a hrán

Grafické používateľské rozhranie obsahuje možnosť nastaviť niektoré atribúty zobrazovania vrcholov a hrán. To sa dá spraviť dvoma spôsobmi – globálne nastavenia všetkých vrcholov alebo hrán a nastavenie atribútov vybraných vrcholov a hrán.

Všetky nastavenia sa ukladajú v používateľských dátach v grafe vo forme reťazcov.

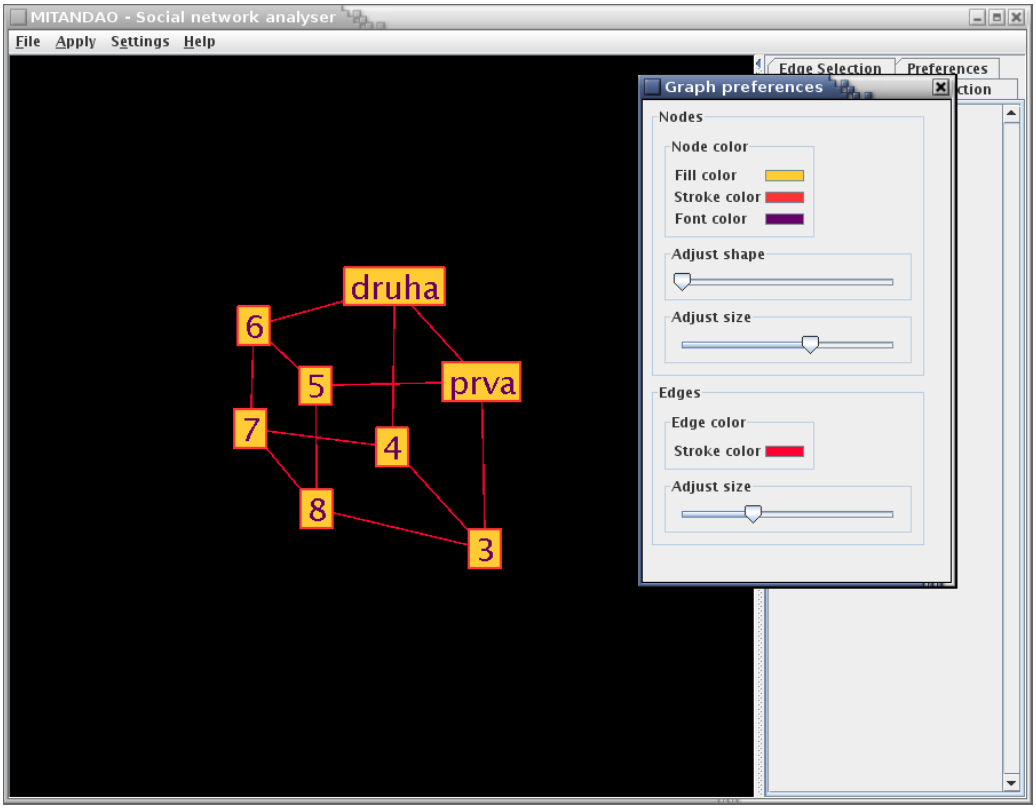
Globálne nastavenia vrcholov a hrán

Tieto nastavenia je možné sprístupniť v položke horného menu Settings -> Graph... Následne sa zobrazí dialóg (obrázok 35), kde sa dajú nastaviť nasledujúce atribúty:

- farba vrcholov
- farba ohraničenia vrcholov
- farba písma
- zakrivenie vrcholov
- veľkosť vrcholov

– hrúbka hrán
– farba hrán

Všetky atribúty nastavené v tomto dialógu sa uchovávajú trvalo pomocou technológie Java Preferences – sú teda prístupné i po ukončení aplikácie.



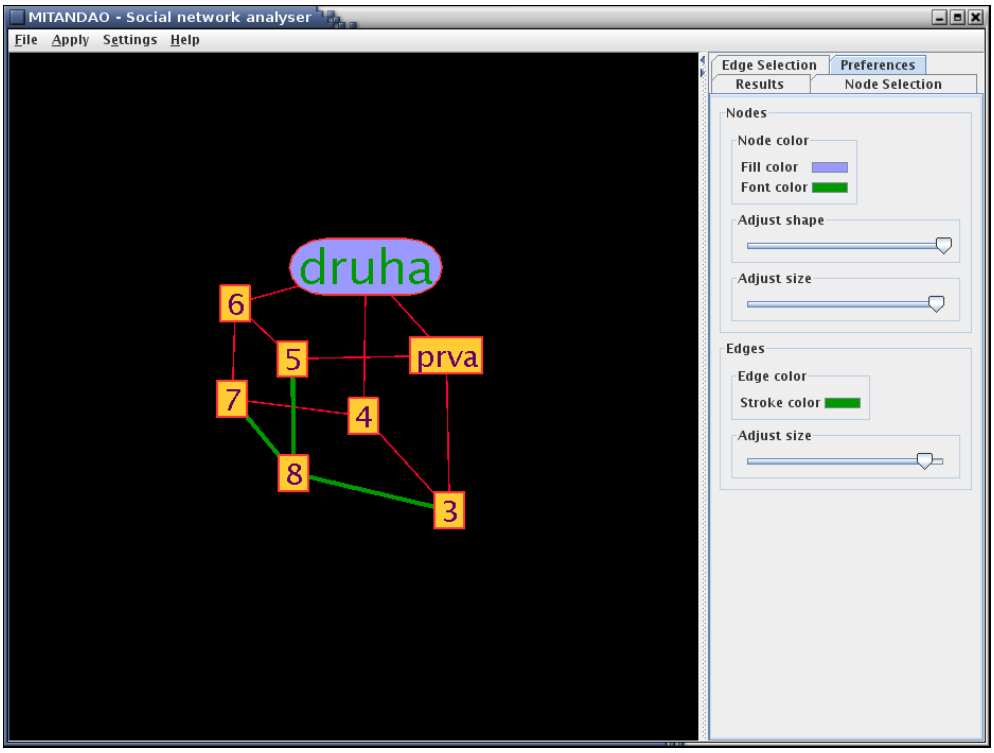
Obrázok 35: Dialóg na nastavenie vrcholov a hrán

Nastavenie vybratých vrcholov a hrán

Po vybratí vrcholov alebo hrán je možné zmeniť atribúty ich zobrazovania v pravom paneli aplikácie v záložke Preferences (obrázok 36). Atribúty, ktoré je možné zmeniť:

– farba vrcholov
– farba písma
– zakrivenie vrcholov
– veľkosť vrcholov
– hrúbka hrán
– farba hrán

Atribúty nastavené v tejto záložke sa pamätajú i vtedy, keď sa graf použije na ďalšiu analýzu, pretože sa uchovávajú a načítavajú z dát uložených v grafe, ktoré sa pri analýze skopírujú do JUNG grafu a späť do Prefuse grafu.



Obrázok 36: Záložka Preferences

7.4. Možnosti rozšírenia systému

Do finálnej prezentácie projektu sa plánujeme venovať dokončovaniu projektu a dopĺňaniu drobných detailov, ktoré nám napadli počas riešenia projektu. Dôležité podľa nás je aplikáciu dostatočne testovať, aby sme odhalili prípadné nedostatky a mohli pracovať na ich odstraňovaní. To, akým smerom sa bude uberať práca na projekte čiastočne závisí aj od posudku konkurenčného tímu na náš projekt.

Radi by sme do prezentácie aplikáciu doplnili o niekoľko ďalších modulov, hlavne modulov na analýzu, prípadne filtrovanie grafu. Tým, že aplikácia je modulárna je pomerne jednoduché ju aj v budúcnosti rozšíriť o ďalšie moduly. Na novovytvorené moduly by sme chceli vytvoriť centrálne úložisko, kde by si ich mohli rôzny používatelia vymieňať.

Tiež sa budeme venovať príprave samotnej prezentácie. Do prezentácie je potrebné taktiež vytvoriť reprezentatívny leták, na ktorom budú vybrané informácie o aplikácii. Spolu s letákom sa bude vytvárať aj celková marketingová stránka projektu.

8. Záver

Predkladaný dokument opisuje našu prácu na projekte v priebehu dvoch semestrov. Prvý semester bol pre nás uvedením a zorientovaním sa v oblasti sociálnych sietí, ako aj v jednotlivých už existujúcich riešeniach. Na základe vykonanej analýzy sme mohli vytvoriť špecifikáciu nami navrhovaného systému.

Po špecifikácii a návrhu systému sme sa rozhodli niektoré jeho časti prototypovať. V prototype sme sa zamerali hlavne na implementáciu používateľského rozhrania a tiež na otestovanie vykresľovania grafu knižnicami Jung a Prefuse. Obe knižnice sme analyzovali v časti 2.1., každá má svoje výhody a práve prototypovaním vykresľovania pomocou nich sme si overili ich funkčnosť, na základe čoho sme sa rozhodli, ktorú budeme v projekte používať. Na základe výsledkov, ktoré sme prototypovaním získali, sme sa rozhodli pre použitie knižnice Prefuse na vykresľovanie grafu a knižnicu JUNG sme sa rozhodli použiť na vnútornú reprezentáciu grafu.

Pri prototypovaní systému sme nenarazili na výraznejšie alebo neriešiteľné problémy, overili sme si pri ňom naše predpoklady, prípadne vyjasnili nejasnosti. Keďže prototypovanie bolo úspešné, pri implementácii projektu sme pokračovali v rozvíjaní vytvoreného prototypu.

Počas druhého semestra sme pracovali na vývoji výsledného produktu, výsledkom je Mitandao – analyzátor sociálnych sietí, ktorý je jednak rozširovateľnou spustiteľnou aplikáciou, no môže byť použitý aj ako knižnica pri vývoji iných aplikácií. Aplikáciu sme vytvorili tak, aby bola modulárna a používateľ si ju mohol prispôsobiť svojim požiadavkám. Mitandao môže mať niekoľko používateľov a používateľské príručky pre jednotlivých používateľov sme zaradili ako prílohy k dokumentácii.

Pri vyvíjaní analyzátora sme narazili na obmedzenia zo strany knižníc JUNG a Prefuse, ktoré sme bližšie opísali v podkapitole 7.1. Produkt, jeho implementácia ako aj možnosti ďalšej práce na projekte sú bližšie opísané v kapitole 7.

Celý systém sme implementovali v jazyku Java pomocou grafickej knižnice Swing.

Použitá literatúra

1. GraphML: http://graphml.graphdrawing.org/index.html
2. Hanneman, R., Riddle, M., Introduction to Social Network Methods Tutorial [online]. 2005. [posledná kontrola: 2007-11-11]. Dostupné na Internete: http://faculty.ucr.edu/~hanneman/nettext/
3. InFlow: http://www.orgnet.com/
4. Java: java.sun.com
5. JUNG: http://jung.sourceforge.net/
6. PAJEK: http://vlado.fmf.uni-lj.si/pub/networks/pajek/
7. Prefuse: http://prefuse.org/
8. weka: http://www.cs.waikato.ac.nz/ml/weka/
9. Witten, I.H., Frank, E., Data Mining. Morgan Kaufmann Pub, 2005. 525 s. ISBN: 0120884070
10. http://java.sun.com/developer/technicalArticles/javase/extensible/

Príloha A: Matica sledovateľnosti požiadaviek

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10
R01	X									
R02	X									
R03							X			
R04							X			
R05							X			
R06	X	X								
R07										X
R08										X
R09			X							
R10							X			
R11				X			X			
R12				X			X			
R13					X	X				
R14					X	X				
R15					X	X				
R16					X	X				
R17					X					
R18					X					
R19					X					
R20					X					
R21					X					
R22					X					
R23					X					
R24					X					
R25					X					
R26					X					
R27					X					
R28					X					
R29						X				
R30						X				
R31						X				
R32						X				
R33						X				
R34									X	
R35									X	
R36									X	
R37								X		

Príloha B: Dodatok k špecifikácii

Pri prototypovaní a pri tvorbe podrobného návrhu sme identifikovali nasledovné funkcionálne požiadavky, ktoré považujeme za opodstatnené a rozhodli sme sa ich začleniť do výsledného systému.

Funkcionálne požiadavky

Vnútorná práca s údajmi

Podpora viacerých grafov na vstupe – používateľ môže definovať viacero grafov na vstupe.

R38: Použitie viacerých grafov z rôznych typov vstupných údajov

R39: Vnútorná reprezentácia realizovaná ako jeden graf zložený z dvoch neprepojených častí

R40: Možnosť definovať atribút na základe, ktorého sa vytvorí prepojenie grafov z rôznych vstupov. Toto prepojenie sa vykoná na základe zhody definovaných atribútov vrcholov v dvoch rôznych grafoch.

Grafické používateľské rozhranie

Vizuálne definovanie postupnosti spracovania sociálnej siete – popri sprievodcovi založenom na záložkách používateľ môže definovať postupnosť spracovania sociálnej siete vizuálne a to definovaním jednotlivých blokov spracovania vo forme grafu.

R41: možnosť definovať viacero blokov spracovania v používateľom definovanom poradí

R42: možnosť definovať paralelné spracovanie viacerých sietí.

Príloha C: Používateľská príručka (angl)

Príloha D: Príručka pre programátora (angl)

Príloha E: Príručka pre používateľa knižnice mitandao (angl)

Príloha F: Technická dokumentácia

Architektúra

Architektúra knižnice Mitandao

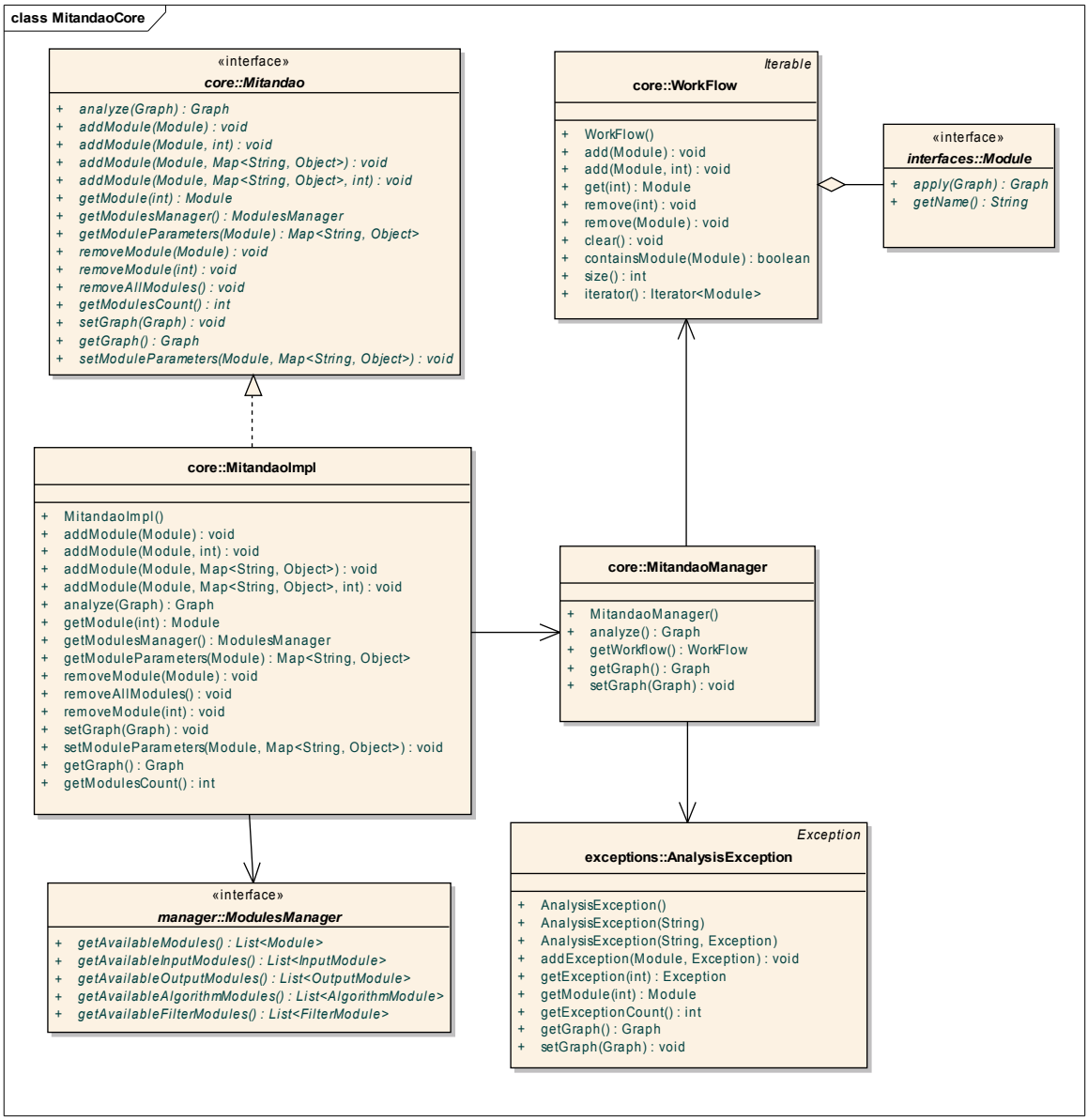
Knižnica Mitandao poskytuje tri základné služby – načítanie modulov, nastavovanie parametrov a zabezpečenie priebehu analýzy.

Priebeh analýzy

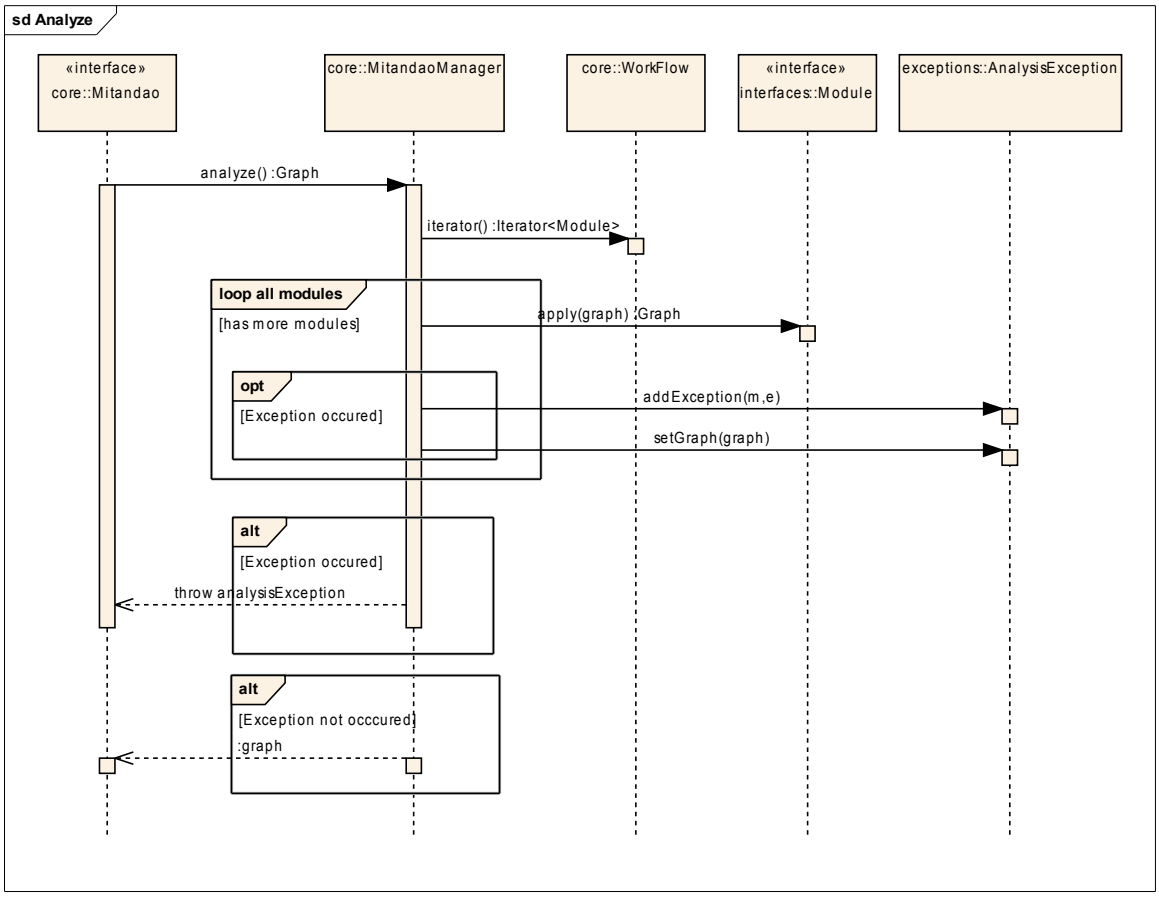
Analýza pozostáva z vykonania metódy `apply()` na všetkých moduloch v toku spracovania. To zabezpečuje trieda `MitandaoManager`, ktorá si v sebe drží jeho inštanciu, rovnako ako aj inštanciu analyzovaného grafu. Ten sa nastavuje metódou `setGraph()`, ktorá sa v triede `MitandaoImpl` volá vždy pred metódou `analyze()`. V prípade, že používateľ zadá ako parameter tejto metóde hodnotu `null`, vytvorí sa nový prázdny graf, nad ktorým bude analýza prebiehať.

Pri analýze môže dôjsť k výnimke v ktoromkoľvek module v toku spracovania, v prípade ich výskytu však knižnica nepreruší svoju činnosť, ale všetky výnimky a moduly, v ktorých nastali, vkladá do vlastnej výnimky `AnalysisException`, ktorú vyhodí až na konci celej analýzy. Táto výnimka si v sebe drží i inštanciu grafu, ten je teda prístupný i v prípade chybného behu analýzy.

Na obrázku 1 je zobrazený diagram tried zodpovedných za analýzu, jej priebeh zobrazuje sekvenčný diagram na obrázku 2.



Obrázok 1: Triedy, ktoré sú potrebné pri analýze

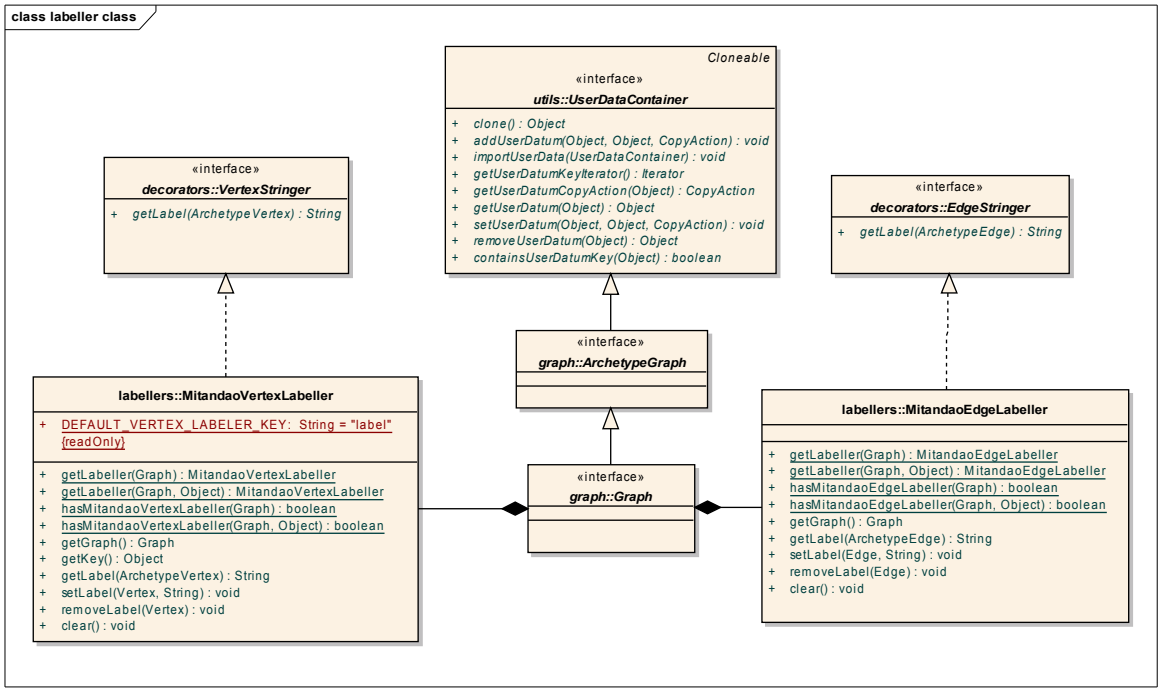


Obrázok 2: Analýza, priebeh

Uchovávanie dát v grafe

Knižnica Mitandao používa na prácu s grafom rozhranie `edu.uci.ics.jung.graph.Graph` a v prípade potreby vytvára jednu jeho konkrétnu implementáciu `edu.uci.ics.jung.graph.impl.SparseGraph`. Rozhranie `Graph` je potomkom rozhrania `UserDataContainer`, je teda schopné uchovávať používateľské dáta v tvare kľúč (Object) – hodnota (Object). Aby sme štandardizovali spôsob uchovávania dát rôznymi modulmi, rozhodli sme sa vytvoriť vlastné štruktúry a pri práci s grafom používať iba tie. To znamená, že každý programátor modulu musí svoje dáta skopírovať do našich dátových štruktúr.

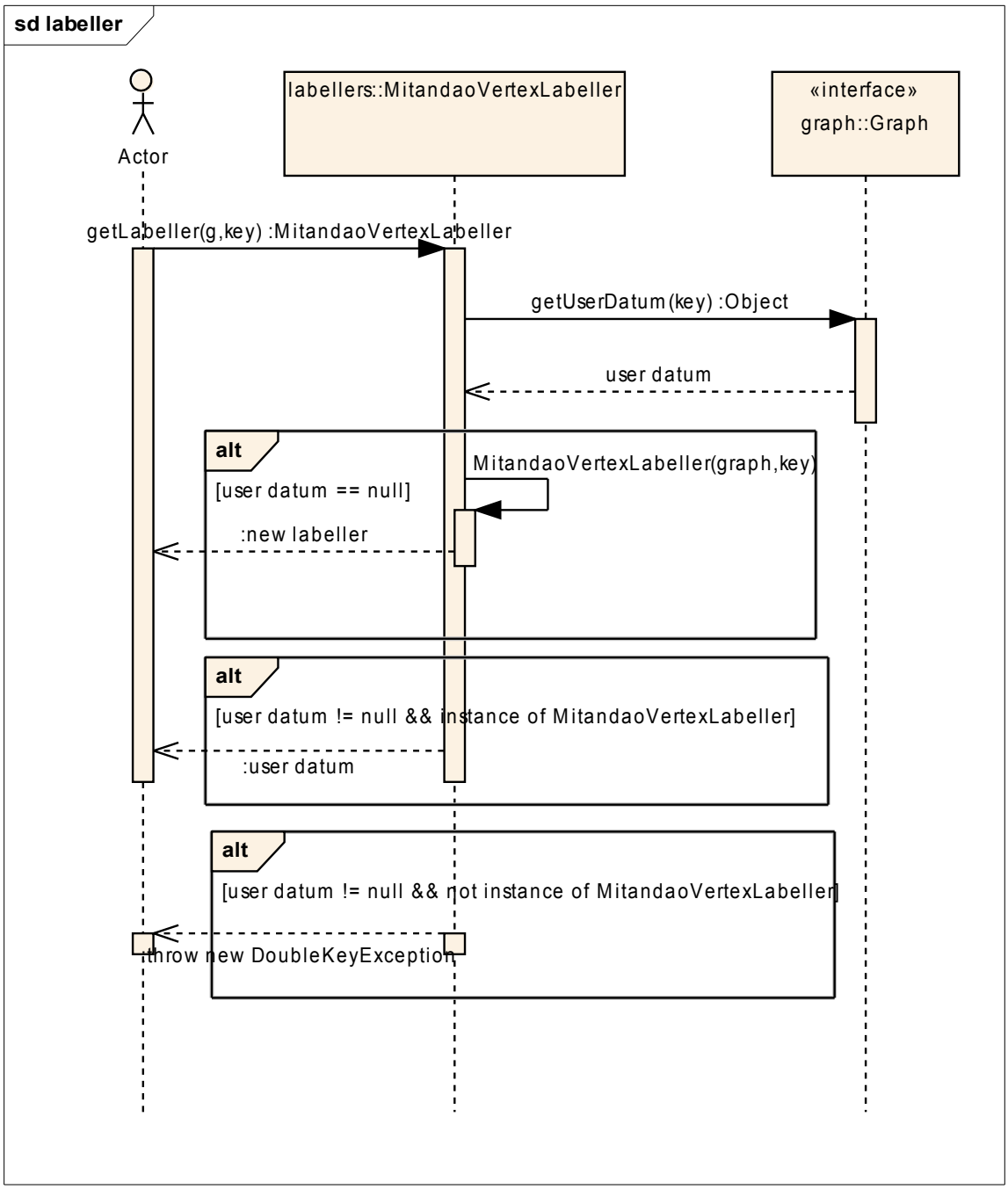
Pri vytváraní dátových štruktúr na uchovávanie údajov sme sa inšpirovali triedou `edu.uci.ics.jung.graph.decorators.StringLabeller`, ktorá však nevyhovovala všetkým našim požiadavkám (napríklad vyžadovala, aby hodnota priradená vrcholu bola unikátna), preto sme ju nemohli použiť priamo. Nami vytvorené triedy sa volajú `MitandaoVertexLabeller` (pre ukladanie hodnôt vrcholov) a `MitandaoEdgeLabeller` (pre ukladanie hodnôt hrán). Každá trieda má v sebe mapu, kde kľúč je vrchol (hrana) a hodnota akýkoľvek reťazec. Hodnoty vrcholov (hrán) sa nastavujú alebo získavajú cez get alebo set metódy. Jedna inštancia triedy je v grafe registrovaná pod jedným kľúčom, ktorým je v prípade výsledkov niektorého modulu plný názov triedy modulu. Slúži teda na uchovávanie jednej vlastnosti vrchola (uzla). Trieda `MitandaoVertexLabeller` obsahuje konštantu `DEFAULT_VERTEX_LABELLER_KEY`, ktorá sa v celej knižnici považuje za kľúč, pod ktorým sú uložené názvy vrcholov. Diagram tried je na obrázku 3.



Obrázok 3: Diagram tried labeller-ov

Inštancia triedy `MitandaoVertexLabeller` ani `MitandaoEdgeLabeller` sa nedá vytvoriť priamo (má konštruktor s viditeľnosťou `protected`), ale dá sa získať jednou z dvoch statickým metód – `getLabeller(Graph graph)` a `getLabeller(Graph graph, Object key)`. Po zavolaní tejto metódy sa buď z grafu vyberie existujúca inštancia pod daným kľúčom (v prípade metódy `getLabeller(Graph graph)` pod kľúčom `DEFAULT_VERTEX_LABELLER_KEY`), alebo ak graf taký objekt neobsahuje, vytvorí sa nová inštancia. Tento prístup má tú výhodu, že používateľ nemusí nikdy zisťovať, či daný labeller existuje alebo nie.

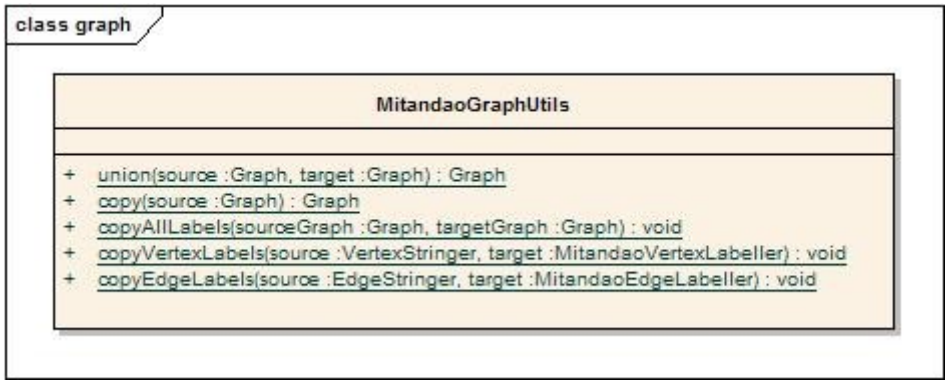
Keďže sú inštancie `MitandaoVertexLabeller` i `MitandaoEdgeLabeller` fyzicky uložené v inštancii grafu, je možné na prístup k nim využiť i metódy grafu, napríklad iterovať cez všetky kľúče pomocou `getUserDatumKeyIterator()`. Sekvenčný diagram nastavenia hodnôt je na obrázku 4.



Obrázok 4: Nastavenie hodnôt

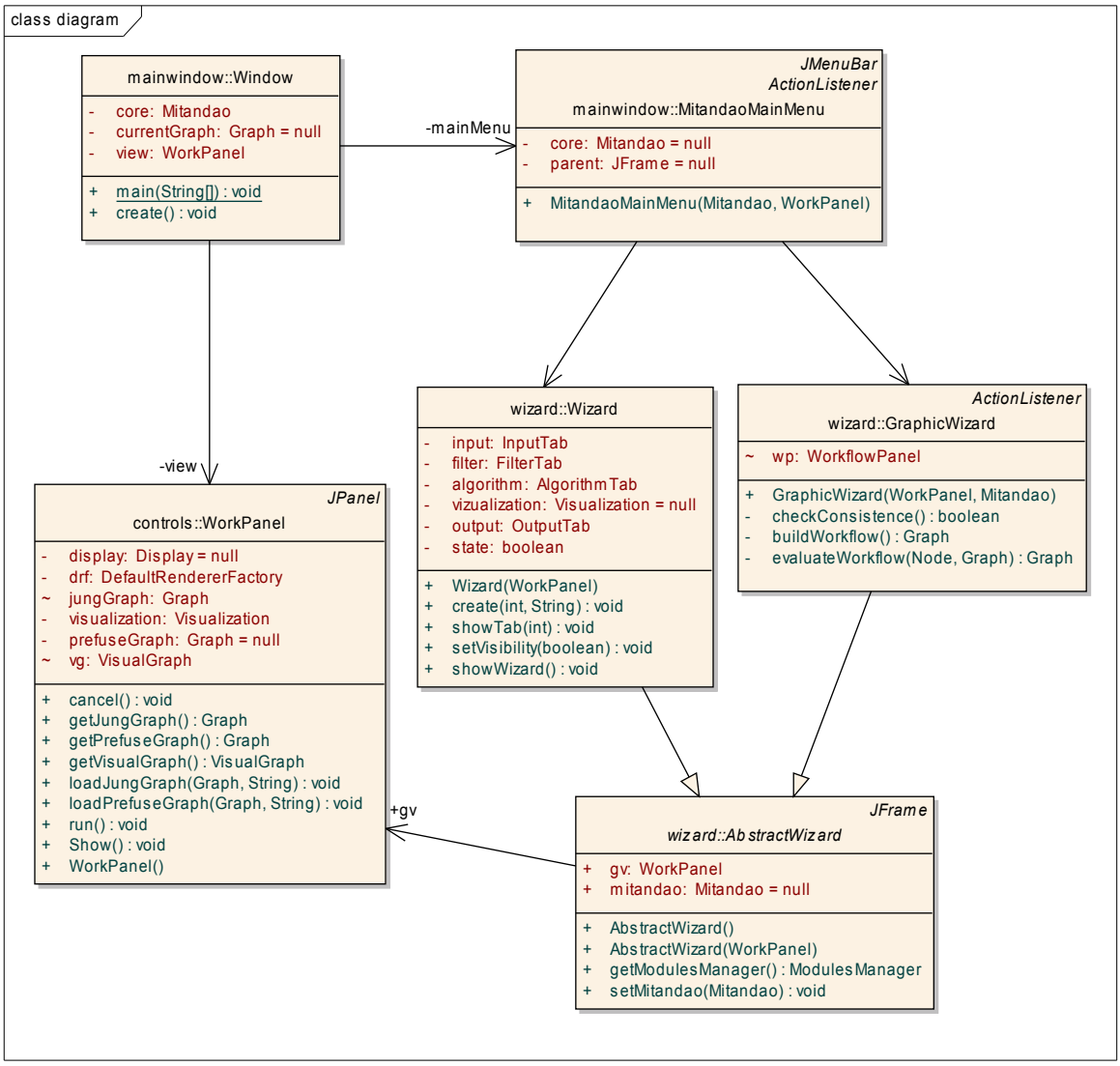
Práca s grafom

Pre prácu s grafom sme vytvorili triedu `MitandaoGraphUtils` (obrázok 5), ktorá poskytuje statické metódy na kopírovanie labellerov, zjednotenie grafov do jedného a na vytvorenie kópie grafu. Pri zjednotení a vytváraní kópie grafu sa ignorujú všetky používateľské dáta v grafoch okrem inštancií `MitandaoVertexLabeller` a `MitandaoEdgeLabeller`.



Obrázok 5: Trieda MitandaoGraphUtils

Architektúra aplikácie Mitandao



Obrázok 6: Diagram tried aplikácie Mitandao

Základnou časťou aplikácie Mitandao je knižnica Mitandao. Aplikáciu tvorí grafické rozhranie, ktoré umožňuje vizuálnu prácu a vizualizáciu sociálnych sietí spracovávaných v knižničnej časti.

Hlavnou časťou grafickej aplikácie je hlavné okno `Window` , ktoré sa otvorí hneď po štarte aplikácie. Trieda *`Window`* obsahuje referenciu na inštanciu objektu Mitandao a túto referenciu poskytuje ďalej objektom, ktoré využívajú jej funkcionalitu. V prípade, že už bola vykonaná analýza alebo načítané vstupné údaje, trieda `Window` okrem referencie na objekt Mitandao obsahuje aj aktuálne spracovávaný graf.

Používateľ má možnosť vybrať si spôsob, akým vytvorí tok spracovania analýzy. Hlavné okno `Window` obsahuje menu `MitandaoMainMenu`, pomocou ktorého je možné spustiť sprievodcov na vytvorenie toku spracovania analýzy. Používateľ má možnosť výberu medzi statickým zadávaním a vizuálnym (dynamickým) zadávaním.

Prvú možnosť sprostredkováva **sprievodca založený na záložkách** sprostredkováva ho trieda `Wizard`. Trieda obsahuje sekvenciu štyroch záložiek v postupnosti vstupný modul, filtračný modul, modul analyzátora a výstupný modul.

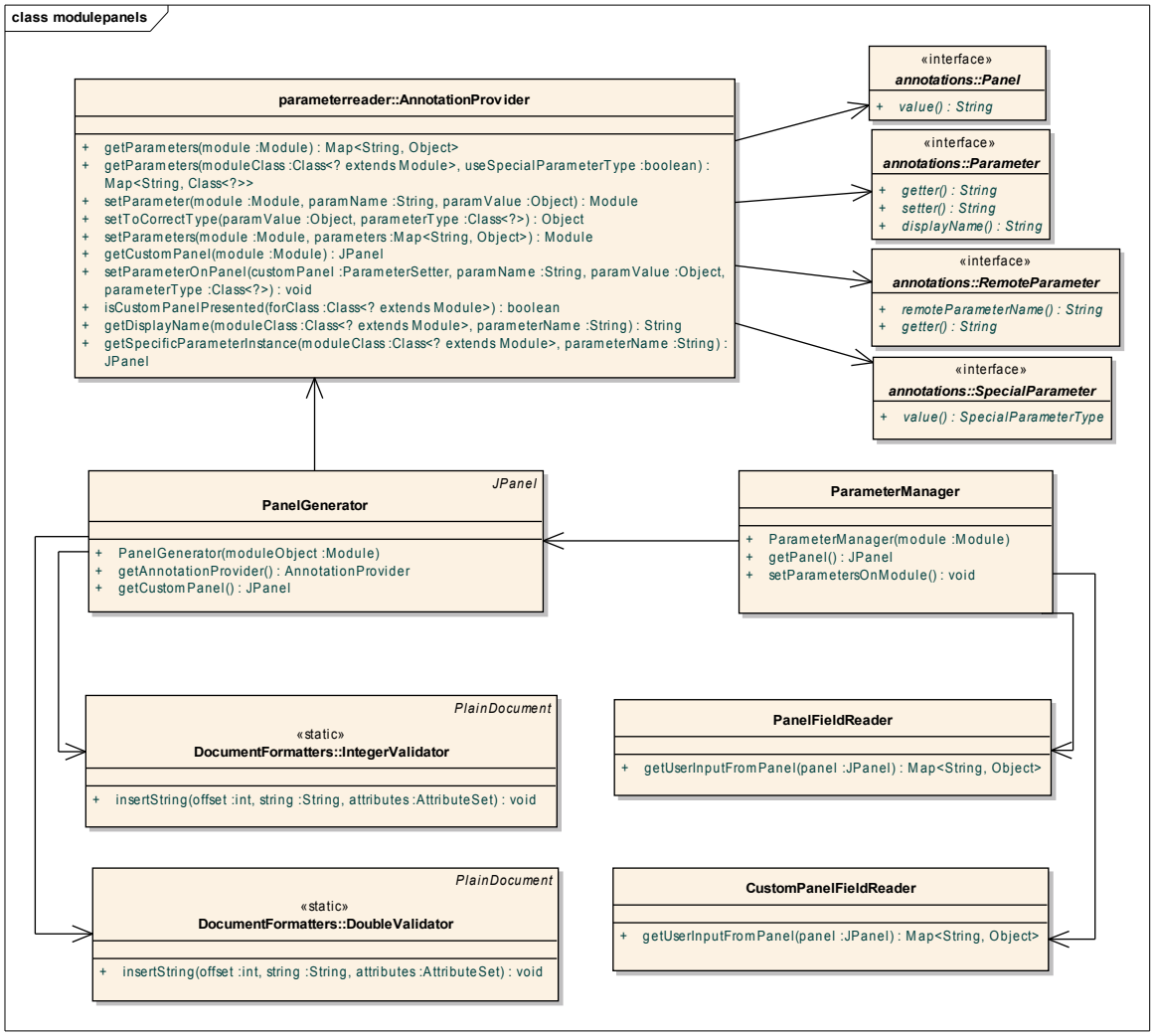
Druhá možnosť zadávania je pomocou **grafického sprievodcu** (trieda `GraphicWizard`). Tento sprievodca umožňuje vizuálne nastaviť poradie jednotlivých modulov a dátové toky medzi nimi. Na nastavenie jednotlivých modulov využíva triedu `Wizard`, ktorá sa spustí v špeciálnom móde s jednou záložkou pre daný typ modulu.

`AbstractWizard` – trieda obsahuje funkcionalitu, ktorá je spoločná pre triedy `Wizard` a `GraphicWizard`, ktoré od triedy `AbstractWizard` dedia. Jedna z dôležitých spoločných vlastností je aj tá, že obe triedy sprievodcov pracujú s objektom `Mitandao`.

Po definovaní procesu spracovania a spustení analýzy sa vyhodnotí celý tento proces a zapíšu sa výsledky analýzy. V prípade, že používateľ zvolil možnosť zobrazenia grafu aktivuje sa komponent `WorkPanel`, ktorý pomocou knižnice Prefuse zobrazí analyzovaný graf.

Mitandao UI Framework

Mitando UI Framework je časťou projektu Mitandao, ktorý slúži na uľahčenie vytvárania grafických používateľských rozhraní modulov do systému. Zabezpečuje automatické generovanie panelov na základe anotácií a synchronizáciu hodnôt medzi modulom a generovaným panelom. Z architektonického hľadiska je rozdelený na časť, ktorá slúži na komunikáciu s používateľom vygenerovaných panelov (`ParameterManager`), na časť, slúžiacu na nízkoúrovňovú prácu s modulom a panelom (`PanelGenerator`, `AnnotationProvider`, `PanelFieldReader` a `CustomPanelFieldReade`r) a na časť slúžiacu ako rozhranie pre programátora grafického modulu (anotácie). Na obrázku 7 je znázornený UML diagram tried systému Mitandao UI Framework.



Obrázok 7: Mitandao UI Framework

Triedy `PanelFieldReader` a `CustomPanelFieldReader` slúžia na čítanie hodnôt grafických komponentov z vygenerovaného panelu (používateľských vstupov).

Triedy `DoubleValidator` a `IntegerValidator` slúžia na overenie vstupných položiek.

Anotácie na spodnej časti diagramu slúžia ako rozhranie pre programátora modulu na vytváranie grafických modulov.

Trieda `PanelGenerator` predstavuje vygenerovaný grafický panel a tiež sa stará o generovanie položiek do tohto panela.

Trieda `AnnotationProvider` slúži na nízkoúrovňové čítanie anotácií z modulu.

Trieda `ParameterManager` poskytuje jednotnú prácu s funkcionalitou rámca.

Úlohy podsystému Mitandao UI Framework sa dajú rozdeliť na nasledujúce:

- Poskytuje jednoduché rozhranie pre používateľa knižnice na vytváranie grafických používateľských rozhraní pre vytvárané moduly
- Poskytuje jednotné rozhranie pre systém na jednotnú prácu s akýmkoľvek grafickým modulom vytvoreným pomocou Mitandao UI Framework

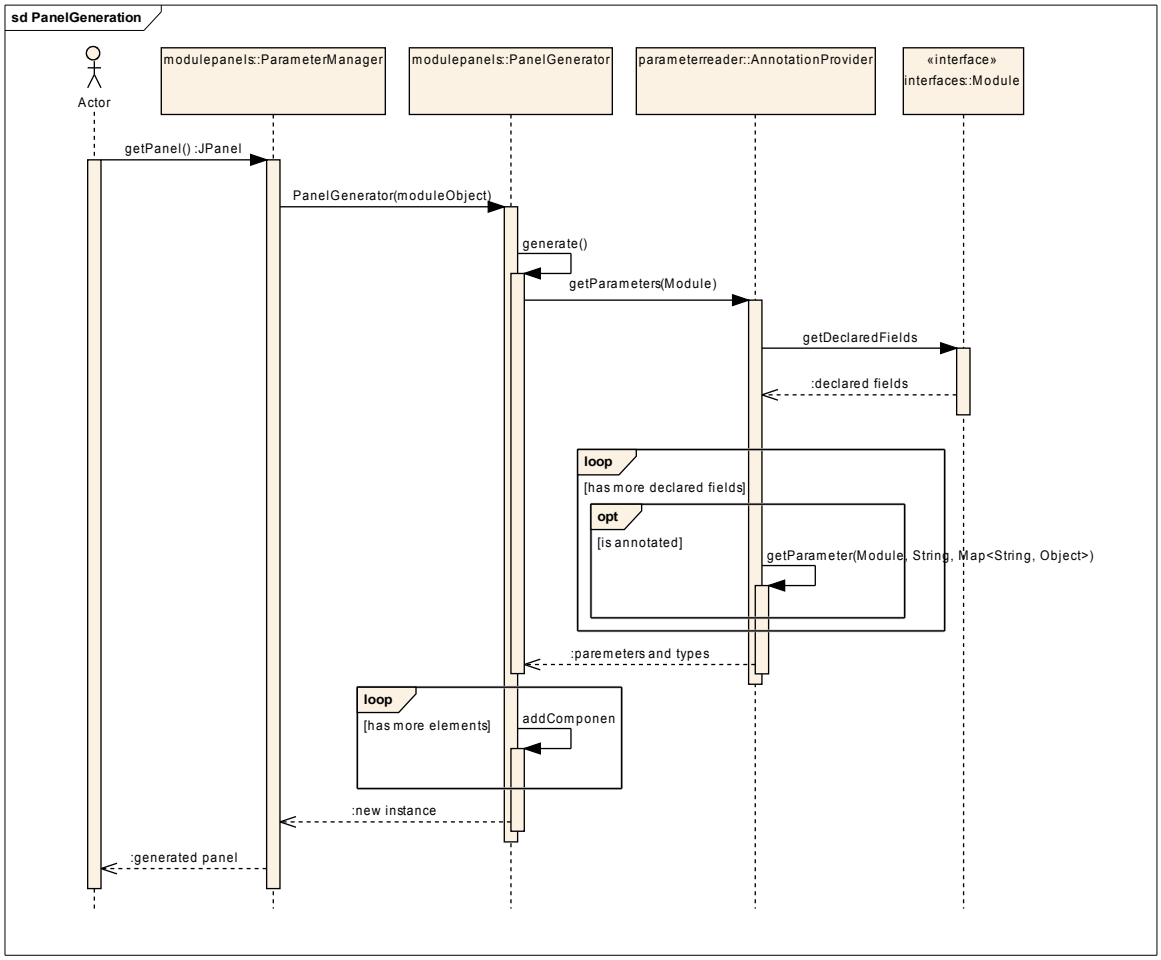
- Generuje grafické panely podľa používateľom definovaných anotácií a zabezpečuje kopírovanie hodnôt z grafických panelov (používateľské vstupy) do modulov a späť

Poskytnutie jednotného rozhrania pre programátora modulu. Rozhranie je riešené pomocou technológie Java Annotation. Mitandaou UI Framework definuje nasledujúce rozhrania:

- **Parameter:** Slúži na anotovanie premenných typu `java.lang.String`, `java.lang.Integer` a `java.lang.Double`. V prípade, že nie je použitá anotácia Panel, ku premenným anotovaným touto anotáciou budú vygenerované položky `JTextField` a `JLabel`. Položka `JTextField` bude mať nastavené formátovanie obmedzujúce používateľský vstup iba na typ premennej, ku ktorej je generovaný (napr. do `JTextField`-u generovanému ku premennej typu `Integer` bude možné vložiť iba hodnotu typu `Integer`). Hodnota položky `JLabel` bude rovnaká, ako je meno premennej, ku ktorej je generovaná (ak nie je prítomný nepovinný parameter anotácie `displayName`), v opačnom prípade bude mať hodnotu parametra `displayName`.
- **Panel:** Slúži na anotovanie tried modulov, ktoré majú byť prepojené s vlastným panelom, ktorý používateľ vyvinul. Jeho povinným parametrom je plné meno panela. Ak je použitý tento parameter, tak sa panely negenerujú, ale sa použije používateľom špecifikovaný panel.
- **RemoteParameter:** Slúži na anotovanie premenných na paneloch, ktoré vyvinul vývojár modulu. Premenné anotované na triede modulu anotáciou Parameter, ktoré majú rovnaké meno, ako premenné na strane panelu anotované anotáciou RemoteParameter, budú automaticky synchronizované.
- **SpecialParameter:** Slúži na anotovanie premenných na strane modulu typu `java.lang.String`. Prepája daný parameter so znovupoužiteľným panelom. Jej povinným parametrom je konkrétna hodnota z číselníku `SpecialParameterType`.

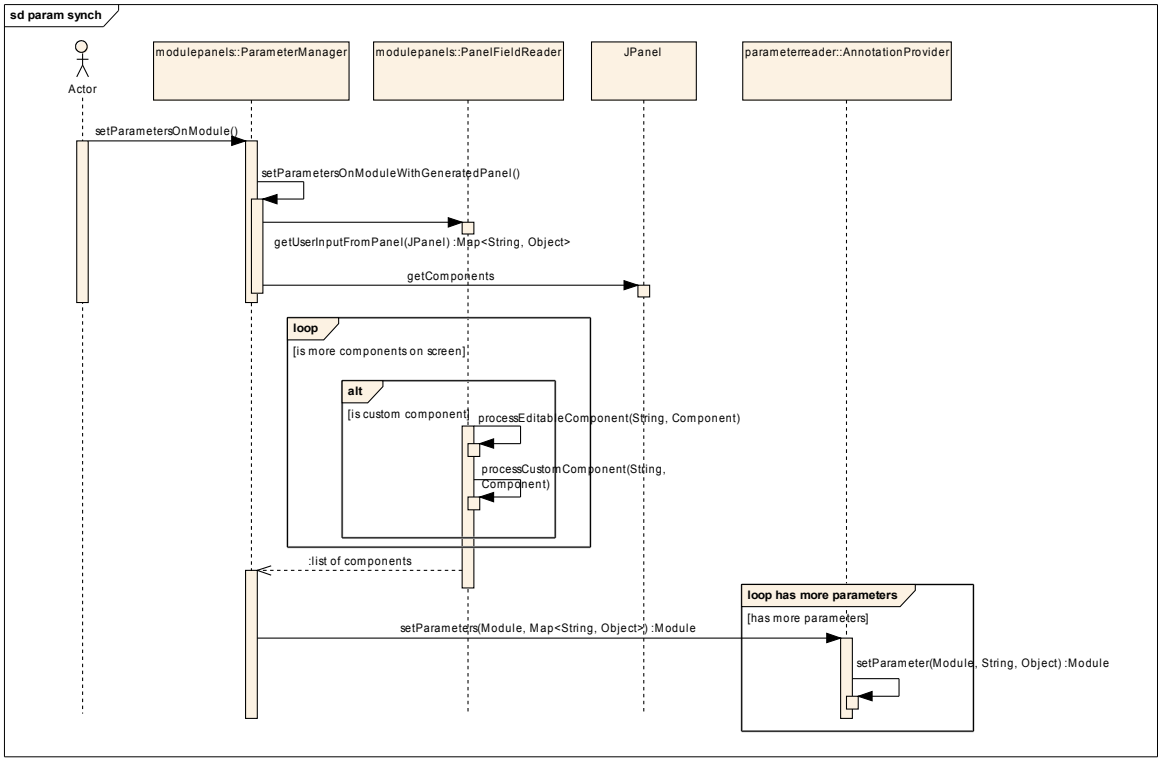
Poskytnutie jednotného rozhrania pre systém. Z pohľadu systému, ktorý je v tomto prípade časť GUI projektu Mitandao, poskytuje Mitandao UI Framework jednoduché rozhranie na prácu s GUI jednotlivých modulov. Trieda, pomocou ktorej pristupuje systém ku grafickým panelom modulov je trieda `ParameterManager`. Ku každému modulu sa používa jedna inštancia triedy `ParameterManager`. Vytvára sa pomocou konštruktora, ktorý má jeden vstup: Modul. Trieda ďalej definuje metódu `getPanel()`, ktorá vygeneruje panel (resp. použije panel definovaný pomocou anotácie `Panel`) a vráti jeho konkrétnu inštanciu. Druhá metóda, ktorú poskytuje, je metóda `setParametersOnModule()`. Táto skopíruje všetky hodnoty z obrazovky do patričných premenných modulu.

Generovanie grafických panelov a kopírovanie hodnôt. Na nasledujúcom obrázku, obrázok 8, je vidieť generovanie grafického panelu z anotácií Parameter alebo `SpecialParameter`. Druhou možnosťou je použitie parametra Panel, v takom prípade by sa získala inštancia panela podľa hodnoty parametra anotácie Panel a tá by sa vrátila používateľovi.



Obrázok 8: Generovanie grafického panelu

Na obrázku 9 je UML diagram kopírovania hodnôt parametrov z grafického panela (zväčša používateľský vstup) do modulu.



Obrázok 9: Kopírovanie hodnôt z grafického panela do modulu

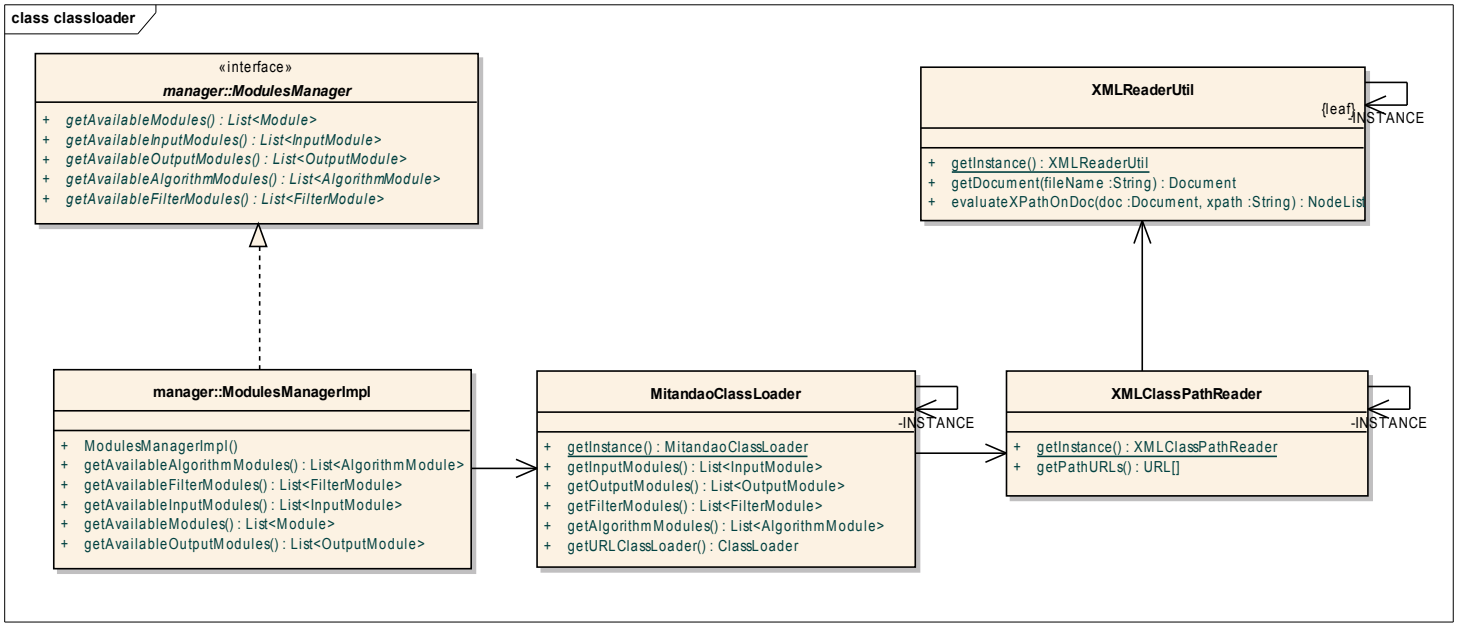
Načítavanie modulov

Súčasťou architektúry jadra je aj trieda `MitandaoClassLoader`, ktorá zabezpečuje načítanie všetkých prítomných modulov v adresároch definovaných v XML súbore `modules_paths.xml`. Trieda `MitandaoClassLoader` je volaná z triedy `ModulesManagerImpl`, ktorá definuje všetky požiadavky na funkčnosť načítavania modulov.

Rozhodli sme sa nepoužiť systémovú cestu ku triedam (CLASSPATH), pretože aplikácia potrebuje rekurzívne prehľadať celú adresárovú štruktúru, aby mohla zistiť, ktoré moduly sú dostupné a tie ponúknuť používateľovi. Na systémovej CLASSPATH môže byť nastavených veľmi veľa adresárov (aj pre rôzne aplikácie) a veľa iných tried okrem požadovaných modulov, čo by mohlo prehľadávanie veľmi spomaliť. Použili sme preto konfiguráciu pomocou XML súboru `modules_paths.xml`, ktorý musí byť umiestnený niekde na CLASSPATH a ktorý obsahuje cesty k adresárom, ktoré sa majú prehľadávať.

Na začiatku načítavania modulov aplikácia rekurzívne prejde všetky adresáre uvedené v konfiguračnom súbore a ak narazí na triedu, ktorá implementuje jedno z rozhraní `InputModule`, `OutputModule`, `AlgorithmModule` alebo `FilterModule`, pridá ju do zoznamu tried (pre každý typ existuje samostatný zoznam). Na samotné vytvorenie tried používame java triedu `URLClassloader`, ktorej nastavíme URL adresy získané z konfiguračného súboru (Obrázok 10).

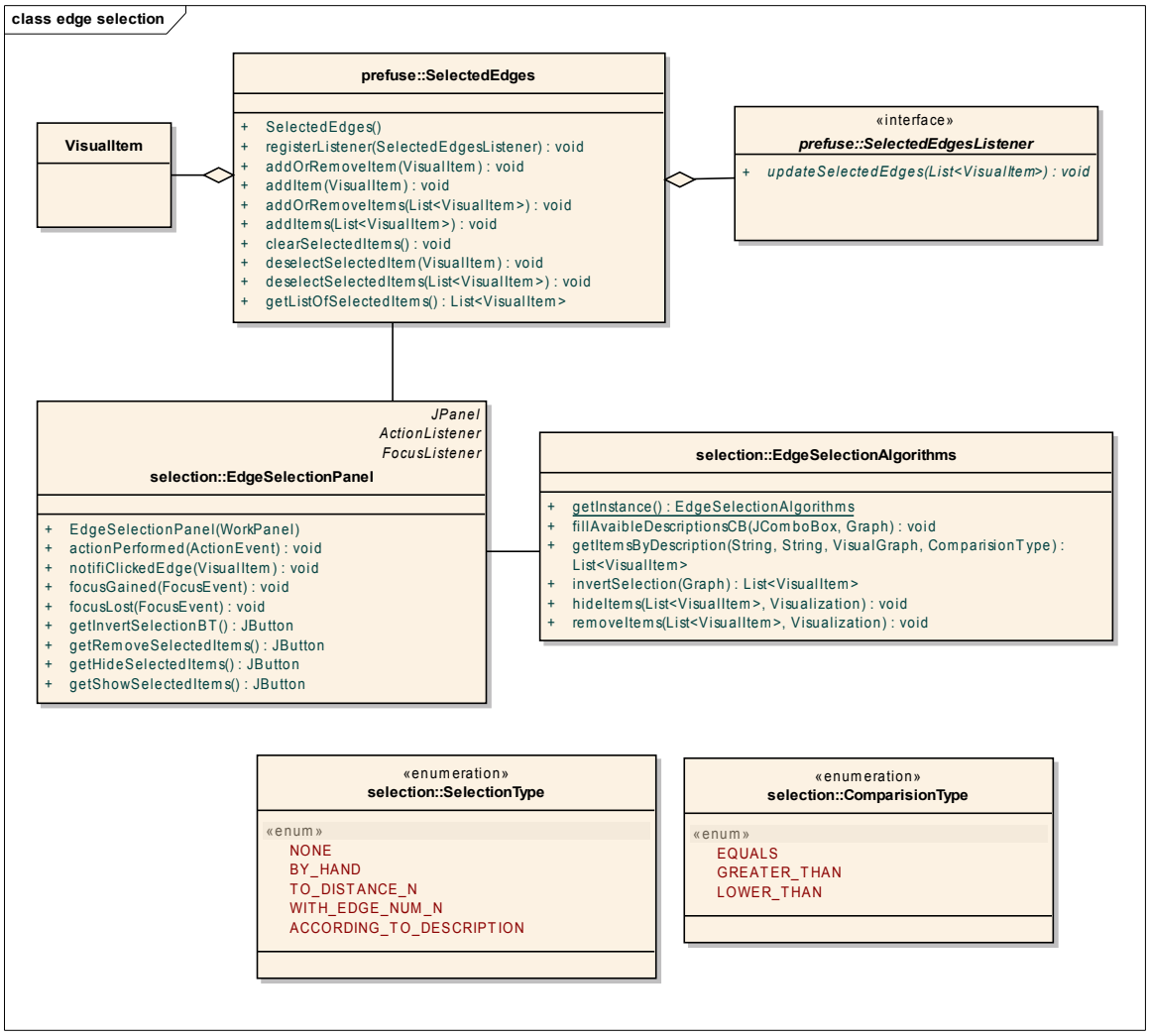
Po získaní zoznamu tried je tento už pre celý beh aplikácie nemenný. Zo zoznamu tried sa pri každej požiadavke na získanie dostupných modulov vytvoria nové inštancie daných tried (aby bolo možné použiť jeden modul vo workflow viackrát).



Obrázok 10: Class diagram k architektúre modulov

Grafické vyberanie vrcholov a hrán

Vyberanie vrcholov a hrán sú dva od seba nezávislé, ale podobné subsystémy projektu Mitandao, ktoré sa používajú na vyberanie vrcholov a hrán na zobrazenom grafe. UML diagram tried subsystému na vyberanie hrán je na obrázku 11, UML diagram tried subsystému na vyberanie vrcholov je analogický.



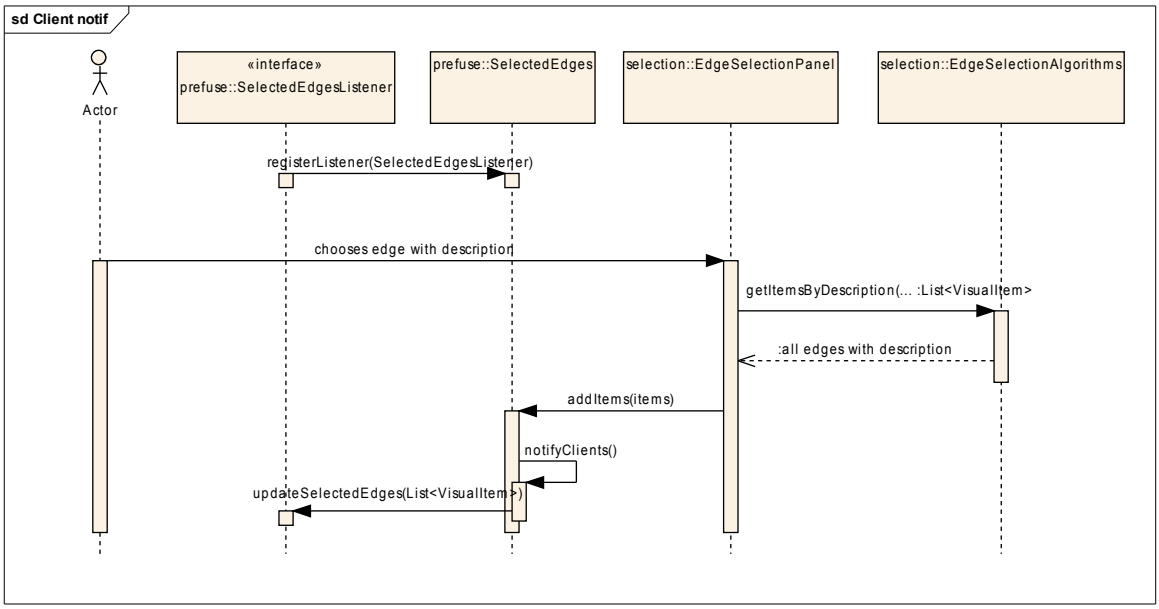
Obrázok 11: Vyberanie hrán

Trieda `NodeSelectionPanel` resp. `EdgeSelectionPanel` je grafický panel a v aplikácii je zobrazený v pravej časti ako tab Node (Edge) Selection. Obsahuje grafické komponenty, pomocou ktorých môže používateľ vyberať jednotlivé funkcionality, ktoré tento subsystém poskytuje. Táto trieda slúži aj ako listener komponentov na panely. Je spojená s triedou `NodeSelectionAlgorithms` resp. `EdgeSelectionAlgorithms`. Táto trieda je vytvorená aplikovaním návrhového vzoru jedináčik, obsahuje algoritmy, ktoré korešpondujú s funkciami, ktoré panel poskytuje (napr. vráti všetky vrcholy s počtom hrán N).

Zoznam všetkých vybratých vrcholov resp. hrán je v triede `SelectedItems` resp. `SelectedEdges`. Na túto triedu sa môžu registrovať také triedy, ktoré potrebujú byť notifikované, ak sa zoznam vybratých vrcholov resp. hrán zmení. Trieda, ktorá sa môže registrovať musí spĺňať rozhranie `SelectedItemsListener` resp. `SelectedEdgesListener`. Takáto trieda je napríklad trieda `ResultsPanel`, ktorá vypisuje zoznam všetkých zvolených vrcholov a hrán, a ich atribúty. Ide o aplikovanie návrhového vzoru Observer.

Pre názornosť je na nasledujúcom obrázku sekvenčný UML diagram, na ktorom

sa najprv registruje listener, potom používateľ zvolí algoritmus na vybratie všetkých hrán podľa popisu, tie sa vyberú a klienti sú o tejto zmene notifikovaní, obrázok 12.



Obrázok 12: Notifikovanie klientov

Ukladanie konfigurácie procesu analýzy

V prípade, že používateľ uloží nadefinovanú konfiguráciu procesu analýzy, táto konfigurácia sa uloží do XML súboru. V tomto XML súbore je uložená štruktúra prepojení jednotlivých modulov a zároveň aj nastavenia pre jednotlivé moduly. Štruktúra XML súboru, do ktorého sa konfigurácia uloží je nasledovná:

```
<?xml version="1.0" ?>
<configuration>
    <nodes>
        <node prefix="" type="input" id="0">
           <module type="sk.fiit.mitandao.modules.inputs.PajekReader">
                <String filePath="C:\Documents and Settings\bubo\Deskto
p\testdata\1.NET"/>
           </module>
        </node>
        <node prefix="" type="filter" id="1">
           <module type="sk.fiit.mitandao.modules.filters.RemoveNodes">
                <Integer firstSpinNum="0"/>
                <Integer secondSpinNum="1"/>
                <String thirdChoice="equal"/>
                <String firstChoice="equal"/>
                <String secondChoice="OR"/>
```

```
            </module>
        </node>
        <node prefix="" type="analyzer" id="2">
            <module type="sk.fiit.mitandao.modules.algorithms.Betweenes
s"/>
        </node>
        <node prefix="" type="analyzer" id="3">
            <module type="sk.fiit.mitandao.modules.algorithms.DegreeDis
tribution"/>
        </node>
        <node prefix="" type="output" id="4">
            <module type="sk.fiit.mitandao.modules.ouputs.PajekWriter">
                <String filePath="E:\output.net"/>
            </module>
        </node>
    </nodes>
    <connections>
        <edge source="0" target="1"/>
        <edge source="1" target="2"/>
        <edge source="1" target="3"/>
        <edge source="3" target="4"/>
        <edge source="2" target="4"/>
    </connections>
</configuration>
```

- celá konfigurácia je uzavretá do elementu **<configuration/>.**
- zoznam modulov je uzavretý v elemente **<nodes/>** a jednotlivé moduly sú uzavreté v elemente **<node/>.** Tento element obsahuje atribúty **id**, **typ** modulu a **prefix** ikony, ktorou sa má modul vizualizovať. Obsah tohto modulu je už závislý na vlastnostiach a nastaveniach každého modulu.
- zoznam prepojení je uzavretý do elementu **<connections/>.** Samotné prepojenia sú reprezentované elementami **<edge/>** , ktoré obsahujú atribúty **source** a **target**. Tieto atribúty reprezentujú zdrojový (**source**) a cieľový (**target**) modul

V aplikácii Mitandao je možné ukladať aj nastavenia jednotlivých súborov. Tieto nastavenia sa ukladajú do jednoduchého XML súboru s nasledovným formátom:

- nastavenie je uložené v elemente **<module/>**. Tento element obsahuje atribút **type**, ktorý definuje typ uloženého modulu.
- samotné nastavenia modulu sú uložené do samostatných elementov pre každé elementárne nastavenie. Názov elementu elementárneho nastavenia je rovnaký ako jeho údajový typ v jazyku Java.

```
<?xml version="1.0" ?>
    <module type="sk.fiit.mitandao.modules.inputs.PajekReader">
        <String filePath="C:\Documents and Settings\bubo\Desktop\testda
ta\1.NET" />
    </module>
```

Technická dokumentácia zdrojových kódov

Zdrojové kódy a ich zaujímavé časti sú zdokumentované vo forme dokumentácie JAVADOC. Táto vygenerovaná dokumentácia sa nachádza na priloženom elektronickom médiu v časti

/Dokumentácia projektu/Technická dokumentácia (JAVADOC)/

Viac informácií o obsahu elektronického média je v časti Príloha G: Obsah elektronického média.

Príloha G: Obsah elektronického média

Priložené CD obsahuje nasledovné adresáre a súbory:

PROJEKT MITANDAO

Dokumentácia projektu – obsahuje finálnu verziu dokumentácie projektu, ako aj používateľské príručky pre každý typ používateľa, používateľa aplikácie, knižnice alebo vývojára.

Webová stránka projektu – obsahuje off-line verziu webového sídla projektu, zo dňa 19.05.2008

Aplikácie Mitandao – Social Network Analyzer – spustiteľná verzia aplikácie

Knižnica Mitandao

Zdrojové kódy projektu

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHONOLÓGIÍ

Časť II. - Riadenie

Obsah

Úvod

Časť dokumentácie s názvom Riadenie obsahuje dokumenty súvisiace s manažmentom projektu.

Ponuka vznikla ešte pred začatím prác na projekte analyzátor sociálnych sietí. Ide o dokument na oslovenie zákazníka, pedagogického vedúceho. Jej úlohou bolo pedagogického vedúceho presvedčiť, že práve náš tím je ten najlepší pre riešenie zadanej úlohy.

Táto časť obsahuje tiež plán na zimný a letný semester. Zaradili sme sem aj zápisnice zo stretnutí s pedagogickým vedúcim tímu, rozdelenie manažérskych funkcií medzi členov tímu a tiež tabuľku podielu práce jednotlivých členov na dokumentácii. Do tejto časti dokumentácie sme sa rozhodli zaradiť aj časť s názvom štábna kultúra, v ktorej sme spísali pravidlá, ktorými sme sa riadili pri písaní zdrojových kódov. Na záver sme pridali posudky, jednak nášho tímu na projekt tímu číslo 1 a tiež ich posudky na náš projekt.

Ponuka

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ponuka

Tímový projekt

Tím SOCKY (socky@kmit.sk)

Lucia Jastrzembská
Tomáš Jelínek
Tomáš Konečný
Katarína Kostková
Ľuboš Omelina

Tím

...je zložený z ľudí s rôznym zameraním od databázových technológií cez umelú inteligenciu až po humanitné vedy.
Viacerí členovia tímu už v minulosti preukázali svoju schopnosť samostatne aj v tíme riešiť zložitejšie problémy a prichádzať s kreatívnymi nápadmi. Dôkazom toho sú umiestnenia na popredných miestach v prestížnych súťažiach (ImagineCup) a vedecké publikácie aj na medzinárodnej úrovni. (International conference on web intelligence 2007 v Silicon Valley).

Zloženie tímu:
Lucia Jastrzembská - zaujíma sa o oblasť umelej inteligencie a jej aplikáciu na riešenie bežných problémov. Tejto téme sa venovala i vo svojej bakalárskej práci, v ktorej sa jej podarilo aplikovať sociálne správanie sa včiel na vyhľadávanie na Internete. So svojou prácou sa zúčastnila viacerých konferencii. Má praktické skúseností s objektovo orientovaným programovaním v jazyku JAVA, ovláda tiež jazyky PHP, HTML, CSS.

Tomáš Jelínek – svoj záujem o umelú inteligenciu prejavil vo svojej bakalárskej práci, v ktorej sa bližšie zaujímal o vyhľadávanie na Internete a s ktorou sa zúčastnil viacerých konferencií (IIT-SRC, Kognícia a umelý život VII).
Má skúsenosti s prácou v menšom tíme aj s prácou na väčších projektoch. Má praktické skúsenosti s programovacími jazykmi Java, PHP, C a s databázovými technológiami MySQL a Oracle. Má teoretické aj praktické znalosti operačného systému Linux.

Tomáš Konečný - ako jediný člen tímu študuje na študijnom odbore Informačné systémy, čomu zodpovedá i jeho záujem o databázové technológie. Okrem objektovo orientovaných jazykov JAVA a C++ ovláda tiež jazyky SQL a PL/SQL, prácu s MySQL a Oracle. Praktické skúsenosti s J2EE a databázovými technológiami nadobudol aj na projekte v práci, kde je súčasťou väčšieho tímu, čo je taktiež prínosom pre tento predmet. Ako absolvent 4-ročného bakalárskeho štúdia disponuje širšou bázou absolvovaných predmetov, čo môže byť taktiež pre tím prínosom.

Katarína Kostková - Praktické zručnosti a skúsenosti s objektovým programovaním v jazyku JAVA. Medzi záujmy patrí tvorba stránok (HTML, CSS). Medzi obľúbené predmety bakalárskeho štúdia patrili Komunikačné systémy a Právo informačných a komunikačných technológií, ktorému sa venovala aj v bakalárskej práci. Má dobré komunikačné schopnosti, či už v slovnej alebo písomnej forme. Počas štúdia absolvovala dva semestre psychológie kde získala aj poznatky zo sociálnej psychológie. Získané poznatky priamo súvisia s problematikou sociálnych sietí a tak nimi môže prispieť k riešeniu projektu.
Ľuboš Omelina – Medzi jeho záujmy patrí robotika a umelá inteligencia, štúdiu ktorých sa rád venuje vo voľnom čase. Nadobudol akademické, ale aj pracovné skúsenosti s prácou v tíme. Má skúsenosti s viacerými programovacími jazykmi (C, C++, C#, JAVA, PHP, assembler) a technológiami (.NET, GTK). Medzi jeho úspechy patrí členstvo vo víťaznom tíme v národnom kole súťaže ImagineCup, kde sa so svojím tímom dostal do celosvetového finále tejto súťaže. V pracovnom živote sa aktívne podieľal aj na väčších projektoch, najmä z oblasti riadiacich a kontrolných systémov.

Medzi veľké prednosti nášho tímu patrí fakt, že všetci členovia absolvovali predmet Umelá inteligencia a jedna členka absolvovala predmet psychológia.

Motivácia

Všetkých členov tímu najviac zaujala téma Analyzátor sociálnych sietí a po spoločnom zhodnotení sme prišli k záveru, že naša práca na tomto projekte by mohla byť prínosná pre obe strany.

Po predstavení tejto témy si každý z nás vytvoril vlastnú predstavu o úlohe a realizácii projektu, táto predstava sa však samozrejme prácou a konzultáciami na projekte bude meniť. Predstavy jednotlivých členov tímu sa pretransformujú vo výsledný jednotný

produkt. Práca na projekte by nám mohla priniesť všeobecný prehľad v oblasti sociálnych sietí, ktoré už dávno nereprezentujú iba vzťahy ľudských jedincov. Sociálne siete reprezentujú tiež vzťahy webových stránok a dokumentov na internete, vzťahy medzi organizáciami, vzťahy v komunikačných sieťach nielen ľudského charakteru a celkovo vzťahy informácií získaných z najrôznejších zdrojov dát.

Projekt, ktorý sme si vybrali, má výskumný charakter a riešenie problémov nám môže poskytnúť veľmi dobrú skúsenosť v oblasti výskumných projektov a postupov.

Výber témy ovplyvnila aj skutočnosť že výstupný produkt môže byť užitočný a to nielen pre ľudí z úzkej tematickej oblasti, ale aj pre širokú verejnosť. Produkt by mohol mať pomerne dobré uplatnenie v oblasti knižničných systémov, ale aj v iných oblastiach.

Projektu môžeme poskytnúť naše skúsenosti s databázovými technológiami aj umelou inteligenciou. Zaujímavá by tiež bola možnosť overiť model hľadania informácie inšpirovaný sociálnym hmyzom na netriviálnej vzorke dát v databáze. Veríme, že projektu budú prínosné aj naše skúsenosti s prácou na veľkých mimoškolských projektoch. Záruky a kvalitu našej práce môžu poskytnúť naše predošlé úspechy v oblasti informatiky.

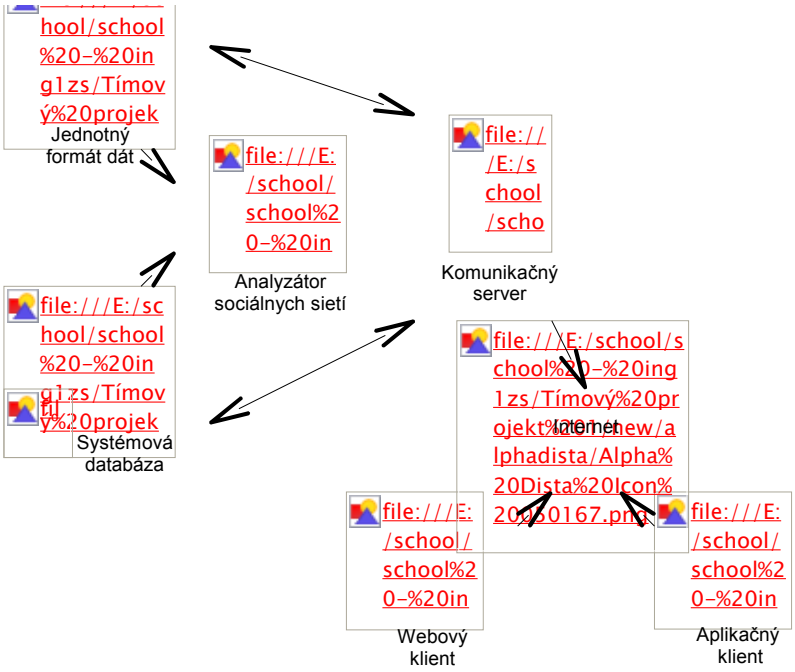
Návrh systému

Náš tím môže poskytnúť riešenie problému analyzovania sociálnych sietí vo forme aplikácie vyvinutej v jazyku JAVA. V prípade požiadavky je možné systém rozšíriť o webové rozhranie. Základná architektúra navrhovaného systému je uvedená na obr. 1.

Systém sa skladá z databázy obsahujúcej údaje, z ktorých sa analyzuje sociálna sieť. Tieto údaje spracúva analyzátor, ktorý si spôsob analýzy, zvolené požívateľské preferencie a nastavenia načítava zo systémovej databázy. Do tejto databázy analyzátor ukladá aj svoje dočasné výpočty a analyzované údaje.

Základná architektúra systému

Nasledujúci diagram zobrazuje navrhovaný systém z architektonického hľadiska. Komunikačný server zabezpečuje prenos analyzovaných údajov a pôvodných údajov pred analyzovaním na klientské aplikácie. Tie sa pripájajú na komunikačný server prostredníctvom Internetu.

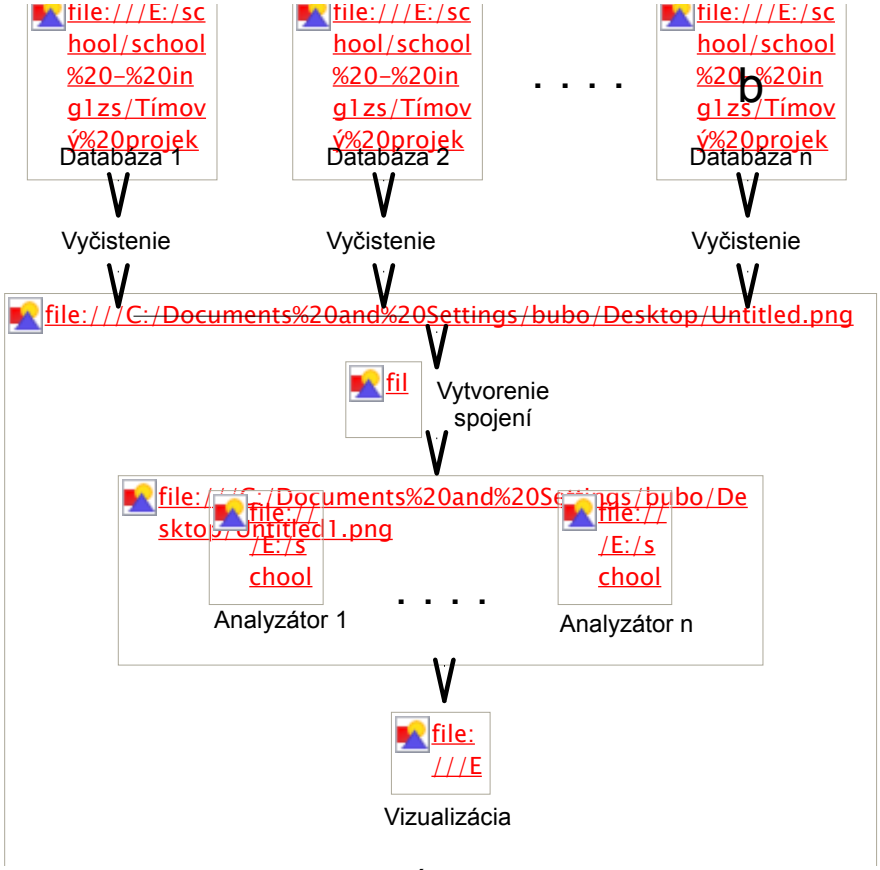


Obr. 1. Model architektúry systému

V týchto aplikáciách sa vizualizujú výsledky analýzy vo forme tabuliek a grafov vo forme zrozumiteľnej používateľovi. Tieto aplikácie popri sprostredkovaní výsledkov analýzy umožňujú užívateľovi zadávať, nastavovať spôsob a predmet analýzy údajov. Celý systém je navrhnutý tak, aby bol pre používateľa jednoduchý na ovládanie, ale zároveň plnil svoju funkcionalitu.

Vnútorná funkcionalita

Nasledujúci diagram zobrazuje navrhovanú architektúra systému z funkčného hľadiska.



Obr. 2. Model vnútornej funkcionality

Databázy, ktoré budú k dispozícii, budú obsahovať údaje v nespracovanej forme tak, ako sa stiahli z Internetu alebo získali iným spôsobom. Každá takáto databáza potrebuje mať vlastný filter, ktorý slúži na prečistenie údajov, čím chápeme vytvorenie jednotnej štruktúry dát, s ktorými sa ďalej môže pracovať jednotným spôsobom. Tieto dáta sa ďalej spracovávajú (vytvára sa daná sociálna sieť) na základe požiadavkov od používateľa. Takouto požiadavkou môže byť vytvorenie spojenia na základe (ne)priateľstva alebo iných vzťahov.

Ďalším krokom spracovávania je už vytvorenú sociálnu analyzovať. Nakoľko má daný systém podporovať viac spôsobov analyzovania siete, bude ich implementovaných niekoľko (samozrejme s možnosťou dodatočného implementovania ďalších). Dáta z týchto analyzátorov musia spĺňať jednotný formát, aby s nimi mohla pracovať klientská aplikácia. Táto môže byť implementovaná aj ako tenký klient, aj ako hrubý a ktorá môže poskytovať služby ako ukladania dát v rôznych formátoch, vykresľovanie grafov atď.

Technológie

Pri vypracovávaní tímového projektu sme sa rozhodli používať najmodernejšie otvorené technológie.

Jazyk a vývojové prostredie

- JDK 6
- Eclipse 3.3
- PostgreSQL

Softvér na podporu tímovej práce

- SVN
- WIKI

Z toho vyplýva požiadavka na server s dostatočnou kapacitou a verejnou IP. V prípade potreby si ho vieme zaobstarať sami.

Ostatný softvér

- Open Office 2.3

Za predpokladu, že daný softvér nie je k dispozícii v laboratóriu pre tímový projekt, všetci členovia tímu vlastnia notebook.

Preferované témy

1. Analyzátor sociálnych sietí
2. Distribuovaný systém na riešenie symetrickej hry
3. Informačný systém na komunikáciu s absolventami
4. Robocup – nové stratégie

Prílohy

Príloha A: rozvrh všetkých členov tímu

Príloha A: Rozvrh členov tímu

	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00
Pondelok	**APS** Kostková, Omelina			**NP** Kostková, Omelina **(začína v 4.tom týždni)**				**ML II** Konečný		**TSST/TIST** Jastrzembská, Jelínek, Konečný, Kostková, Omelina		**VSS/VSI** Jastrzembská, Jelínek, Konečný, Kostková, Omelina		
						OOAaNS Jastrzembská, Jelínek		**OOAaNS** Jastrzembská, Jelínek						
Utorok							**Kódovanie** Kostková		**MSI** Jastrzembská, Jelínek, Kostková, Omelina		**MSI** Jastrzembská, Jelínek, Kostková, Omelina			
						PDT Konečný						**BMIS** Konečný	**BMIS** Konečný	
Streda	**NS** Jastrzembská, Jelínek, Omelina		**NS** Jastrzembská, Jelínek		**NS** Omelina				**DD** Jastrzembská, Jelínek		**DD** Jastrzembská, Jelínek			
					PDT Konečný			**ML II** Konečný						
Štvrtok	**Kódovanie** Kostková			**NP** Konečný, Kostková, Omelina **(iba prvé tri týždne)**				**ASS** Jastrzembská, Jelínek, Kostková, Omelina						
										AIS Konečný				
Piatok	**NP** Konečný **(začína v 4.tom týždni)**			**NP** Konečný, Kostková, Omelina **(iba prvé tri týždne)**				**DPŠ** Kostková						

Plán

Plán na zimný semester

dátum	To do	done	Kto	Výstup
2. týždeň (02.10.07)	Štúdium materiálov o analýze sociálnych sieti, rozdelenie si funkcií v tíme	√ √	Všetci členovia tímu, na tímových stretnutiach	Dokument
3. týždeň (08.10.07)	Štúdium a analýza: - materiálov o sociálnych sieťach, - existujúcich softvérových riešení, - používaných algoritmov, - štúdium knižníc Jung a Prefuse - možností reprezentácie grafov, Vytvorenie webovej stránky projektu	 √ √ √ √ √ predĺžená	 Všetci členovia tímu Jastrzembská, Jelínek, Kostková, Omelina Jastrzembská, Jelínek Konečný Omelina Kostková, Omelina	 Dokument Dokument Dokument Dokument Dokument web
4. týždeň (15.10.07)	Špecifikácia požiadaviek (prípady použitia) na produkt, Štúdium algoritmov používaných na analýzu sociálnych sietí, inšpirácia pre náš návrh **nasadenie webovej stránky**	predĺžená √ √	Všetci členovia tímu Všetci členovia tímu Kostková	Dokument Dokument web
5. týždeň (22.10.07)	Špecifikácia požiadaviek (prípady použitia) Preskúmať možnosti riešenia Návrh riešenia problému	predĺžená √ predĺžená	Omelina Jastrzembská, Jelínek, Konečný Všetci	Dokument Dokument UML
6. týždeň (29.10.07)	Špecifikácia požiadaviek (prípady použitia) Návrh riešenia v zmysle architektúry systému	predĺžená predĺžená	Omelina, Konečný, Kostková Omelina, Konečný	Dokument UML

7. týždeň (05.11.07)	Dokončenie špecifikácie Dokončenie návrhu architektúry Dokončenie dokumentácie	√ √ predĺžená	Všetci Konečný, Omelina Kostková	Dokument, UML Dokument, UML Dokument
8. týždeň (12.11.07)	Dokončenie dokumentácie **Odovzdanie analýzy, špecifikácie a návrhu** Vypracovanie posudku	√ √	Všetci Všetci	Dokument Dokument
9. týždeň (19.11.07)	Vypracovanie posudku Zjemnenie návrhu **Odovzdanie posudku**	√ predĺžená	Všetci Všetci	Dokument Dokument
10. týždeň (26.11.07)	Implementácia prototypu - GUI - vykresľovanie v JUNG - vykresľovanie v Prefuse Anotácie Pluginy Zjemnenie návrhu	 √ √ √ √ √ predĺžená	 Kostková Jastrzembská Omelina Jelínek Konečný Všetci	 Kód Kód Kód Kód Kód Dokument
11. týždeň (03.12.07)	Integrovanie častí prototypu Prezentáciu na ontologickú pracovnú dieľňu Zjemnenie návrhu	√ √ √	Všetci Jastrzembská, Jelínek, Konečný Všetci	Kód Prezentácia Dokument
12. týždeň (10.12.07)	Dokončenie dokumentácie častí prototypu Integrovanie dokumentácie	√ √	Všetci Kostková	Dokument Dokument
13. týždeň (17.12.07)	**Odovzdanie a prezentácia projektu**	√		

Plán na letný semester

dátum	To do	done	Kto	Výstup
1. týždeň (18.02.2008)	- **odovzdanie posudku prototypu** - úprava GUI - wizard - graf zobrazenie a editovanie - interfejs knižnice - načítanie údajov - načítanie modulov	 √ √ √ pr pr	 Kostková Omelina Jastrzembská Jelínek Konečný	 kód kód kód kód kód
2. týždeň (25.02.2008)	- preskúmať používanie J Unit testov a Log 4 J - nastavovanie parametrov - načítanie modulov, vytvorenie viacerých inštancií modulu - integrácia kódu do projektov v SVN - editovanie grafu	√ √ pr pr pr	Jastrzembská Jelínek Konečný všetci Omelina	kód kód kód kód
3. týždeň (03.03.2008)	- prvá verzia aplikácie - načítanie - zobrazenie - načítanie modulov - editacia grafu, graficky prievodca - integrovanie kódu do SVN - buildovanie pomocou ANT-u	 √ √ pr √ √ √	 TJ ĽO, KK TK ĽO LJ všetci	 kód kód kód kód
4. týždeň (10.03.2008)	- uloženie do pajeku - vytvoriť moduly analýzy - možnosť spájania grafov - vytvorenie Abstraktného wizarda a následné úpravy - nastavenie hodnôt do panelu - načítanie modulov	√ √ √ √ √ pr	LJ LJ LJ ĽO, KK TJ TK	kód kód kód kód kód kód

5. týždeň (17.03.2008)	- vytvoriť moduly analýzy - Checkpointy, serializáciou - zovšeobecnenie FileChoosera - dynamické načítanie - čítanie z databázy - editovanie hrán - zobrazenie názvu vrcholu - práca s atribútmi grafu - nástroje na prácu z grafom JUNGu	√ pr √ √ pr √ √ √ √	KK KK TJ TK TK ĽO ĽO LJ LJ	kód kód kód kód kód kód kód kód kód
6. týždeň (24.03.2008)				
7. týždeň (31.03.2008)	- čítanie z databázy - aplikácia analýzy a filtra priamo - nelineárny workflow - pravé menu na info o uzloch - pravé menu na vyberanie vrcholov ručne, podľa vlastnosti ...	pr √ √ √ √	TK KK ĽO LJ TJ	kód kód kód kód kód
8. týždeň (07.04.2008)	- modul na načítanie z databázy - fiter modul - signalizácia nevyplnených modulov - ukladanie a načítanie nastavení modulov - ukladanie globálnych nastavení - reprezentácia atribútov vrcholov - zobrazenie výsledkov analýz v bočnom paneli	pr √ √ √ √ √ √	TK KK ĽO ĽO LJ LJ TJ	kód kód kód kód kód kód kód
9. týždeň (14.04.2008)	- tvorba dokumentácie - príručka pre používateľa knižnice - zobrazenie vlastností pre hrany - skrývanie vrcholov a hrán - technicka dokumentacia - anotacie - modul na čítanie z databázy	 √ √ √ √ pr	 LJ TJ TJ TJ TK	 doc kód kód doc kód

9. týždeň (14.04.2008)	- technická dokumentácia – classloader - wizard, čítanie grafu z workpanelu - osnova dokumentácie a doplnenie riadenia - používateľská príručka - ukladanie procesu analýzy - oprava chýb v grafickom wizardovi	pr √ √ pr √ √	TK KK KK KK, ĽO ĽO ĽO	doc kód doc doc kód kód
10. týždeň (21.04.2008)	- dodať svoju časť k dokumentácií produktu - z results odstrániť source a target - zdokumentovať anotácie, sk - odstránenie chýb v module na načítavanie z databázy - passwd písať ako hviezdičky - zdokumentovať classloader, en a sk - do main window dodať reagovanie na klávesové skratky - zjednocovať dokumentáciu - zmeniť infinity zobrazovanie (aby sa graf nehýbal) - ošetriť nulovosť modulov	√ √ √ pr pr √ √ √ √ √	LJ TJ TJ TK TK KK KK ĽO ĽO	doc kód doc kód kód doc kód doc kód kód
11. týždeň (28.04.2008) 11. týždeň (28.04.2008)	- úprava dokumentácie - spojenie dokumentácie - príprava CD - zbuildenie aplikácie **- odovzdanie produktu** **a dokumentácie k produktu** (používateľská príručka) - dokončovanie aplikácie a dokumentácie na druhé odovzdanie	√ √ √ √ √ √	všetci KK ĽO TK – spravili **TJ, LJ a ĽO** všetci všetci	doc doc cd kód kód a doc
12. týždeň (05.05.2008)	- nové načítavanie modulov - testovanie produktu konkurenčného tímu - oslovenie testerov - dokončenie databázového modulu	√ pr √	LJ a TJ TJ, TK, KK KK, ĽO	Kód doc

12. týždeň (05.05.2008)	- pridanie modulov - checkpointy - testovanie produktu - kompletizácia dokumentácie	pr √ √ pr pr	TK KK ĽO všetci KK	kód kód kód doc
13. týždeň (12.05.2008)	- posledné úpravy a dokončovanie **- odovzdanie celkového výsledku projektu (produkt so zmenami v rámci údržby, dokumentácia)**	√	všetci	kód a doc

Úlohy členov tímu

Roly členov tímu

Pre úspešné fungovnie tímu, je dôležité identifikovať a určiť role jednotlivých členov tímu. My sme si rozdelili nasledujúce funkcie v tíme.

Ľuboš Omelina	Vedúci tímu, Manažér podporných technológií
Lucia Jastrzembská	Manažér kvality
Tomáš Jelínek	Manažér rizík
Tomáš Konečný	Manažér vývoja
Katarína Kostková	Manažér plánovania

Okrem manažérskych rolí sme identifikovali aj nasledujúce.

Lucia Jastrzembská	Hlavní zodpovední za analýzu
Tomáš Jelínek	
Tomáš Konečný	Hlavný architekt
Katarína Kostková	Dokumentarista

Rozdelenie práce na projekte

Analýza	Všetci
Špecifikácia	Všetci
Návrh	Všetci
Dokumentácia	Všetci
Integrácia dokumentácie	Kostková
Web	Kostková, Omelina

Podiel práce na dokumentácii

Hlavným dokumentaristom v našom tíme je Katarína Kostková, ktorej úlohou bolo vytvoriť šablónu dokumentácie a dokumentáciu postupne integrovať. V nasledujúcej tabuľke je zaznamenaný podiel prác jednotlivých členov tímu na dokumentácii.

	Úvod	**Katarína Kostková**
1.	Analýza problémovej oblasti	**Lucia Jastrzembská** (1.1.1., 1.3.1.) **Tomáš Jelínek** (1.1.2., 1.2.1., 1.2.2., 1.2.3., 1.2.10., 1.3.1., 1.3.3.) **Tomáš Konečný** (1.2.8., 1.2.9.) **Katarína Kostková** (1.2.4, 1.2.5, 1.3.2) **Ľuboš Omelina** (1.2.6., 1.2.7., 1.3.2.)
2.	Analýza technológií a postupov	**Tomáš Konečný** (2.1., 2.3.) **Ľuboš Omelina** (2.2)
3.	Špecifikácia	**Lucia Jastrzembská** (3.1.) **Tomáš Jelínek** (3.1.) **Tomáš Konečný** (3.2.) **Ľuboš Omelina** (3.1., 3.2., 3.3.)
4.	Návrh	**Lucia Jastrzembská** (4.2) **Tomáš Jelínek** (4.2) **Tomáš Konečný** (4.1.) **Ľuboš Omelina** (4.1., 4.3.)
5	Podrobný návrh	**Lucia Jastrzembská**
6	Prototypovanie	**Tomáš Jelínek** (6.3) **Tomáš Konečný** (6.2) **Katarína Kostková** (6.4) **Ľuboš Omelina**(6.1)
7	Produkt	**Lucia Jastrzembská** (7.1, 7.2.1,) **Tomáš Jelínek** (7.1, 7.2.1, 7.3.1, 7.3.3, 7.3.4, 7.3.5) **Katarína Kostková** (7.1, 7.3, 7.4) **Ľuboš Omelina** (7.2.2, 7.3.2)
8	Záver	**Katarína Kostková**
	Príloha A	**Lucia Jastrzembská** **Tomáš Jelínek**
	Príloha B	**Ľuboš Omelina**
	Príloha C	**Tomáš Konečný** **Katarína Kostková** **Ľuboš Omelina**
	Príloha D	**Tomáš Jelínek**
	Príloha E	**Lucia Jastrzembská**

	Príloha F	**Lucia Jastrzembská** **Tomáš Jelínek** **Ľuboš Omelina**
	Príloha G	**Ľuboš Omelina**
	Riadenie	**Katarína Kostková**
	Štábna kultúra	**Lucia Jastrzembská** **Tomáš Jelínek**

Štábna kultúra

Dohodnuté konvencie pre projekt Mitandao

Menné konvencie

- všetky názvy budú v angličtine
- všetky balíčky sa budú písať iba malými písmenami
- všetky balíčky budú patriť do balíčka sk.fiit.mitandao
- mená všetkých tried a rozhraní budú začínať veľkým písmenom, rovnako ako každé ďalšie slovo v názve.

  Príklad:

  XMLParser

  MyClass
- mená metód a členských premenných budú začínať malými písmenami, každé ďalšie slovo v názve bude začínať veľkým písmenom. Skratka v názve bude celá malými písmenami, ak sa nachádza na začiatku názvu, v opačnom prípade bude celá veľkými písmenami.

  Príklad:

  xmlParser

  myXMLParser

  myMethod
- všetky gettery a settery budú mať názov get+meno premennej, set+meno premennej, okrem getteru na bollean hodnotu, ktorý bude mať názov is+názov premennej

Odsadzovanie

- na odsadzovanie kódu sa bude používať prednastavené odsadzovanie vo vývojovom prostredí Eclipse
- medzi menom metódy a prvou zátvorkou (nebude žiadna medzera
- otváracia zátvorka { sa nachádza na rovnakom riadku ako deklarácia
- uzatvárajúca zátvorka } sa nachádza na samostatnom riadku
- pri všetkých konštrukciách (if, for, while) sa budú používať zátvorky, aj keď budú obsahovať iba jeden príkaz (okrem prípadov, keď majú for a while prázdnu implementáciu)

  Miesto:

  if (podmienka) príkaz;

  Použiť:

```
if (podmienka) {
príkaz;
}
```
- príklad odsadzovania príkazu if-else

```
if (podmienka) {
        príkazy;
} else {
        príkazy;
}
```

- príklad odsadzovania try-catch

```
try {
        príkazy;
} catch (ExceptionClass e) {
        príkazy;
}
```

Prázdne riadky

- medzi deklaráciami tried a rozhraní v jednom súbore budú dva voľné riadky
- medzi deklaráciami premenných bude jeden voľný riadok
- medzi deklaráciami metód bude jeden voľný riadok
- prázdne riadky sa budú používať i na rozdelenie logických sekcií vrámci metódy

Deklarácie

- jednoúčelové premenné sa deklarujú čo najbližšie miestu použitia (napríklad v deklarácii for cyklu)
- premenné vyjadrujúce stav objektu sa deklarujú na začiatku triedy

Komentáre

- všetky komentáre sa budú písať v JavaDoc v anglickom jazyku
- komentáre budú vyjadrovať najmä účel a spôsob použitia, nielen prepísanie mena prvku
- každý java súbor bude obsahovať hlavičku, ktorá bude obsahovať účel súboru a meno autora
- každá trieda a rozhranie bude obsahovať popis účelu a použitia
- všetky public metódy (okrem getter a setter) budú obsahovať popis účelu a použitia
- všetky konštanty a enumerácie sa budú komentovať
- všetky ostatné metódy a členské premenné sa budú komentovať v prípade potreby

Príkazy

- jeden príkaz na riadok

  Nepoužívať:

  a++; b--;

  Používať:

  a++;

  b--;

Programátorské zvyklosti

- všetky premenné triedy budú private alebo protected a bude sa k nim pristupovať cez gettery a settery

- metódy budú mať public modifikátor iba v prípade, ak je to nutné a budú použité mimo balíčka
- preferuje sa použitie enumerácií pred konštantami
- preferuje sa premenná typu rozhrania pred premennou typu triedy

  namiesto:

  ```
  MyList list = new MyList()
  ```

  použiť:

  ```
  List list = new MyList()
  ```
- ak je to možné, preferuje sa používanie rozhraní ako argumentov metód
- preferuje sa používanie iterovania cez kolekciu pomocou for-each miesto iterátora

  namiesto:

  ```
  Iterator i = kolekcia.iterator();
  while(i.hasNext()){
          príkazy;
  }
  ```

  použiť:

  ```
  for (Prvok p : kolekcia){
          príkazy;
  }
  ```
- pri kolekciách sa budú vždy uvádzať generické typy (napríklad List<String>); výnimkou sú situácie, keď generické typy nepodporuje knižnica tretej strany
- pri používaní kolekcií je potrebné vyvarovať sa použitiu typu Vector, pretože je synchronizovaný a tým pádom pomalší. Preferuje sa používať ArrayList.
- v prípade odchytávania výnimky je potrebné nikdy nenechať prázdny catch blok, bude nutné minimálne zalogovať výnimku; v prípade, že to situácie vyžaduje, musí byť v komentári vysvetlený dôvod neošetrenia výnimky

Použité zdroje

1. Code Conventions for the Java Programming Language (**java**.sun.com/docs/codeconv)
2. Steve McConnel: Dokonalý kód. Computer Press, a.s. Brno, 2006.

Zápisnice zo stretnutí

Zápisnica zo stretnutia č.1

Dátum konania: 9.10.2007
Miesto konania: FIIT STU BA
Moderátor: Michal Barla
Zápis vypracoval: Ľuboš Omelina

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Na prvom stretnutí prebiehal úvod do projektu a oboznámenie sa s hodnotením ponuky. Boli rozdelené roli v tíme a začiatok analýzy problémovej oblasti. Každý člen tímu referoval svoje poznatky získané z knihy Introduction to Social Network Methods. Na konci stretnutia boli rozdelené úlohy pre každého člena tímu

Priebeh stretnutia

1. Oboznámenie s podmienkami a hodnotením projektu.
2. Oboznámenie sa s hodnotením ponuky a jej detailné rozobratie. Boli identifikované jej slabé stránky.
3. **Pridelenie rolí**:

   Lucia Jastrzembská - manažér kvality

   Tomáš Jelínek - manažér rizík

   Tomáš Konečný - manažér vývoja a hlavný architekt

   Katarína Kostková - manažér plánovania a dokumentarista

   Ľuboš Omelina - manažér podporných technológií a vedúci tímu
4. Referovanie prečítanej literatúry (Introduction to Social Network Methods)

   Lucia Jastrzembská - kapitoly 1 - 3

   Tomáš Jelínek - kapitoly 4 a 5

   Tomáš Konečný - kapitoly 10 a 11

   Katarína Kostková - kapitoly 6 a 7

   Ľuboš Omelina - kapitoly 8 a 9

5. Spoločné analyzovanie problémovej oblasti a vytvorenie obrazu o použití nových poznatkov v projekte.
6. Rozdelenie úloh
   - **Lucia Jastrzembská, Tomáš Jelínek** – Podrobná analýza softvéru Veka
   - **Katarína Kostková** - preštudovať UCINET, založiť dokumentáciu, návrh hrubého plánu, web
   - **Tomáš Konečný** - Analyzovať knižnice použiteľné na vizualizáciu grafov a sociálnych sietí
   - **Ľuboš Omelina** - Analyzovať formát ukladania grafu, spojazdniť a vytvoriť užívateľské účty SVN, TRACK, DOTproject
   - **Každý** – spísať preštudovanú časť knihy **Introduction to Social Network Methods**

Zápisnica zo stretnutia č.2

Dátum konania: 16.10.2007
Miesto konania: FIIT STU BA
Moderátor: Ľuboš Omelina
Zápis vypracoval: Katarína Kostková

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Zreferovali sme si úlohy z predchádzajúceho týždňa. Každý člen tímu povedal čo mal robiť, či je úloha hotová a zreferoval získané poznatky. Na konci stretnutia sme si dohodli úlohy na ďalší týždeň.

Priebeh stretnutia

1. Pripomenutie na čom sme sa dohodli minulý týždeň.
2. Diskusia o zistených poznatkoch, možnostiach znovu použitia a inšpirujúcich prvkoch existujúcich softvérových riešení.
3. Rozdelenie úloh
   - **Lucia Jastrzembská, Tomáš Jelínek** – preskúmať možnosť znovupoužitia prvkov softvéru Weka.
   - **Katarína Kostková** - dokončenie a nasadenie webovej stránky, zosumarizovanie doterajšej dokumentáciu, vloženie úloh členov tímu do dotProject-u.
   - **Lucia Jastrzembská, Tomáš Jelínek, Tomáš Konečný, Ľuboš Omelina** - Preštudovanie existujúcich softvérových riešení, špecifikácia požiadaviek na systém.

Zápisnica zo stretnutia č.3

Dátum konania: 23.10.2007
Miesto konania: FIIT STU BA
Moderátor: Katarína Kostková
Zápis vypracoval: Lucia Jastrzembská

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Hlavnou témou bolo dohodnutie detailov okolo funkčnej špecifikácie.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa.

Kto	Úloha	Stav
LJ, TJ	- preskúmať možnosť znovupoužitia prvkov softvéru Weka	Splnená
KK	- dokončenie a nasadenie webovej stránky, - zhrnutie a kompletizácia doterajšej dokumentáciu, - vloženie úloh členov tímu do dotProject-u	Splnené
LJ, TJ, TK, ĽO	- preštudovanie existujúcich softvérových riešení	Splnená
LJ, TJ, TK, ĽO	- špecifikácia požiadaviek na systém.	Predĺžená

2. Zhrnuli sa témy, ktoré sa preberali na neformálnom stretnutí 17. októbra.
3. Predstavenie novej webovej stránky.
4. Predstavenie softvérových riešení, ktoré boli preštudované počas týždňa.
5. Diskusia o podobe užívateľského rozhrania.
6. Diskusia o možnosti využitia distribuovaného počítania.

7. Diskusia o možnostiach vizualizácie grafov.
8. Rozdelenie úloh
   - **Lucia Jastrzembská –** preskúmať možnosti prúdového spracovania dát.
   - **Tomáš Jelínek** – preštudovať 7. kapitolu z knihy Web Mining.
   - **Tomáš Konečný** - zistiť riešenia distribuovaných algoritmov na analýzu soc. sietí.
   - **Katarína Kostková** - vytvorenie šablóny dokumentu.
   - **Ľuboš Omelina** - založenie SVN kont, práca na UC špecifikácii.
   - **Každý** – podieľať sa na dokončení špecifikácie.

Zápisnica zo stretnutia č.4

Dátum konania: 30.10.2007
Miesto konania: FIIT STU BA
Moderátor: Lucia Jastrzembská
Zápis vypracoval: Tomáš Konečný

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Na základe ďalších výsledkov z analýzy sme sa zaoberali došpecifikovaním funkčných požiadaviek a diskusiou o forme UC špecifikácie.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	- preskúmať možnosti prúdového spracovania dát	Splnená
TJ	- preštudovať 7. kapitolu z knihy Web Mining.	Splnená
KK	- vytvorenie šablóny dokumentu.	Splnené
TK	- zistiť riešenia distribuovaných algoritmov na analýzu sociálnych sietí.	Splnená
ĽO	- založenie SVN kont - práca na UC špecifikácii.	Splnená Predĺžená

2. Zhrnuli sa témy a úlohy z minulého týždňa s týmto výsledkom:
   - Pri skúmaní prúdového spracovania sme nenašli žiadne materialy ani postupy, ktoré by sa dali aplikovať na náš problém. Preto sme za rozhodli touto funkčnosťou ďalej nezaoberať.

- TJ nám zreferoval o prístupoch a algoritmoch z knihy Data Mining, pričom sme usúdili, že ich možno implementovať ako ďalšie zaujímavé pohľady na analýzu sociálnych sietí spolu s už predstavenými základnými algoritmami a metrikami, ktoré sa štandardne používajú.
- TK zreferoval zistené skutočnosti o problematike distribuovaných algoritmov pre sociálne siete. Pararelizovať niektoré algoritmy by bolo veľmi zložité, je to problematika sama o sebe preto sme usúdili, že vrámci nášho projektu na toto nebude časový priestor.
- KK nám predstavila novú formu a členenie dokumentu. Zhodli sme sa na spôsobe formátovania v podobe odrážok ako aj vyjadrovaní sa v 1. osobe množného čísla pri písaní dokumentov.
- LO nám predstavil svoj pohľad na UC špecifikáciu a nástroje na kreslenie UML diagramu. Nasledovala diskusia o „kráse“ výstupu z daného nástroja.

3. Opäť sme sa pozreli doplnenú webovú stránku.
4. Vrátili sme sa k problematike výberu používateľského rozhrania. Zatiaľ ostáva rozhodnutie použiť SWING. Pripojili sme aj pohľad na používateľské rozhranie vrámci architektúry, kde sme zvažovali ako pridávať do tohto rozhrania nové funkčnosti a nastavenie parametrov pre pridanú funkčnosť cez paneli.
5. Diskusia o podobe o architektúre a zabezpečeniu rozšíriteľnosti pomocou zásuvných modulov. Bol prezentovaný postoj k čo najmenšej miere používať xml a podobe riešenia cez reflexiu java tried.
6. Začali sme hovoriť o príprave metodík pre jednotlivé oblasti.
7. Rozdelenie úloh:
   - **všetci –** pracovať na špecifikácií v podobe UC modelov v UML.
   - **Lucia Jastrzembská** a **Tomáš Jelínek** budú neprítomní na nasledujúcom stretnutí, preto sa k daným problematikám budú vyjadrovať a prispievať názormi cez email.
   - **Ľuboš Omelina, Tomáš Konečný -** návrh architektúry

Zápisnica zo stretnutia č.5

Dátum konania: 06.11.2007
Miesto konania: FIIT STU BA
Moderátor: Tomáš Konečný
Zápis vypracoval: Ľuboš Omelina

Zúčastnení

Členovia tímu	Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	prof. Mária Bieliková

Téma stretnutia

Na stretnutí sme rozobrali ďalšie nedostatky špecifikácie a rozobrali sme doteraz vytvorené prípady použitia. Identifikovali sme nové prípady použitia a diskutovali o prvej verzii architektúry. Na tomto stretnutí sme mali hosťa prof. Máriu Bielikovú, ktorej sme predstavili náš projekt. Po skončení stretnutia s pedagogickým vedúcim, sme diskutovali o detailoch systému a snažili sme sa o vytvorenie jednotnej predstavy o vytváranom systéme.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
KK	- aktualizácia webu a zapracovanie vytvorených častí do dokumentu.	Splnené
TK	- návrh architektúry	Predĺžená
ĽO	- návrh architektúry	Predĺžená
Všetci	- pracovať na špecifikácií v podobe UC modelov v UML.	Predĺžená

2. Na začiatku stretnutia sme prebrali úlohy z minulého týždňa. Medzi úlohy patrilo dopracovanie špecifikácie a tvorba hrubého návrhu.

3. Zo špecifikácie sme diskutovali o UC ktoré vytvoril TK. Zhodnotili sme, že v UC by sa nemal vyskytovať systém ako účastník diagramu a tiež, že nebudeme vytvárať UC na štart aplikácie.

   V prvej verzii hrubej architektúry sme sa dohodli na ďalšom rozobratí časti používateľského rozhrania a architektonickom oddelení časti používateľskej aplikácie a knižnice, s ktorou táto aplikácia pracuje.

4. Na tomto stretnutí sa zúčastnil hosť prof. Mária Bieliková.

   - referovali sme vývoj celého projektu. Oboznámili sme ju s analyzovanou literatúrou a existujúcimi softvérovými riešeniami.

   - predstavili sme špecifikáciu a hrubý návrh nášho systému.

   - diskutovali sme o možnostiach a zámeroch, ktoré by sme mohli uplatniť v projekte.

   Záver:

   - orientovanie sa na znovupoužiteľnosť jadra aplikácie
   - sústrediť sa na vytvorenie menej častí systému, avšak ich hlbšie prepracovať

5. Dohodli sme sa na základných prípadoch použitia:
   1. Vstup a načítanie dát. /načítanie rozličných formátov/
   2. Aplikovanie filtra /máme rôzne typy hrán a uzlov, vyberieme si čo nás zaujíma/
   3. Aplikovanie algoritmu analýzy
   4. Výber zobrazenia/vykreslenia (typ grafu 2D, 3D …)
   5. Vizuálna úprava grafu /vizuálny filter/ - skrývanie /hide/ uzlov pre lepšiu prehľadnosť
   6. Štrukturálna úprava grafu vrámci zobrazenia zrušenie niektorých vrcholov/ hrán (narozdiel od skrývanie sa zmení graf a jeho atribúty teda počet uzlov, hustota, …...)
   7. Výstup: Uloženie výsledku ako grafu, čiže obrázku, uloženie dátovej reprezentácie grafu /štruktúra + pridaná hodnota výstupu z algoritmov, atribúty grafu, hustota, stupeň ulov, centrálny uzol/
   8. Registrovanie nového komponentu pre aplikáciu
   9. Nastavenia aplikácie
   10. Logovanie

6. Dohodli sme sa, že kartový sprievodca bude mať poradie kariet

   vstup -> filter -> algoritmus -> výstup

   pričom

   - vizuálne zobrazenie bude forma výstupu
   - k niektorým krokom v určitom prípade bude umožnené daný krok vynechať a to spôsobom, že v type jednotlivej akcie (napríklad výber typu algoritmu) bude možné vybrať typ **žiadny/neaplikovať**

- za skončením každého procesu sa uloží výsledok a aktuálny stav aplikácie (ang. checkpoint)

7. Zistili sme nedostatky použitia vybraného UML nástroja, pričom sme sa dočasne (kým sa nevrátia Lucia a Tomáš z USA a nevyjadria sa) dohodli, že budeme používať Enterprise Architect.

8. Úlohy na ďalší týždeň:
   - **Lucia Jastrzembská, Tomáš Jelínek, Ľuboš Omelina, Tomáš Konečný, Katarína Kostková**
     - pracovať na špecifikácií a prípadoch použitia
     - hrubý návrh architektúry jadra systému a aplikácie
     - dopracovanie obsahovej stránky dokumentácie
   - **Katarína Kostková**
     - práca na podrobnom pláne, zjednocovanie dokumentácie

Zápisnica zo stretnutia č.6

Dátum konania: 13.11.2007
Miesto konania: FIIT STU BA
Moderátor: Ľuboš Omelina
Zápis vypracoval: Tomáš Jelínek

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Témou stretnutia bolo predovšetkým prediskutovanie finálnej časti dokumentácie ako po formálnej tak aj po obsahovej stránke. Diskutovalo sa tiež o návrhu architektúry a o drobných úpravách v UC a v požiadavkách. Tiež sa diskutovalo o návrhu používateľského rozhrania.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
KK	- aktualizácia obsahovej časti dokumentácie	Splnené
TK	- návrh architektúry	Predĺžená
ĽO	- návrh architektúry	Predĺžená
Všetci	- pracovať na špecifikácií v podobe UC modelov v UML.	Predĺžená

2. Na začiatku stretnutia sme diskutovali o vytvorených UC. Záver bolo, že UC nastavenia premenujeme na UC zmena nastavení.
3. Ďalším bodom bola diskusia o architektúre systému. Riešili sme problém, do akej časti architektúry zaradiť knižnicu JUNG, nakoľko ju využíva aj jadro na internú reprezentáciu grafu, tiež sa však používajú algoritmy v ňom implementované.

   Záver: Knižnica JUNG bode v spodnej vrstve ako množina algoritmov, a bude sa

správať ako každý iný algoritmus, tiež však bude aj v jadre.

4. Ďalej sme diskutovali o dokumente.

   - Otázka bola, či sa budú v dokumentácii vyskytovať aj veci, ktoré sa možno vo finálnej fáze nebudú implementovať.

   Záver: Áno, budú.

   - Ďalšia otázka bola, či sa majú vymenovať externé knižnice, na ktorých náš systém stavia

   Záver: Áno.

5. Ďalej sme diskutovali o tom, či sa máme v popise GUI odkazovať na jednotlivé požiadavky zo špecifikácie.

   Záver: Nie.

6. Ďalšou vecou bolo riešenie otázok o formátovaní dokumentácie.

   - Otázka bola, či sa má odsadiť prvý riadok.

   Záver: Áno.

   - Ďalšia otázka bola, či použiť po každom odstavci prázdny riadok, alebo nie.

   Záver: Kompromisné riešenia – použije sa vynechanie väčšej medzery ako štandardnej, ale nie celý nový riadok (0.25cm).

7. Diskutovalo sa o návrhu.

   Záver: Je treba rozpísať core podrobnejšie, ideálne do tabuľky.

8. Zadelenie úloh na ďalší týždeň:
   - **Lucia Jastrzembská**
     - úprava požiadavky podľa UC
     - úprava návrhu GUI
   - **Tomáš Jelínek**
     - úprava požiadavky podľa UC
     - úprava návrhu GUI
   - **Tomáš Konečný**
     - upraviť UC6 – UC10
     - úprava návrhu
   - **Katarína Kostková**
     - Opraviť ostávajúce chyby v dokumentácii a jej integrovanie do jednotného celku
   - **Ľuboš Omelina**
     - upraviť UC1 – UC5
     - úprava návrhu

Zápisnica zo stretnutia č.7

Dátum konania: 20.11.2007
Miesto konania: FIIT STU BA
Moderátor: Tomáš Jelínek
Zápis vypracoval: Katarína Kostková

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Pedagogický vedúci nás oboznámil s postrehmi k dokumentácii. Pozreli sme si dokumentáciu tímu, ktorú máme hodnotiť a zreferovali si výhrady.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ a TJ	- úprava požiadaviek podľa UC - úprava návrhu GUI	Splnená
KK	- opraviť ostávajúce chyby v dokumentácii a jej integrovanie do jednotného celku	Splnené
TK	- upraviť UC06 až UC10 - úprava návrhu	Splnená
ĽO	- upraviť UC06 až UC10 - úprava návrhu	Splnená

2. Prešli sme si odovzdanú dokumentáciu, pedagogický vedúci nám povedal svoje

výhrady. Dokuemntácia nemala výraznejšie chyba a nedostatky. Do budúcnosti sme sa rozhodli odstrániť nasledovné chyby:

- obrázky v dokumentácii nielen číslovať, ale aj doplniť o výstižný popis
- ak je možné a vhodné uviesť ilustračný príklad, obrázok na ľahšie pochopenie javu opísaného textom

3. Dohodli sme sa na štruktúre posudku a na taktike začať a skončiť kladmi práce. Nevyzdvihovať nedostatky, spomenúť tie, ktoré môže posudzovaný tím využiť.
4. Rozdelenie úloh:

- **všetci –** zjemniť návrh, určiť čo budú ponúkať interfejsy.
- **Lucia Jastrzembská** a **Tomáš Jelínek -** zjednotiť pripomienky, spísať ich do viet, vytvoriť posudok
- **Tomáš Konečný –** vytvoriť prvu aplikaciu knižnic JUNG a Prefuse

Zápisnica zo stretnutia č.8

Dátum konania: 27.11.2007
Miesto konania: FIIT STU BA
Moderátor: Katarína Kostková
Zápis vypracoval: Lucia Jastrzembská

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Náplňou stretnutia bola predovšetkým diskusia o technológiách, ktoré by bolo možné použiť v projekte, najmä s ohľadom na to, či ich bude nutné prototypovať alebo nie.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ a TJ	- zjednotiť pripomienky, spísať ich do viet, vytvoriť posudok	Splnená
TK	- vytvoriť prvú aplikaciu knižnic JUNG a Prefuse	Splnená
všetci	- zjemniť návrh, určiť čo budú ponúkať interfejsy.	Predĺžená

2. Na začiatku stretnutia sme diskutovali o vytvorenom posudku pre druhý tím a o ich posudku pre náš projekt.

   Záver: Posudok druhého tímu bol prijatý a nebudú k nemu napísané žiadne pripomienky.

3. Ďalším bodom bola diskusia návrhu systému. Každý člen predstavil svoju verziu návrhu, ku ktorej následne ostatní povedali svoje pripomienky.

   Záver: Konečná verzia rozhrania pre prístup do systému.

4. Ďalej sme diskutovali o anotáciách v jazyku JAVA, Tomáš Jelínek predstavil túto technológiu ostatným členom tímu.

   Záver: Dohodli sme sa, že budeme túto technológiu v našej knižnici využívať na označenie parametrov pre jednotlivé moduly. Každá anotáciu bude môcť uviesť svoju getter a setter metódu, pričom o nastavenie parametrov sa bude starať jadro systému. Je ju však potrebné prototypovať, či ju bude možné v systéme použiť na dynamické nastavovanie parametrov.

5. Ďalším bodom bolo predstavenie JAVA technológie na dynamické načítavanie zásuvných modulov (Tomáš Konečný).

   Záver: Dohodli sme sa, že túto technológiu budeme prototypovať, či ju bude možné v systéme využiť.

6. Začali sme hovoriť o príprave metodík pre jednotlivé oblasti.

7. Nakoniec sme zhrnuli, čo bude potrebné prototypovať a dohodli si úlohy na prototype pre ďalší týždeň.

Lucia Jastrzembská

- vytvorenie grafu v JUNG a vykonanie záťažových testov

Tomáš Jelínek

- prototypovanie dynamického načítavania parametrov určených pomocou anotácií

Tomáš Konečný

- prototypovanie dynamického načítavania modulov

Katarína Kostková

- vytvorenie Sprievodcu

Ľuboš Omelina

- vytvorenie grafu v Prefuse a vykonanie záťažových testov

Zápisnica zo stretnutia č.9

Dátum konania: 04.12.2007
Miesto konania: FIIT STU BA
Moderátor: Lucia Jastrzembská
Zápis vypracoval: Tomáš Jelínek

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Témou stretnutia bolo diskutovanie výsledkov prototypovania. Tiež sme diskutovali, ako riešiť problémy, ktoré prototypovanie nastolilo.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	- prototypovanie a záťažové testy knižnice JUNG	Splnená
TJ	– prototypovanie dynamického nastavovania parametrov	Splnená
KK	- prototypovanie grafického používateľského rozhrania	Splnené
TK	- prototypovanie dynamického načítavania modulov	Splnená
ĽO	- prototypovanie a záťažové testy knižnice Prefuse	Splnená

2. Na začiatku stretnutia sme diskutovali prototypovanie TK. Nastal problém, že aj napriek tomu, že bolo spravené všetko podľa dokumentácie, výsledok nefungoval.

   Záver: úloha bola predĺžená do ďalšieho týždňa.

3. Ako druhý bod sme kontrolovali grafické používateľské rozhranie prototypované KK. Bolo vytvorené jednoduché rozhranie hlavnej aplikácie s dohodnutými tlačidlami a sprievodca na načítavanie grafov.

   Záver: prototyp bol prijatý, dohodli sme sa do neho dorobiť aj zobrazenie grafov.

4. Ako tretí bod sa diskutoval prototyp vytvorený TJ. Bola vytvorená jednoduchá knižnica na zisťovanie paramterických hodnôt z modulu a nastavovanie parametrov s príkladom aplikácie, ktorá túto knižnicu používa.

   Záver: prototyp bol prijatý.

5. Diskutovali sa výsledky záťažových testov vykonaných LJ a LO. Záťažové testy ukázali, že pri grafe s 50 vrcholmi a hranami medzi všetkými vrcholmi fungovala knižnica Prefuse spoľahlivo a rýchlo narozdiel od knižnice JUNG.

   Záver: na vizualizovanie bude použitá knižnica Prefuse.

6. Zadelili sa úlohy na ďalší týždeň

Lucia Jastrzembská, Tomáš Jelínek, Tomáš Konečný

- vytvoriť prezentáciu na ontologickú pracovnú dieľňu

všetci

- definovanie rozhraní

Zápisnica zo stretnutia č.10

Dátum konania: 11.12.2007
Miesto konania: FIIT STU BA
Moderátor: Tomáš Jelínek
Zápis vypracoval: Tomáš Konečný

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Finalizácia dokumentu, diskusia okolo prototypov a detailného návrhu.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ, TJ, TK	- vytvoriť prezentáciu na ontologickú pracovnú dieľňu	Splnená
všetci	- definovanie rozhraní	Splnená

2. Na pracovnej dielni PeWe bol od prezentovaný projekt a na základe odozvy vznikli návrhy na zváženie o doplnenie funkcionality mergovanie sietí, a undo, prípadne inej vizualzácií reprezentácie grafov.
3. Tomáš Jelínek prišiel z informáciou o možnosti modelovania workflow pomocou JUNG.
4. Prezentácia prototypu zásuvných modulov.
5. Zjemňovanie návrhu.
6. Diskusia o problematike spojenia Jung a Prefuse.
7. Diskusia o možnom termíne prezentácie projektu.
8. Rozdelenie úloh pre finalizáciu dokumentu:

- **Katarína Kostková** – Finalizácia dokumentu
- **Ľuboš Omelina** – Kontaktovanie druhého tímu za účelom dohodnutia prezentácie
- **Všetci** – dodať dokumentáciu prototypov.

Zápisnica zo stretnutia č.11

Dátum konania: 25.02.2008
Miesto konania: FIIT STU BA
Moderátor: Ľuboš Omelina
Zápis vypracoval: Katarína Kostková

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Zhrnutie činností členov tímu počas skúškového obdobia a v prvom týždni semestra. Zhodnotenie posudku prototypu, ktorý sme vypracovali pre tím číslo 1.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	- vytvoriť interfejs knižnice	Splnená
TJ	- vytvorenie modulu na načítanie údajov	Predĺžená
TK	- načítavanie modulov	Predĺžená
KK	- úprava GUI, wizarda	Splnená
ĽO	- zobrazenie a editovanie grafu	Predĺžená

2. Pedagogický vedúci skontroloval aktuálnosť webovej stránky, SVN a dotprojektu.
3. ĽO predviedol pokrok, ktorý dosiahol v editovaní grafu.
4. Ostatný členovia tímu slovne zhodnotili svoju minulotýždňovú prácu na projekte.

5. Diskutovali sme o použití J Unit testov, ktoré sme sa rozhodli zapracovať do systému.
6. Dohodli sme, že na úvod vypracujeme najjednoduchší cyklus, načítanie, zobrazenie a uloženie grafu, na ktorom budeme stavať ďalšie fukncionality.
7. Diskutovali sme o konvenciách písania kódu.
8. Rozdelenie úloh na ďalší týždeň:

   - **Lucia Jastrzembská** – vyrobiť projekt pre moduly a pripojiť ho na SVN, pozrieť sa na fungovanie J Unit testov, pozrieť **Log 4 J**
   - **Tomáš Jelínek –** pokračovanie v úlohe z minulého týždňa
   - **Tomáš Konečný** – pozrieť sa na možnosť vytvoriť viacero objektov načítaného modulu
   - **Katarína Kostková** – aktualizovať plán a pridať ho na web,
   - **Ľuboš Omelina** – pokračuje v úlohe z minulého týždňa, pozrieť sa na možnosť reprezentovať vrcholy obrázkami
   - **Všetci** – zaintegrovanie svojich častí kódu do projektov na SVN

Zápisnica zo stretnutia č.12

Dátum konania: 3.3.2008
Miesto konania: FIIT STU BA
Moderátor: Katarína Kostková
Zápis vypracoval: Lucia Jastrzembská

Zúčastnení

Členovia tímu	Tomáš Konečný Katarína Kostková Ľuboš Omelina Lucia Jastrzembská Tomáš Jelínek
Vedúci pedagóg	Michal Barla
Hostia	

Téma stretnutia

Hlavnou témou stretnutia bolo prediskutovanie problémov, na ktoré jednotliví členovia tímu narazili pri implementovaní svojich častí analyzátora.

Priebeh stretnutia

1. Kontrola úloh z minulého týždňa.

Kto	Úloha	Stav
KK	- vytvorenie Sprievodcu	Predĺžená
TK	- vytvorenie frameworku pre načítavanie modulov	Predĺžená
ĽO	- editácia grafu	Predĺžená
TJ	- vyrobenie frameworku pre dynamické nastavovanie parametrov	Splnené
LJ	- vyrobiť projekt pre moduly a pripojiť ho na SVN, pozrieť sa na fungovanie J Unit testov, pozrieť **Log 4 J**	Splnené

2. Ako prvý porozprával o svojich výsledkoch Tomáš Konečný. Počas týždňa sa

snažil riešiť problém s vytváraním modulov – nepodarilo sa mu však nájsť riešenie, preto mu bola úloha predĺžená. Zároveň sa dohodlo, že ak sa ho nepodarí vyriešiť konfiguračným súborom, pokúsi sa to riešiť prekrytím niektorých knižničných metód.

3. Tomáš Jelínek predstavil svoj modul na správu parametrov a názorne ukázal jej použitie.
4. Lucia Jastrzembská predstavila logovanie a tvorbu testov pomocou Junitov, ktoré počas týždňa zaintegrovala do workspace.
5. Ľuboš Omelina sa počas týždňa venoval najmä štúdiu a nastavovaniu SVN, pretože viacero členov tímu malo s tým pri synchronizácie svojej práce problémy.
6. Katarína Kostková bola pri vytváraní Sprievodcu závislá od práce všetkých členov tímu, preto sa jej úloha predĺžila do ďalšieho týždňa.
7. Nakoniec sa prediskutovala možnosť vytvorenia kontrolných bodov v aplikácii, takisto ako aj problém odchytávania výnimiek.
8. Zadelili sa úlohy na ďalší týždeň

Tomáš Konečný

- vytvoriť modul pre načítanie modulov

Lucia Jastrzembská

- vytvoriť načítanie a ukladanie grafu do / z formátu Pajek
- vytvoriť ant skript

Tomáš Jelínek

- vytvoriť špecializovaný modul na otváranie súborov
- kontrola vstupov
- vytvorenie Junit testov

Ľuboš Omelina

- vytvoriť modul na vytváranie workflow

Katarína Kostková

- zaintegrovať zmeny od Tomáša Jelínka do GUI Sprievodcu

Zápisnica zo stretnutia č.13

Dátum konania: 10.3.2008
Miesto konania: FIIT STU BA
Moderátor: Lucia Jastrzembská
Zápis vypracoval: Tomáš Konečný

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Na stretnutí sme preberali plnenie úloh za minulý týždeň ako aj diskutovali ako riešiť niektoré problémy.

Priebeh stretnutia

1. Kontrola úloh z minulého týždňa

Kto	**Úloha**	**Stav**
KK	Zaintegrovať okno sprievodcu	Splnená
TK	Načítavanie modulov	Predĺžená
ĽO	Editácia grafu, klikací sprievodca	Splnená
LJ	Buildenie pomocou Ant-u	Splnená
TJ	Doplnenie anotácií	Splnená

2. V súvislosti s danými úlohami sme preberali nasledujúce témy
   - TJ porozprával o novom layoute
   - LJ zreferovala o práci s Ant-om, pre lepšie použitie je potrebný plugin Ant for Eclipse, taktiež dospela k názoru, že používanie JUNIT-u je pohodlnejšie priamo a nie cez Ant.
   - ĽO hovoril o potrebe úpravy konvertoru medzi JUNG a PREFUSE, keďže v súčasnej podobe sa nekonvertujú vlastnosti vrcholov.

3. Prebrali sme problematiku, či môže byť vykresľovanie ako samostatný modul – pričom k tomuto problému sme neprijali jednoznačný záver a budeme sa k nemu musieť ešte vrátiť.
4. Ďalej sme riešili problematiku dvoch sprievodcov, zhodli sme sa na fakte, že by funkcie jadra mali byť volané iba z jedného miesta a taktiež problém komu ostane referencia na graf po skončení workflow. Výsledkom diskusie je, že vznikne spoločný predchodca – Abstraktná trieda sprievodcu, ktorý bude zapúzdrovať spoločné funkcie.
5. Problematiku checkpointov budeme riešiť serializáciou do súboru cez output modul.
6. Rozdelenie úloh.

   - **Katarína Kostková, Ľuboš Omelina** -vytvorenie Abstraktného sprievodcu a následne upravenie nadväzujúcich sprievodcov.
   - **Lucia Jastrzembská** – pozrieť sa na možnosť spájania grafov, preskúmať ďalšie možné moduly na rozšírenia z voľne dostupných knižníc.
   - **Tomáš Jelínek** – Nastavenie hodnôt do panelu.
   - **Tomáš Konečny** – finalizácia načítavania modulov.

Zápisnica zo stretnutia č.14

Dátum konania: 17.03.2008
Miesto konania: FIIT STU BA
Moderátor: Tomáš Konečný
Zápis vypracoval: Ľuboš Omelina

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Referovanie pokroku pri implementácii, konzultovanie aktuálneho stavu pri implementácii a ďalších krokov, rozdelenie ďalších prác na projkte.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	zistiť možnosť spájania grafov, preskúmať ďalšie možné moduly na rozšírenia z voľne dostupných knižníc.	Splnená
TJ	nastavenie hodnôt do panelu, aby bolo možné v používateľskom rozhraní zistiť už nastavené hodnoty.	Splnená
TK	finalizácia načítavania modulov.	Predĺžené (Integrácia na SVN)
ĽO, KK	vytvorenie abstraktného sprievodcu a následné upravenie nadväzujúcich sprievodcov.	Splnená

2. Žiadny z dostupných programov nedokáže spojiť dva grafy vytvorené pomocou knižnice JUNG. Na to aby sme mohli spájať rôzne grafy je potrebné vytvoriť

potrebný „merger“. Jedna z možností je prerobenie existujúcich, prerobenie je však nutné, pretože existujúce kopírujú iba štruktúru a nie atribúty (tobôž :o) nie názvy vrcholov).

3. Vytvorením abstraktného wizarda sa otvorila možnosť konverzie postupu spracovania (workflow) medzi wizardami. Zhodli sme sa na tom, že budeme túto konverziu podporovať iba jedným smerom a to z tabového wizarda do grafického wizarda. Opačná konverzia by spôsobila viacero problémov, preto ju podporovať nebudeme.
4. Sťahovanie hodnôt do panelu už funguje a grafická aplikácia aplikácia už vytvára workflow a používa knižnicu Mitandao.
5. Na stretnutí sme pomohli TK s riešením problému s interfejsmi.
6. Porovnali sme aktuálne splnené požiadavky zo špecifikácie, aby sme mali prehľad o stave projektu a zistili, ktorým častiam sa ešte musíme venovať

Rozdelenie úloh

Lucia Jastrzembská

- vytvorí nástroje na prácu s grafom knižnice JUNG. Medzi dôležité veci patrí zistenie a vytvorenie nástroja, ktorý bude vedieť manipulovať s atribútmi daného grafu

Tomáš Jelínek

- zovšeobecnenie FileChoosera. Všeobecný FileChooser umožní jeho používanie vo viacerých akciách (napr. open, save, ...)

Tomáš Konečný

- dokončenie dynamického načítavania modulov, vytvorenie modulu na načítanie dát z relačnej SQL databázy.

Katarína Kostková

- vytvorí modul checkpoint na to, aby bolo možné priebežne ukladať analyzované siete. Na tento modul použije serializáciu alebo pajek formát. Prirobí check-box do tabového wizarda, ktorým si používateľ zvolí či chce spracovávaný graf zobraziť.

Ľuboš Omelina

- vytvorí menu na editáciu hrán v grafe, zobrazenie názvov vrcholov, tooltipov a prípadne ďalšiu vizuálnu úpravu grafu,

všetci

- udržiavanie projektového denníka.

Zápisnica zo stretnutia č.15

Dátum konania: 01.04.2008
Miesto konania: FIIT STU BA
Moderátor: Ľuboš Omelina
Zápis vypracoval: Tomáš Jelínek

Zúčastnení

Členovia tímu	Tomáš Konečný Katarína Kostková Ľuboš Omelina Lucia Jastrzembská Tomáš Jelínek
Vedúci pedagóg	Michal Barla
Hostia	

Téma stretnutia

Témou stretnutia bolo diskutovanie spôsobu používania labelerov a možnosti vytvárania kontrolných bodov aplikácie (checkpointov).

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
KK	- vytvorenie modulu checkpoint. Vytvoriť checkbox do tabového wizarda	Čiastočne splnené
TK	- dokončenie dynamického načítania modulov, vytvorenie modulu na načítanie dát.	Čiastočne splnené / Predĺžená
ĽO	- vytvorenie menu na grafickú editáciu hrán	Splnené
TJ	- Zovšeobecnenie FileChoosera.	Splnené
LJ	- vytvorenie nástrojov na prácu s knižnicou Jung, hlavne nástroja, ktorý bude vedieť manipulovať s atribútami daného grafu	Splnené

2. Na začiatku stretnutia sme diskutovali problém s checkpointami. Problém je v tom, že knižnica Jung nepodporuje serializovanie grafu. Diskutovali sme nasledujúce možnosti:
    1. doprogramovať podporu priamo do knižnice
    2. dediť všetky potrebné triedy a doprogramovať funkčnosť do nich
    3. využiť ako checkpoint zápis do súboru

    Zvolili sme tretiu z uvedených možností
3. Diskutovali sme problémi, ktoré riešila Lucia Jastrzembská pri práci s knižnicou Jung. Oboznámila nás s riešením ako aj s labelermi, ktoré doprogramovala a vysvetlila, ako sa s nimi pracuje.
4. Tomáš Jelínek vysvetlil akým spôsobom zovšeobecnil FileChooser a akým spôsobom sa teraz používa ako aj možnosť, ako dorobiť ďalšie špeciálne parametre. Michal Barla upozornil na možnosť lepšej implementácie istej časti kódu.
5. Zadelili sa úlohy na ďalší týždeň

    Tomáš Konečný

    - hľadanie najlepšej možnej alternatívy, ako by mohol fungovať modul na načítavanie databáze

    Lucia Jastrzembská

    - vytvoriť pravé menu, ktoré bude podporovať zmenu farby vrcholov

    Tomáš Jelínek

    - vytvoriť pravé menu, ktoré bude podporovať vyberanie vrcholov ručne, podľa nejakej vlastnosti, do hĺbky N od daného vrcholu a vrcholy s počtom hrán N

    Ľuboš Omelina

    - vytvoriť nelineárny workflow v grafickom wizardovi

    Katarína Kostková

    - vytvoriť podporu na analýzu a filtrovanie grafu z horného menu

Zápisnica zo stretnutia č.16

Dátum konania: 07.04.2008
Miesto konania: FIIT STU BA
Moderátor: Tomáš Jelínek
Zápis vypracoval: Tomáš Konečný

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Na stretnutí sme preberali plnenie úloh za minulý týždeň, zosumarizovali sme si objavené chyby v aplikácií ako aj diskutovali ako riešiť niektoré ďalšie problémy pri vývoji. Zhodnotili sme, ktoré UC ešte nemáme pokryté a koľko času bude treba venovať dokumentácií.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	Vytvoriť pravé menu, ktoré bude podporovať zmenu farby vrcholov	Splnená
TJ	Vytvoriť pravé menu, ktoré bude podporovať vyberanie vrcholov ručne, podľa nejakej vlastnosti, do hĺbky N od daného vrcholu a vrcholy s počtom hrán N	Splnená
TK	Hľadanie najlepšej možnej alternatívy, ako by mohol fungovať modul na načítavanie databáze	čiastočne
KK	Vytvoriť podporu na analýzu a filtrovanie grafu z horného menu	Splnená

LO	Rozvetvenie pre nelineárny workflow.	Splnená

2. Zistili sme nejaké chyby v aplikácií. Konkrétne Classloader sa nekorektne choval pod Linuxom kedy zle orezal cestu k triedam. K tomu problému sa TK vráti po odovzdaní dokumentácie. Michal pošle návrh úpravy ako riešili podobný problém v inom projekte pomocou konfiguračného .propreties súboru.
3. Chyba sa našla aj pri Kartovom sprievodcovi, pokiaľ sa nešlo postupne ale po jednotlivých kartách sa preskakovalo. Treba upraviť tvorbu workflow pre tento prípad.
4. Hovorili sme o implementácií filtra, základne typu sme stanovili na („neni“, iba izolanty, n-počet hrán). Prípadne spojením viacerých podmienok pomocou and, or.

Rozdelenie úloh

Lucia Jastrzembská

Zápis aktuálnej konfigurácie zobrazenia, ktorá sa uchová pre ďalšie spustenie aplikácie.

Tomáš Jelínek

V pravom bočnom paneli zobraziť zoznam atribútov vrcholu.

Tomáš Konečný

Viac kolový komunikačný panel pre načítanie z databázy.

Katarína Kostková

Filtračný modul.

Ľuboš Omelina

Zmena veľkosti obrázkov.

Zápisnica zo stretnutia č.17

Dátum konania: 14.04.2008
Miesto konania: FIIT STU BA
Moderátor: Tomáš Konečný
Zápis vypracoval: Ľuboš Omelina

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Referovanie pokroku pri implementácii, konzultovanie aktuálneho stavu pri implementácii a ďalších krokov. Rozdelenie ďalších prác na projkte a hodnotenie už vytvorených častí. V rámci stretnutia sme analyzovali zhodu projektu so špecifikáciou.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	Implementácia reprezentácie atribútov vrcholov v grafe (v nich sa ukladajú výsledky analýz). Ukladanie globálnych nastavení.	Splnená
TJ	Implementácia zobrazenia výsledkov analýz v bočnom panely aplikácie (iba zobrazenie výsledkov pre vrcholy).	Splnená
TK	Pokračovanie v implementácii modulu pre načítanie dát z databázy	Predĺžené
KK	implementácia modulu pre filtrovanie grafu na základe stupňa uzla a modulu pre výpočet stupňa uzla.	Splnená

LO	Upravenie grafického sprievodcu (signalizácia nevyplnených modulov), ukladanie a načítanie nastavení modulov	Splnená

2. Pri zobrazení vlastností (výsledkov analýz) jednotlivých vrcholov vznikol problém, že ak používateľ vykoná analýzu viac krát, alebo jedna analýza bude mať viacero výsledkov tak nie je možné rozlíšiť ich názvy v zobrazení. Predbežné riešenie sme navrhli tým spôsobom, že tento problém si ošetrí tvorca príslušného modulu, aby sa jeden názov daného poľa neopakoval
3. Na stretnutí sme diskutovali o spôsobe a rozsahu dokumentovania projektu. V najbližšom čase bude potrebné vytvoriť záverečný dokument, ktorý bude obsahovať dopracovanú časť riadenie. Ďalej je potrebné vytvoriť príručku pre ľudí, ktorý budú pracovať s výsledkami našej práce. Dohodli sme sa, že okrem záverečného dokumentu o projekte vytvoríme nasledovné dokumenty:

   - Používateľská príručka (eng)

     (popísané elementárne vlastnosti produktov a funkcionalita grafickej aplikácie)
   - Príručka používateľa knižnice Mitandao (eng)

     (písané formou tutoriálu)
   - Príručka programátora (v zmysle rozširovateľa projektu) (eng)

     (popísaná rozmiestnenie modulov a zdrojových kódov projektu)
   - Technická dokumentácia (sk)

     (popísaná architektúra systému a vybrané (zaujímavé) časti zdrojových kódov)

   Vybrané dokumenty budú písané v anglickom jazyku a to najmä preto, aby sme tým umožnili používanie našej aplikácie, knižnice a ich prípadnému rozširovaniu aj ľuďmi z celého sveta.

Rozdelenie úloh

Lucia Jastrzembská

- vyrieši problém s atribútami jednotlivých analýz a dokončí reprezentáciu atribútov jednotlivých vrcholov v grafe. Začne písať príručku pre programátora v anglickom jazyku, v ktorej opíše najmä spôsob použitia vytvorenej knižnice Mitandao v používateľom definovaných aplikáciách.

Tomáš Jelínek

- vytvorí zobrazenie vlastností (výsledkov analýz) aj pre hrany v grafe podobným spôsobom ako sa zobrazujú vlastnosti vrcholov. Aj pre hrany a aj pre vrcholy doplní možnosť zmazania, skrytia vybranej skupiny entít

(entita = vrchol OR hrana). Opis práce s anotáciami, písanie technickej dokumentácie (písanie dokumentácia zároveň zostáva až do odovzdania projektu)

Tomáš Konečný

- dokončenie modulu pre načítanie údajov z databázy. Tento modul dokončí (ako aj sám prisľúbil) do 16.04. a niekedy vo večerných hodinách to uloží na SVN. Zvyšok času do najbližšieho stretnutia sa bude venovať opisu jeho vytvorených častí (classloader)v anglickom jazyku. Tento popis sa zaintegruje do príručky programátora (technickej dokumentácie).

Katarína Kostková

- upraví wizarda tým spôsobom, že bude načítavať graf z WorkPanel-u a nie z knižnice Mitandao. Odstráni z používateľského rozhrania položky, ktoré nemajú funkcionalitu. Doplní osnovu nových častí dokumentácie projektu (časti implementácia a produkt). Založí nasledovné dokumenty:
  - Používateľská príručka (eng)
  - Príručka používateľa knižnice Mitandao (eng)
  - Príručka programátora (v zmysle rozširovateľa projektu) (eng)
  - Technická dokumentácia (sk)

Ľuboš Omelina

- implementácia ukladanie celého definovaného procesu analýzy, pričom sa budú ukladať aj všetky nakonfigurované moduly do jedného súboru (samozrejme aj opätovné načítanie definovaného procesu analýzy). Opravenie chýb v grafickom sprievodcovi. Začiatok písania používateľskej príručky v anglickom jazyku.

všetci

- každý z členov tímu sa bude snažiť čo najviac prispieť do dokumentácie a príručiek v súlade so svojím **vedomím** a **svedomím** (a svojou prácou na projekte), takým spôsobom, aby sme boli schopný odovzdať projekt kompletne zdokumentovaný a zároveň si nemuseli upierať spánok posledné dni pred odovzdaním. V prípade že toto nebude možné tak sa bude snažiť o zmiernenie týchto dôsledkov. Ďakujem.

Zápisnica zo stretnutia č.18

Dátum konania: 21.04.2008
Miesto konania: FIIT STU BA
Moderátor: Ľuboš Omelina
Zápis vypracoval: Katarína Kostková

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Prevažnú časť stretnutia sme sa venovali kontrole aktuálneho stavu projektu, produktu a dokumentácie. Na základe zistení sme identifikovali úlohy na posledný týždeň pre prvým odovzdaním ako aj úlohy, ktoré budeme riešiť do druhého odovzdania.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	– príručka používateľa knižnice – zmeny nastavení hrán	Splnená
TJ	– atribúty hrán – príručka pre programátora	Splnená
TK	– modul na načítanie z databázy	Predĺžené
KK	– vytvorenie dokumentov pre príručky – doplnenie dokumentácie – úpravy vo wizardovi – používateľská príručka	Splnená
LO	– uloženie konfigurácie – používateľská príručka	Splnená

2. Pozreli sme si používateľskú dokumentáciu pre používateľa knižnice Mitandao a príručku pre používateľa (aplikácie). Dohodli sme sa, že do oboch doplníme spoločný úvod, ktorý bude hovoriť o tom, čo to je Mitandao.
3. TK predviedol ako sa posunul pri vytváraní modulu na načítavanie z databázy, vyskytla sa však chyba, ktorú má TK za úlohu čo najskôr odstrániť.
4. Pedagogický vedúci si pozrel a otestoval celú aplikáciu, čím nám pomohol odhaliť drobné nedostatky, resp. navrhol malé vylepšenia. Niektoré z nich stihneme spracovať už teraz, niektoré si nechávame na čas po prvom odovzdaní.

Rozdelenie úloh

Lucia Jastrzembská

- doplniť úvod a slovník pojmov v príručke pre používateľa knižnice
- dodať svoju časť k dokumentácii produktu

Tomáš Jelínek

- z results odstrániť source a target
- v edge tabe vypísať edge, nie node
- zdokumentovať anotácie, slovensky

Tomáš Konečný

- odstránenie chýb v module na načítavanie z databázy
- passwd písať ako hviezdičky
- zdokumentovať classloader, anglicky a slovensky

Katarína Kostková

- do main window dodať reagovanie na klávesové skratky
- zjednocovať dokumentáciu

Ľuboš Omelina

- zmeniť infinity zobrazovanie (aby sa graf nehýbal)
- na properties vysvietiť results tab
- ošetriť nulovosť modulov
- linkovanie

všetci

- **dokumentovať a upraviť zdrojové kódy** podľa stanovených konvencií

Zápisnica zo stretnutia č.19

Dátum konania: 28.04.2008
Miesto konania: FIIT STU BA
Moderátor: Katarína Kostková
Zápis vypracoval: Tomáš Konečný

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Hlavnou témou boli prípravy na prvé odovzdanie produktu 30.4.2008. Aká má byť úprava dokumentácie a forma odovzdaného produktu. Diskutovalo sa najmä rozdelenie opisu produktu medzi technickú dokumentáciu a kapitolu 7 -Produkt. Zároveň M. Barla objavil pár chýb, resp. funkčností, ktoré sa mu nepozdávali. Navrhoval doplnenie tooltipov, modálne okná na ošetrenie výnimiek. Objavené defekty označenie a odznačenie vrcholu, nevyplnenie všetkých parametrov filtra.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	- dodať svoju časť k dokumentácií produktu	Splnená
TJ	- z results odstrániť source a target - v edge tabe vypísať edge, nie node - zdokumentovať anotácie, slovensky	Splnená
TK	- odstránenie chýb v module na načítavanie z databázy - passwd písať ako hviezdičky - zdokumentovať classloader, anglicky a slovensky	Zdokumentované, nič neimplantoval, *** na stretnutí
KK	- do wizarda dodať reagovanie na klávesové skratky	Splnená

	- zjednocovať dokumentáciu	
LO	- zmeniť infinity zobrazovanie (aby sa graf nehýbal) - na properties vysvietiť results tab - ošetriť nulovosť modulov - linkovanie	Splnená

2. Diskusia o správnom rozčlení dokumentácie medzi kapitolu 7 – produkt, Technickú dokumentáciu a prílohy.
3. Čo by malo byť obsahom CD. MB by bol rád keby bol obsah formou statickej web stránky s linkami, ale stačí mu aj readme.txt.
4. MB objavil vrámci pravidelného testovanie isté chyby/vlastnosti/funkčnosti aplikácie, u ktorých by ocenil zmenu.
    - Označenie a následne odznačenie vrcholu javí známky nestability.
    - Pri niektorých nastaveniach sprievodcu by považoval za prínos tooltip nápovedu.
    - MB by ocenil klávesové skratky nie len v hlavnom menu ale aj podmenu.
    - Aplikácia sa nechová úplne správne pri nevyplnení všetkých parametrov filtra.

Rozdelenie úloh

Všetci

Úprava svojej časti v dokumentácií a prílohách, každý podľa svojej časti implementácie. Najme presunúť príliš podrobné opisy z kap. 7 do Technickej časti dokumentácie a pridať HL opis do kap7 prípadne screanshoty.

Katarína Kostková

Spojenie a úprava dokumentácie.

Ľuboš Omelina

Príprava CD

Tomáš Konečný

Zbuildenie aplikácie

Zápisnica zo stretnutia č.20

Dátum konania: 05.05.2008
Miesto konania: FIIT STU BA
Moderátor: Tomáš Konečný
Zápis vypracoval: Lucia Jastrzembská

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Hlavnou témou stretnutia bolo zhodnotenie vecí, ktoré sa odovzdali minulý týždeň a načrtnutie plánu na ďalšie obdobie.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
Všetci	- úprava svojej časti v dokumentácií a príloh, každý podľa svojej časti implementácie. - presunúť podrobné opisy z kap. 7 do Technickej časti dokumentácie - pridať HL opis do kap. 7 prípadne screenshoty.	Splnená
TK	- Zbuildenie aplikácie	Splnená, vykonané LJ, LO, TJ
KK	- Spojenie a úprava dokumentácie.	Splnená
LO	- Príprava CD	Splnená

2. Na začiatku stretnutia sme zhodnotili časti dokumentácie, ktorá bola odovzdaná predchádzajúci týždeň. Keďže sme sa počas týždňa dozvedeli, že nie je potrebné odovzdávať kapitolu Produkt ani technickú dokumentáciu, rozhodli sme tieto kapitoly zatiaľ nechať otvorené a priebežne ich dopĺňať. Pri kontrolovaní obsahu zip súboru (ktorý mal obsahovať obsah CD) sa prišlo na to, že obsahuje zbytočné duplicity, ktoré bude treba odstrániť.
3. Keďže pri buildení aplikácie sa vyskytli problémy s načítavaním modulov, dohodli sme sa, že bude nutné prepracovať triedu MitandaoClassLoader tak, aby fungovala aj vo vývojovom prostredí, aj v hotovej aplikácii.
4. Ďalším bodom stretnutia bola diskusia o testovaní našej aplikácie. Dohodli sme sa, že budeme testovať aplikáciu cez GUI a takisto aj použitie knižnice. KK a LO sa podujali na získanie externých testerov. Zároveň sme sa dohodli, že by bolo vhodné vykonať aj crash testy. Všetky chyby sa budú zadávať do systému Trac, prístup okrem testerov umožníme i konkurenčnému tímu.
5. KK prišla počas týždňa na to, že keď je zobrazený v aplikácii graf a dá sa naň aplikovať analýza alebo nové načítanie grafu a pritom sa vyskytne výnimka, pôvodný graf sa stratí. Keďže by to mohlo byť pre používateľa nepríjemné, dohodli sme sa, že toto správanie zmeníme.
6. Ďalej sme sa dohodli, kto bude testovať produkt konkurenčného tímu (TK, TJ, KK)
7. Nakoniec sme zhrnuli veci, ktorých implementáciu sme odložili až po odovzdaní Používateľských príručiek – vytvorenie checkpointov, vytvorenie ďalších modulov, zobrazovanie wizardov na základe klávesovej skratky.
8. Zadelili sa úlohy, termín ich vyriešenia bol definovaný do konca semestra.

Lucia Jastrzembská

- práca na novom načítavaní modulov
- zistiť, kde bude hostovaný program (code.google.com alebo sourceforge.net)
- testovanie produktu konkurenčného tímu, napísanie posudku
- spraviť revíziu kódu, napísať výhrady ku kvalite jednotlivým členom tímu

Tomáš Jelínek

- práca na novom načítavaní modulov
- testovanie produktu konkurenčného tímu, napísanie posudku

Katarína Kostková

- osloviť externých testerov
- testovanie produktu konkurenčného tímu, napísanie posudku
- vytvorenie klávesových skratiek na spúšťanie wizardov
- úprava zobrazovania grafov pri výskyte chyby
- vytvorenie nových modulov algoritmov

Tomáš Konečný

- dokončenie modulu pre načítavanie dát z DB

Ľuboš Omelina

- checkpointy
- rozbehanie TRAC-u
- osloviť externých testerovvytvoriť leták

Zápisnica zo stretnutia č.21

Dátum konania: 12.05.2008
Miesto konania: FIIT STU BA
Moderátor: Lucia Jastrzembská
Zápis vypracoval: Tomáš Jelínek

Zúčastnení

Členovia tímu	Lucia Jastrzembská Tomáš Jelínek Tomáš Konečný Katarína Kostková Ľuboš Omelina
Vedúci pedagóg	Michal Barla
Hostia	---

Téma stretnutia

Na stretnutí sme sa venovali vymedzovaniu tých vecí, ktoré je ešte treba dorobiť do finálnej verzie programu a dokumentácie.

Priebeh stretnutia

1. Skontrolovali sme úlohy z minulého týždňa:

Kto	Úloha	Stav
LJ	- Práca na novom načítavaní modulov - Vybratie hostovacieho servera - Testovanie produktu konkurenčného tímu - Revízia kódu	Čiastočne splnené / PredĺženáPredĺžená
TJ	- Práca na novom načítavaní modulov - Testovanie produktu konkurenčného tímu	Čiastočne splnené / PredĺženáPredĺžená
TK	- dokončenie modulu na načítanie dát z DB testovanie produktu konkurenčného tímu	Čiastočne splnené / PredĺženáPredĺžená

KK	- Oslovenie testerov - testovanie produktu konkurenčného tímu - vytvorenie klávesových skratiek na púšťanie wizardov - úprava zobrazenia grafov pri výskyte chyby - vytvorenie nových modulov algoritmov	Čiastočne splnené / predĺžené
LO	- checkpointy - rozbehanie tracku - oslovenie externých testerov	Splnená

2. MB sa rozhodol skúsiť pomerne rozsiahlu databázu na aplikácii Mitandao. Narazili sme ale na problém, že modul na načítavanie dát z DB nepokrýval ten spôsob ukladania dát, ktorý bol v DB
3. TK pri riešení DB modulu narazil na obmedzenia Mitandao UI Frameworku, ktoré neumožňovali prenášanie dát medzi modulom a jeho panelom keď sú dáta v poly.
4. KK prišla s myšlienkou mať aktývny work panel už pri spustení aplikácie
5. LJ a TJ povedali, čo spravili s novým classloadrom. Narazili na problém, odkiaľ by sa mali počítať relatívne cesty. Záverom bolo, že sa majú počítať od koreňového adresára aplikácie.
6. Zadelili sme si úlohy:

   Lucia Jastrzembská

   - predĺžené úlohy

   Tomáš Jelínek

   - predĺžené úlohy

   Katarína Kostková

   - predĺžená

   Tomáš Konečný

   - pokryť v DB moduly aj možnosť, ktorá je v DB

   Ľuboš Omelina

   - vytvoriť leták
   - zrealizovať myšlienku popísanú v bode 4

Posudky