UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Evidenčné číslo : f3f30ad4-f8c7-4f99-8b99-5dd50407b134

EDITOR DIAGRAMOV PRE VZDIALENÝCH POUŽÍVATEĽOV

2011

Dagmar Havelková

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

EDITOR DIAGRAMOV PRE VZDIALENÝCH POUŽÍVATEĽOV

Bakalárska práca

Študijný program: Aplikovaná informatika

Študijný odbor: 9.2.9 Aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky, FMFI, UK

Školiteľ: Mgr. Pavel Petrovič, PhD.

Bratislava 2011

Dagmar Havelková

Čestné prehlásenie

Čestne prehlasujem, že túto bakalársku prácu som vypracoval samostatne s použitím uvedených zdrojov.

V Bratislave dna 03.6.2011 .....

Dagmar Havelková

Poďakovanie

Moje poďakovanie patrí predovšetkým školiteľovi Mgr. Pavlovi Petrovičovi, PhD. za ochotu, konštruktívne pripomienky a vedenie tejto práce.

Dagmar Havelková

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Dagmar Havelková
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Editor diagramov pre vzdialených používateľov

Cieľ: Navrhnúť a implementovať editor UML diagromov pre vzdialených používateľov, ktorí sú pripojení na Internet a súčasne pracujú na vývoji zdieľaného diagramu. Okrem editovania diagramu sa môžu dohovárať v textovom okne a posielať si súbory.

Literatúra: M.Cade, H. Shiel: Sun Certfied Enterprise Architect for Java EE Study Guide, Prentice Hall, 2010.

Vedúci: Mgr. Pavel Petrovič, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 12.10.2010

Dátum schválenia: 25.10.2010

doc. RNDr. Mária Markošová, PhD.
garant študijného programu

Dagmar Havelková
študent

Pavel Petrovič
vedúci

Abstrakt

Úloha tejto bakalárskej práce je navrhnúť a implementovať UML editor, ktorý bude umožňovať vývoj diagramov vzdialeným používateľom. Na trhu je momentálne množstvo UML editorov, tie však neposkytujú možnosti pre interaktívne vytváranie diagramov vzdialeným používateľom. Práca obsahuje stručný úvod do jazyka UML, analýzu existujúcich nástrojov, návrh aplikácie a informácie o implementácii.

Abstract

The goal of this thesis is to design and implement online UML editor, where distant users can cooperate during the process of making diagrams. In present there are plenty of UML editors on the market but they lack the feature of online cooperation. Thesis contains brief introduction to UML, analysis of existing UML editors, design of the application and implementation details.

Obsah

1 Úvod

„Niektorí ľudia začnú hovoriť o chvíľku skôr, než začnú myslieť.“
Jean De La Bruyere

Kým človek začne prednášať pred publikom pripraví si prejav. Ten tvorí postupne na základe logickej osnovy. Je k tomu vedený od základnej školy. Jeden z prvých poznatkov, ktoré sa na prvej hodine slohu dozvie je, že práca má tri časti. Prvá je úvod, druhá obsah a tretia záver. Každá má obsahovať určité špecifické informácie.

Pri vytváraní systému je to podobné. Programátor hovorí prostredníctvom programovacieho jazyka s počítačom a ten prejav posúva koncovému užívateľovi. Čím je systém rozsiahlejší tým viac potrebuje osnovu – návrh. V návrhu sa pretvoria požiadavky zákazníka podané v bežnej ľudskej reči do špeciálneho jazyka, ktorý tvorí osnovu pre programátora. Tento špeciálny jazyk sa nazýva UML. Je to komplexný grafický jazyk vytvorený pre potreby komplexného návrhu systému, nezávisle od programovacích jazykov a platforiem. Umožňuje zaznamenávať všetko od požiadaviek zákazníka cez správanie sa systému v rôznych situáciách až po inštaláciu systému.

1.1 Motivácia a ciele práce

Moja motivácia pri tejto práci je veľmi sebecká. Z praxe viem ako funguje práca na veľkých projektoch, na ktorých pracujú desiatky ľudí. Ani stovka dobrých programátorov nevytvorí komplexný systém bez podrobného návrhu. Ich nezávislá práca v paralelnom čase je bez návrhu teoreticky aj prakticky nemožná bez ohľadu na kvalitu ich výkonu.

Čím je projekt väčší, tým viac očakávaných a neočakávaných problémov pri jeho vývoji vzniká. S týmito problémami sa človek pracujúci na projekte stretáva denne a musí ich riešiť. Veľa krát sú to problémy zavinené nedostatočným návrhom. Nezrovnalosti medzi komunikáciu rôznych externých systémov, problémy pri spolupráci rôznych modulov systému a podobne. Riešenie takýchto problémov je veľmi zdĺhavé, pretože je závislé na ďalších ľuďoch, ktorý majú nedostatok času a množstvo svojich problémov. Toto je jeden s reálnych následkov zlého návrhu systému. Ľudia, ktorí nestíhajú svoju prácu a paradoxne čakajú. Čakajú na riešenia iných ľudí. Hodiny.

Preto, aby sme sa takýmto problémom vyhli je potrebná kvalitná práca človeka, ktorý systém navrhuje a tiež správne pochopenie problematiky programátorom. Nástroj Online UML Editor má za úlohu dopomôcť študentom škôl zameraných na výučbu informatiky pri učení sa jazyka UML.

Ciel práce je vytvoriť

- jednoduchý UML editor s intuitívnym ovládaním, vhodný pre študentov, ktorý nemajú skúsenosti s prácou v jazyku UML
- umožniť študentom efektívne pracovať na vývoji diagramov zo vzdialených miest v reálnom čase
- sprostredkovať komunikáciu vzdialených študentov

1.2 Štruktúra práce

Tento dokument sa logický člení na 3 časti : prípravou na vývoj, vývojom a budúcnosťou systému.

Príprava na vývoj systému je popísaná v druhej a tretej kapitole. Druhá kapitola s názvom Unified Modeling Language obsahuje teoretický úvod k jazyku UML. Popisuje históriu jazyka, druhy diagramov, ktoré tento grafický jazyk tvoria a obsahuje podrobný popis základných stavebných prvkov diagramov. Doplnená je podkapitola s názvom Rozšírenia UML, v ktorej sú popísané mechanizmy používane na rozšírenie jazyka UML pre špeciálne prípady, ktoré jadro jazyka nepokrýva.

Tretia kapitola Analýza existujúcich nástrojov pre vývoj UML obsahuje prakticky nadobudnuté poznatky z práce s UML editormi. Sú rozdelené podľa prístupnosti na voľne prístupné nástroje a komerčné nástroje. Pridaná je aj podkapitola obsahujúca popis a zhodnotenie práce s prídavnými modulmi do prostredí NetBeans a Eclipse. Koniec tvorí zhrnutie a porovnanie spomínaných editorov.

Vývoj systému je popísaný v kapitolách Požiadavky na systém, Návrh systému, Implementácia a Nasadenie systému.

V kapitole Požiadavky na systém je súhrn požiadaviek, ktoré má systém spĺňať. Sú v nej popísaný cieľový užívatelia systému, moduly aplikácie a obsahuje náčrt grafického návrh užívateľského rozhrania.

Kapitola Návrh systému obsahuje technický pohľad na systém. Na jej začiatku je popis použitých technológií. Ďalej popisuje modul aplikácie Prihlasovanie a návrh databázy, ktorú tento modul využíva. Podkapitola Trieda Diagram zahŕňa podrobný popis návrhu realizácie diagramu prostredníctvom triedy Diagram. Nadväzuje na ňu kapitola Komponenty s popisom komponentov, z ktorých sa diagram skladá. Nasledujúce podkapitoly Prenášanie objektov prostredníctvom server-socked technológie a Prenos správ a udalostí sa zaoberajú sieťovou komunikáciou . Ďalšia časť Spracovanie udalostí obsahuje popis akcií, ktoré sa vykonajú po prijatí informácie o udalosti od vzdialeného užívateľa. Popis zosúladenia diagramov užívateľov je v podkapitole Synchronizácia. Posledná podkapitola Ukladanie a export diagramu v rôznych formátoch obsahuje zoznam a popis štandardných výstupov UML editorov, ktoré bude podporovať systém Online UML Editor a popis špeciálneho typu výstupu, ktorý uchováva informácie špecifické pre navrhovaný systém .

Kapitola Implementácia obsahuje informácie o programovaní systému. Zameriava sa na zložitejšie problémy, ktoré sa vyskytli pri implementácií a na zvolený spôsob riešenia.

V kapitole Nasadenie a použitie je návod ako vytvoriť databázu užívateľov, ktorú aplikácia využíva, informácie o spustení a nastavení programu a stručné informácie o tom ako používať aplikáciu.

Budúcnosť systému popisuje kapitola Možné rozšírenia. Obsahuje nápady ako systém vylepšiť.

2 Unified Modeling Language

Pri vytváraní každého rozsiahleho systému, je nevyhnutnou súčasťou vývoja návrh systému. Návrh má zachytiť kompletnú štruktúru systému ešte pred začatím implementácie. Je podkladom pre implementáciu. Jazyk UML je súhrnom pravidiel, ktoré umožňujú vytvoriť grafickú vizualizáciu kompletného návrhu objektovo orientovaného systému.

2.1 História

Vývoj softvéru v počiatkoch začínal a končil programovaním. Pri malých systémoch to bolo dostačujúce, ale ako sa časom softvérové možnosti rozširovali a veľkosť programov rástla, začala vznikať potreba návrhu systémov. V 60. rokoch Larry Constantine ako jeden z prvých priekopníkov skúmajúcich návrh systémov vyslovil myšlienku, že softvér má byť pred implementáciou navrhnutý. Vznikli modelovacie nástroje, ako napríklad SSADM.

Počítače boli postupom času čoraz viac využívané aj v iných odvetviach ako v matematike. Preto bolo potrebné zmeniť prístup a možnosti programovacích jazykov. Postupný vývoj nových technológií smeroval k objektovej orientácii jazykov. V roku 1966 vzniká programovací jazyk Simula. Tento jazyk je z hľadiska vývoja programovacích jazykov dôležitý, pretože v ňom Ole-Johan Dahl a Kristen Nygaard zaviedli ideu triedy. O niečo neskôr v roku 1970 bol na trh uvedený jazyk SmallTalk, ten obsahuje z hľadiska objektovej orientácie mnoho podstatných konceptov. Napríklad dedičnosť, či správy. Tento jazyk je vyvíjaný dodnes. V 70. rokoch pohla ďalej vývoj objektovo orientovaných jazykov aj Barbara Lisková. V jazyku CLU výrazne posunula dopredu model abstraktného dátového typu. V roku 1980 prišiel Martin Richardse s jazykom BPCL. Skrátene sa jazyk označoval ako B. Ďalší vývojový stupeň tohto jazyka bolo C-éčko, ktoré neskôr Bjarne Stroustrup modifikoval do objektovo orientovaného jazyka C++. Ten sa stal jedeným s najpoužívanejších objektovo orientovaných jazykov.

Vzhľadom na tieto podstatné zmeny v technológii programovania, bolo nutné prispôsobiť aj modelovacie metódy. Vzniklo viacero metód, ktoré umožňovali vytvoriť návrh systému, ktorý bol vyvíjaný v objektovo orientovanom jazyku. V 90. rokoch sa začal vývoj jazyka UML vo firme Rational Software. Jeden s cieľov pri vývoji tohto jazyka bolo zabezpečenie zjednodušenia a konsolidácie existujúcich metód na návrh objektovo orientovaného systému. Pracujú na ňom Grady Booch, Jim Rumbaugh a neskôr sa pridáva Ivar Jacobson. Ako najdôležitejšie ciele si stanovili :

- vytvorenie zrozumiteľného grafického modelovacieho jazyka
- zaistenie rozšírení jazyka pre špecifické prípady
- nezávislosť na konkrétnych programovacích jazykoch
- integrácia najlepších postupov

V novembri 1997 OMG (Object Management Group) prijíma UML 1.1 ako priemyselný štandard.

Mesiac/Rok	Verzia	Popis
1995	Unified Method 0.8	Booch a Rumbaugh vytvárajú Unified Method 0.8 zlúčením poznatkov z Boochovej metódy a OMT.
06/1996	UML 0.9	Pripojenie technológie OOSE s príchodom Ivara Jacobsona.
11/1997	UML 1.1	Verzia prijatá OMG (Object Management Group).
03/2000	UML 1.3	Zapracované aktualizácie, ktoré vznikli na podnet užívateľov.
09/2001	UML 1.4	
07/2004	UML 1.4.2	
03/2003	UML 1.5	
07/2005	UML 2.0	Vo verzii 2.0 sú pridané vnorené komponenty, vylepšenia diagramov správania a vylepšenie vzťahov medzi štrukturálnymi diagramami a diagramami správania.
08/2007	UML 2.1.1	
11/2007	UML 2.1.2	
02/2009	UML 2.2	
05/2010	UML 2.3	
03/2011	UML 2.4 – Beta 2	

2.2 Druhy diagramov a ich použitie

Vo verzii UML 2.0 delíme diagramy na 3 základné typy. Sú to štrukturálne diagramy, diagramy správania a diagramy interakcie.

Štrukturálne diagramy zobrazujú statické rozvrhnutie systému. Obsahujú popisy tried, objektov, rozhraní a vzťahy medzi týmito entitami.

- **Diagram tried** zobrazuje statický pohľad na dizajn systému. Obsahuje rozloženie tried, rozhraní a zachytáva vzťahy medzi nimi. Je to jeden z najvyužívanejších diagramov.
- **Diagram komponentov** zachytáva pohľad na systém z implementačného hľadiska. Pod pojmom komponent rozumieme napríklad implementáciu niekoľkých tried. Diagram zobrazuje skupinu takýchto komponentov a vzťahy medzi nimi.
- **Diagram nasadenia** zachytáva konfiguráciu hardvéru. Napríklad prepojenie zariadení ako sú počítače a servery, spôsob akým majú byť pospájané, ich nastavenia , prípadne popis softvéru, ktorý má na zariadeniach byť nainštalovaný pre bezchybný chod modelovaného systému.
- **Diagram balíčkov** zobrazuje balíčky a vzťahy medzi nimi. Balíček môže byť napríklad zoskupenie viacerých tried. Tento druh diagramu používa notáciu totožnú s diagramom tried.
- **Diagram objektov** zachytáva objekty a ich vzťahy v určitom čase. Objekt je inštancia triedy systému. Tento druh diagramu používa notáciu totožnú s diagramom tried.
- **Diagram zloženej štruktúry** je nový diagram definovaný vo verzii UML 2.0 . Zastrešuje komplexný pohľad na zložité komponent v ktorom sa združujú napríklad triedy a komponenty súčasne.

Diagramy správania zobrazujú dynamickú časť systému. Zachytávajú interakcie systému v rôznych situáciách.

- **Stavový diagram** zobrazuje postupnosť stavov objektov, prechody medzi stavmi a akcie súvisiace so zmenami stavu v systéme, do ktorých sa objekt môže dostať.
- **Diagram aktivít** popisuje postupnosť aktivít, ktoré systém vykonáva vzhľadom na určené podmienky. Vo verziách UML 1.X bol považovaný za určitý druh stavového diagramu, no od verzie 2.0 sa s týmto diagramom nespája. Sémantika diagramu sa zmenila a jej základ bol vytvorený podľa grafického modelovacieho jazyka Petriho sietí. Petriho siete sa používajú na znázornenie chodu procesov v dynamických systémoch.
- **Diagram prípadu použitia** zaznamenáva funkcionalitu systému z pohľadu užívateľa. Zobrazuje vzťahy medzi rôznymi typmi užívateľov a funkciami systému. Diagram zaznamenáva pohľad na systém z hľadiska návrhu.

Základy **diagramov interakcie** pochádzajú z diagramoch správania. V novej verzii UML 2.0 do tejto skupiny patria sekvenčný diagram, diagram komunikácie, diagram prehľadu interakcií a diagram časovania. Diagramy interakcie zachytávajú dynamický pohľad na systém. Zobrazujú sa v nich objekty a interakcie medzi nimi.

- **Sekvenčný diagram** zobrazuje časovo usporiadanú postupnosť procesov vykonávaných medzi objektmi.
- **Diagram komunikácie** je nový pojem od verzie UML 2.0 . V princípe to, ale nie je nový diagram. Je obmena, resp. zjednodušenie diagramu spolupráce z predošlých verzií UML. Zachytáva komunikáciu medzi objektmi. Štruktúra a povaha jednotlivých prvkov je zhodná s sekvenčným diagramom, takže v jednoduchších prípadoch je možné robiť konverziu medzi týmito diagramami.
- **Diagram prehľadu interakcií** sa skladá z väčšej miery z prvkov diagramu aktivít a sekvenčného diagramu. Zobrazuje referencie na sekvenčné diagramy, ktoré spája pomocou komponentov diagramu aktivít. Vytvára pohľad na systém, alebo jeho časť z vyššej úrovne.
- **Diagram časovania** zobrazuje stavy objektov vzhľadom na čas. Je dôležitou súčasťou návrhov Real-time systémov, kde je kladený dôraz na časovú presnosť. Pre tento diagram je možné používať dve notácie. Menej podrobná forma diagramu zobrazuje stavy objektov vzhľadom na relatívne časové jednotky. Podrobnejšia forma zobrazuje stavy objektov vzhľadom na absolútny čas.

2.3 Základné stavebné prvky

Každý diagram je poskladaný z presne definovaných prvkov. Tie môžu reprezentovať statické časti systému, vzťahy medzi nimi, správanie systému, prípadne elementy, ktoré v zložitejších diagramoch zoskupujú určité časti systému.

Štrukturálne elementy opisujú statické časti systému. Patria medzi ne elementy trieda, spolupráca, prípad použitia, rozhranie, komponent a poznámka.

- **Trieda** je element, ktorý reprezentuje vlastnosti určitej skupiny objektov. Je určená názvom a obsahuje zoznam atribútov, ktoré určujú vlastnosti odvodeného objektu a zoznam operácií, ktoré sa vykonávajú v súvislosti s objektom.
  Prvok trieda je graficky reprezentovaný ako obdĺžnik, rozdelený do troch častí. V prvej časti je názov triedy. Druhá obsahuje atribúty a tretia popis operácií triedy. Druhá a tretia časť sa nemusí zobraziť ak trieda nemá žiadne atribúty a operácie.
- **Rozhranie** je abstraktný popis operácií a podtried, ktoré sa implementujú na triedu. Jedna trieda môže implementovať naraz viacero rozhraní. Grafická reprezentácia rozhrania je

takmer totožná s reprezentáciou triedy rozdiel je len v tom, že element typu rozhranie obsahuje v prvej časti obdĺžnika kľúčové slovo „interface“.

- **Prípad použitia** je element, ktorý obsahuje názov a popis použitia systému užívateľom. Je reprezentovaný elipsou. V jej strede je vypísaný názov funkcionality. Mal by byť pokiaľ možno čo najvýstižnejší.
- **Komponent** reprezentuje reálnu implementáciu jednej, alebo viacerých tried. Zapuzdruje ich do jedného celku, ktorý pomocou rozhraní komunikuje s ostatnými časťami systému. Môže obsahovať atribúty a operácie. Vo verziách UML 1.X existoval pre elementy typu komponent špeciálny vzor. Od verzie 2.0 je reprezentovaný obdĺžnikom, ktorý je označený kľúčovým slovom „component“ , alebo ikonou v ľavom hornom rohu.

Zoskupovanie v diagramoch zabezpečuje komponent balíček.

- **Balíček** je element, ktorý zoskupuje časti systémov do logických celkov. Využíva sa napríklad na vyjadrenie súvislostí medzi podsystémami. Grafická reprezentácia je zoskupenie dvoch obdĺžnikov pripomínajúce zložku.

Behaviorálne elementy zachytávajú správanie systému. Patria sem dva prvky - interakcia a stav.

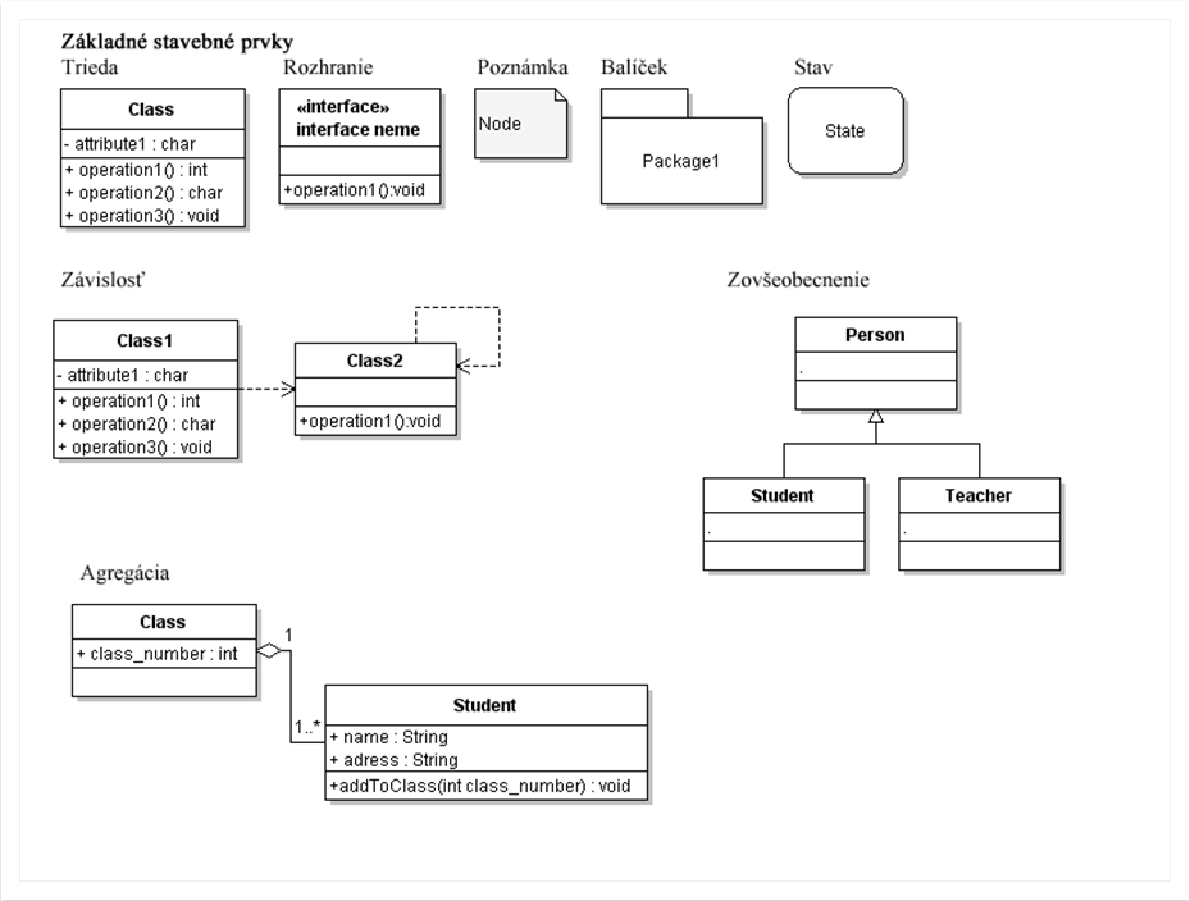
- **Interakcia** reprezentuje vzťah medzi objektami diagramu. Je zobrazené ako plná čiara so šípkou na konci.
- **Stav** je element zobrazujúci stav určitého objektu v závislosti od času, alebo sekvencie akcií vykonávaných nad objektom. Je reprezentovaný štvorcom s oblými hranami, v ktorom je text s popisom stavu.

Medzi anotácie patrí jediný element a tým je poznámka.

- **Poznámka** slúži na doplnenie informácií, ktoré popisujú určitú časť systému. Môže to byť element, alebo skupina elementov iného typu. Graficky sa zobrazuje ako obdĺžnik s preloženým pravým horným rohom. V obdĺžniku je zobrazený text poznámky.

Elementy zobrazujúce vzťahy sú dôležitou súčasťou notácie UML. Patria k nim elementy, ktoré vyjadrujú závislosť, zovšeobecnenie, zoskupenie (agregáciu) a realizáciu.

- **Závislosť** zobrazuje všeobecný vzťah medzi dvoma elementmi. Používa sa tam, kde ešte nie je presne špecifikované aký vzťah medzi elementmi je. Reprezentovaná je ako prerušovaná čiara ukončená šípkou.
- **Zovšeobecnenie** vyjadruje vzťahy typu rodič – dieťa. Používa sa napríklad na označenie vzťahu medzi rodičovskou triedou a jej potomkovou triedou, alebo medzi rozhraním a triedami, ktoré ho implementujú. Grafická reprezentácia je plná čiara s trojuholníkovou prázdnou šípkou na konci.
- **Zoskupenie (Agregácia)** zobrazuje vzťah medzi elementmi, kde platí, že jeden je časťou toho druhého. Zobrazuje sa ako plná čiara na konci ktorej je nevyplnený kosoštvorec. V koncových bodoch môže byť aj špeciálny popis upresňujúci vzťah elementov.



2.4 Rozšírenia UML

Očakávame, že jazyk UML bude upravovaný podľa objavujúcich sa nových potrieb a pre špecifické oblasti. Zároveň, ale nechceme, aby sa museli základné koncepcie jadra predefinovať, či opakovane implementovať v každej špeciálnej oblasti. Preto si myslíme, že by mechanizmus rozširovania mal podporovať odchýlky od všeobecného prípadu a nemal by vyžadovať

implementáciu základných konceptov OOA&D. Základné koncepty by sa nemali meniť nad rámec nevyhnutnosti. [2]

Toto bol jeden s cieľov, ktorý si dali autori jazyka UML predtým ako ho začali vyvíjať. Už vtedy mysleli na možnosť, že existujú špecifické prípady, ktoré jadro UML nepokrýva. Reálne sa rozšírenia používajú v prípade, že je potrebné napríklad :

- prispôsobiť terminológiu vzhľadom na použitú platformu
- vytvoriť prvky, ktoré reprezentujú špecifické a doteraz nedefinované konštrukcie
- prispôsobiť notácie špecifikovaných prvkov kvôli výstižnejšej reprezentácii
- doplniť sémantiku
- pridať neexistujúcu sémantiku
- pridať nové obmedzenia
- pridať doplňujúce informácie.

Existujú dva spôsoby ako si možno jazyk UML prispôsobiť. Prvá možnosť je doplniť metamodel UML prostredníctvom profilov. Tento spôsob sa nazýva ľahké rozšírenie. Druhým spôsobom je editovať metamodel UML.

Pre ľahké rozšírenie sú definované tri mechanizmy na úpravu metamodelu a to stereotypy (stereotypes), obmedzenia (constraints) a označené hodnoty. Profil je súhrn definícií týchto rozšírení aplikovateľný na určitú časť metamodelu . Je možné použiť aj viacnásobné aplikovanie rôznych profilov na tú istú časť metamodelu ak spĺňajú predpoklad, že v definovaných obmedzeniach nenastávajú konflikty. Veľkou výhodou použitia ľahkého rozšírenia je podpora tejto technológie v modelovacích nástrojoch.

Mechanizmus rozšírenia prostredníctvom stereotypov umožňuje vytvárať nové elementy odvodené z existujúcich elementov. Umožňuje doplnenie sémantiky, ale nedovoľuje meniť štruktúru metatriedy pôvodného elementu. Príkladom stereotypu, ktorý je v UML už nadefinovaný je napríklad element Rozhranie (označený kľúčovým slovom <<interface>>). Nový prvok môže byť označený štyrmi spôsobmi :

- názvom stereotypu v dvojitých ostrých zátvorkách , napríklad element Rozhranie
- názvom stereotypu v dvojitých ostrých zátvorkách a ikonou pridanou do pravého horného rohu
- pridanou ikonou umiestnenou v pravom hornom rohu

- novou ikonou, ktorá nahradí pôvodný symbol elementu, napríklad element Aktér

Mechanizmus rozšírenia formou doplnenia podmienok eliminuje vznik porušení špecifických obmedzení modelovanej domény. Obmedzenie môže byť pravidlo, alebo podmienka ktorá musí byť vždy splnená. Je definované v prirodzenom jazyku, alebo v jazyku OCL (Object Constraint Language). V diagrame je zobrazené ako text v zložených zátvorkách.

Prostredníctvom mechanizmu doplnenia označených hodnôt pridávame elementom nové vlastnosti. Takto pridaná vlastnosť obsahuje jej meno a hodnotu. Meno vlastnosti môže byť napríklad farba a jej hodnota bude červená. Označené hodnoty sa zapisujú vo forme meno = hodnota (farba = červená).

Vzhľadom na všeobecne známu snahu zjednotiť formu popisu systému, o ktorú sa jazyk UML snaží, konzorcium OMG prijalo štandardné profily pre často používané domény.

- UML Profil pre architektúru CORBA a model komponentov architektúry CORBA CCCMP (UML Profile for CORBA & CORBA® Component Model CCCMP) - je profil, ktorý je prispôsobený pre prácu s architektúrou Common Object Request Broker Architecture používanú pri distribuovaní objektov po sieti. Spája funkcie profilov UML Profil pre architektúru CORBA (UML Profile for CORBA) a UML Profil pre model komponentov architektúry CORBA CCM (UML Profile for CORBA Component Model CCM), ktoré má nahradiť.
- UML Profil pre distribúciu dát (UML Profile for Data Distribution) – rozširuje UML pre potreby návrhu real-time systémov pomocou Data Distribution Service.
- UML Profil pre integráciu aplikácií EAI (UML Profile for Enterprise Application Integration) - poskytuje rozšírenie metaúdajov pre podrobnejší popis rozhraní aplikácií a dát, ktoré rozhrania prenášajú.
- UML Profil pre distribuované objektové programovanie EDOC (UML Profile for Enterprise Distributed Object Computing) - zjednodušuje návrh komponentov EDOC
- UML Profil pre modelovanie a analýzu real-time systémov a vnorených systémov MARTE (UML Profile for Modeling and Analysis of Real-time and Embedded Systems) - poskytuje prostriedky na vytváranie návrhu real-time systémov a vnorených systémov. Je náhradou za starší profil s názvom UML Profil pre plánovanie, výkonnosť a čas (UML Profile for Schedulability, Performance and Time).

- UML Profil pre QoS modelovanie a charakteristiku porúch (UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms) - je rozšírenie pre technológiu Quality of Service, ktorá sa zaoberá kvalitou komunikačných systémov a sietí.
- UML Profil pre SDR technológiu (UML Profile for Software Radio) – je rozšírením UML pre modernú rádiovú technológiu Software Defined Radio, ktorá softvérovo spracováva prijatý signál, bez fyzického použitia demodulátorov, alebo filtrov.
- UML Profil pre vývoj systémov pre čipy SoCP (UML Profile for System on a Chip) - technológia SoCP sa zaoberá integráciou komponentov elektronického systému na jeden čip. Prostredníctvom tohto rozšírenia jazyka UML je možné podrobne popísať moduly a kanály čipov a komunikáciu medzi nimi.
- UML profil pre hlasové aplikácie (UML Profile for Voice) – toto rozšírenie umožňuje bližší popis systémov využívajúcich hlas.
- UML profil pre testovanie (UML Testing Profile) - je prostriedok pre zastrešenie testovania aplikácií.

V prípade, že je potrebné popísať doménu veľmi presne existuje ešte možnosť úpravy metamodelu. K tejto metóde sa pristupuje vtedy, keď nestačí dopĺňanie metamedelu, ale je potrebné vykonať zmeny napríklad v štruktúre elementov, alebo sa vyžaduje odstránenie existujúcich obmedzení. Veľkou nevýhodou takéhoto rozšírenia je, že ho štandardne editory pre vývoj softvéru v UML nepodporujú.

3 Analýza existujúcich nástrojov pre vývoj UML

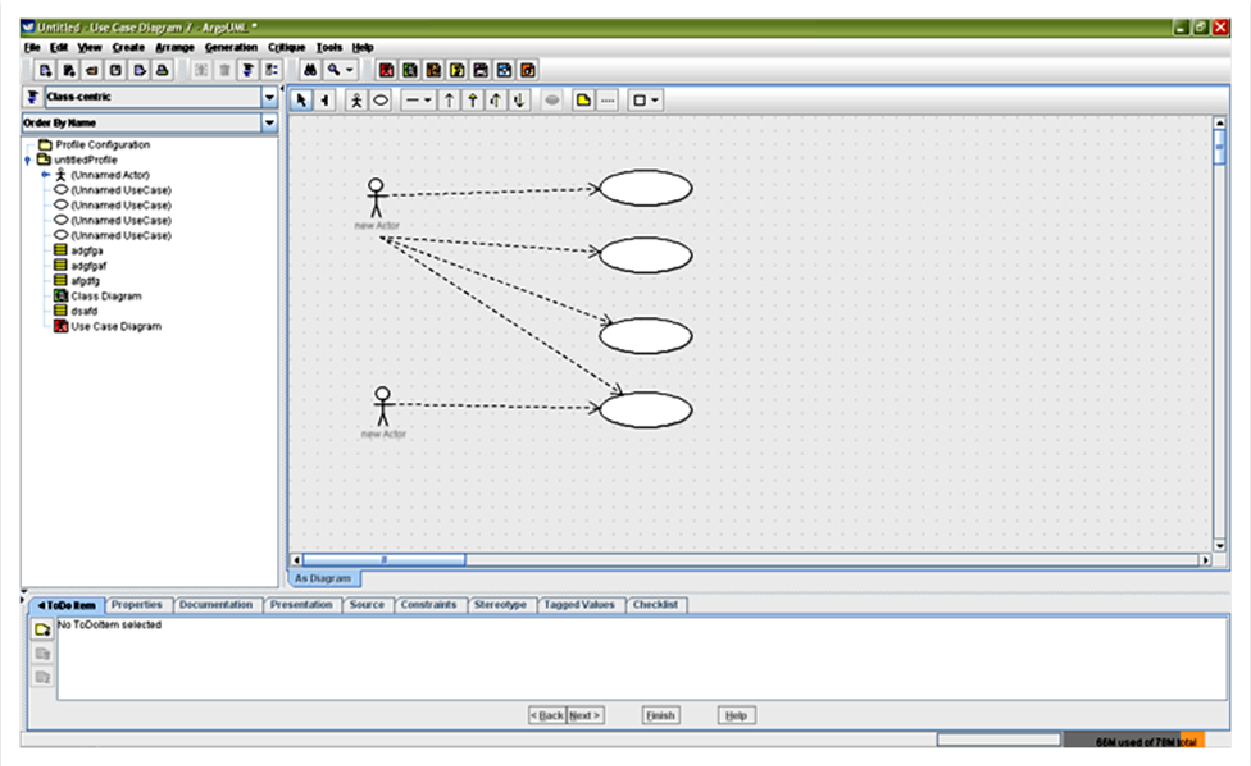
V tejto kapitole je hodnotenie vybraných UML editorov rôznych druhov. Porovnávanie editorov je vykonávané systematicky so zameraním sa na rovnaké hodnotiace kritériá (inštalácia softvéru, podporované diagramy, GUI, prehľadnosť pri práci s projektmi, intuitívnosť ovládania, grafické zobrazenie diagramov, podporované formáty, iné plusy a mínusy). Na konci kapitoly je sumár s vyhodnotením analýzy.

3.1 Voľne prístupné editory

Výhodou voľne prístupných editorov je v prvom rade, že sú zadarmo. Napriek tomu sa veľakrát poskytovanými funkciami vyrovnajú komerčným nástrojom. Používajú ich najmä študenti, prípadne malé firmy.

3.1.1 AgroUML

Inštalácia programu prebehla bez problémov. Po spustení aplikácie sa objaví očakávané rozvrhnutia užívateľského rozhrania.



V hornej časti programu je menu, pod ním je rýchle menu, ktoré štandardne obsahuje nástroje pre prácu s projektom, paletu nástrojov pre prácu s komponentom a paletu s ikonami pre vytváranie

nových diagramov. V ľavom stĺpci je zoznam častí projektu. Zoznam sa dá prispôsobiť a podľa možností ktoré si užívateľ vyberie zobrazuje podrobný zoznam tried, balíčkov, diagramov a iné. Tieto sa ďalej dajú triediť podľa typov alebo mena. Vzhľadom na možnosti týchto prispôsobení je projekt veľmi prehľadný z rôznych hľadísk.

Na pravo od panela je časť programu s kresliacou plochou. V jej hornej časti sa nachádza panel komponentov diagramu. Ten sa prispôsobuje podľa toho aký druh diagramu užívateľ vytvára. Ikony na panely sú jednoduché, zrozumiteľné, bez textovej reprezentácie. Tá sa zobrazí ak je kurzor myši nad tlačidlom vybraného komponentu. Zvláštnosťou pri pridávaní komponentov bolo, že väčšina komponentov sa pridáva stlačením tlačidla s vybraným komponentom a následne kliknutím na kresliacu plochu. Ak však užívateľ pridáva komponent Coment, ten nečaká na kliknutie myši, ale pridá sa automaticky do ľavého horného rohu kresliacej plochy. Ak je na ploche označený existujúci komponent, Coment sa pridá k nemu a automaticky sa prepoja.
Editovanie existujúcich komponentov je štandardné. Označené komponenty je možné napríklad vymazávať a presúvať. Pri dvojkliku na komponent systém zobrazí políčko pre zmenu názvu. Ostané parametre označeného komponentu sa menia v spodnej časti programu, kde je panel s podrobnými informáciami o komponente.

Panel s vlastnosťami komponentu je komplexný. Neumožňuje iba zmeny základných vlastností komponentov, ale navyše je umožnené pridávať takzvané ToDo poznámky, informácie o autorovi a verzii. Ďalej program dáva možnosť meniť základné prvky dizajnu komponentu, napríklad farbu obsahu a farbu čiary. Pridávanie kódu ku komponentu v jazykoch CSharp, PHP4, PHP5, C++, SQL a Java. Zaujímavá je posledná záložka tohto menu, ktorá je prístupná len pre niektoré komponenty ako napríklad Checklist. Obsahuje súhrn otázok z ktorých užívateľ môže označiť tie, ktoré majú spojitosť so zvoleným komponentom.

Program umožňuje ukladanie vytvorených projektov do troch formátom a to sú AgroUML compressed preject file (.zargo), AgroUML project file (.uml), XMI compressed project file (.zip). Ďalej poskytuje export do XML formátu XML Metadata Interchange (.xmi) a do grafických formátov PNG (.png), GIF (.gif), Scalable Vector Graphics (.svg), PostScript (.ps) a Encapsulated PostScript (.eps).

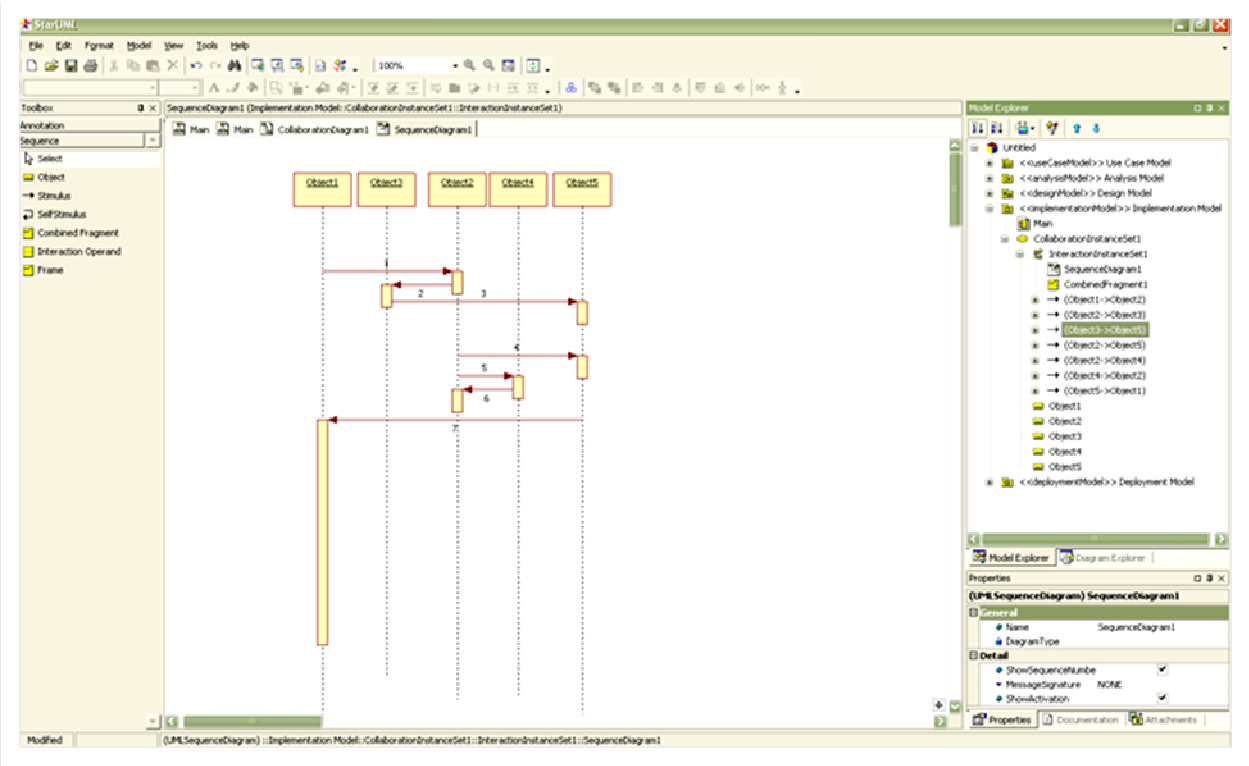
Program je vhodný pre začiatočníkov aj pokročilých systémových architektov. Je dostatočne prehľadný jednoducho ovládateľný a intuitívny. Verzia 0.32.1 pracuje podľa štandardov UML 1.4. Programu sa dá na prvý pohľad vytknúť veľmi strohá grafika diagramov.

Hodnotiace kritérium	Popis výsledku hodnotenia
Inštalácia softvéru	Jednoduchá
Podporované diagramy	Diagram prípadu použitia, Diagram tried, Sekvenčný diagram, Diagram spolupráce, Stavový diagram, Diagram aktivít a Diagram nasadenia
GUI	Prehľadné. Možnosť úpravy podľa potrieb užívateľa
Prehľadnosť pri práci s projektmi	Velmi dobrá
Intuitívnosť ovládania	Dobrá
Grafické zobrazenie diagramov	Priemerné s možnosťou jednoduchých úprav
Podporované formáty	.zargo (AgroUML compressed project file), .uml (AgroUML project file), .zip (XMI compessed project file), Export do XMI, Export do grafických formátov png, gif, svg, ps, eps
Špeciálne +	Checklist ku komponentom
Špeciálne -	

3.1.2 StartUML

Program StartUML je ľahko prístupný na internete a inštalácia prebehla bez problémov.

Tak ako v predchádzajúcom prípade je v hornej časti klasické menu pod ktorým je štandardné rýchle menu obsahujúce položky ako vytvoriť nový projekt, uložiť, kopírovať, priblížiť a podobne. Pod ním sa na ľavej strane nachádza panel s komponentmi, v strede je kresliaca plocha a v pravo hore prehľad toho čo obsahuje model a prehľad diagramov. V pravo dole sa nachádza panel s troma záložkami. V prvej záložke je súhrn vlastností aktuálneho komponentu s možnosťou ich zmeny. V druhej záložke môže užívateľ editovať / prezerať dokumentáciu komponentu a v poslednej záložke môže pridávať prílohy ku komponentu (odkazy na internetové zdroje, alebo súbory z počítača).



Po vytvorení nového projektu systém vytvorí automaticky päť základných modelov. Model prípadov použitia, model analýzy systému, model dizajnu systému, model implementácie a model nasadenia. Každý obsahuje jeden príslušný diagram. V projekte sa dajú dodatočne vytvárať aj ďalšie modely. Užívateľ má možnosť vytvárať diagramy iba k modelom.

Program podporuje deväť typov diagramov a to diagram prípadu použitia, diagram tried, sekvenčný diagram, diagram spolupráce, stavový diagram, diagram aktivít, diagram komponentov, diagram nasadenia a diagram zloženej štruktúry. A rozšírenia dvoch z týchto diagramov sekvenčný diagram úloh a diagram spolupráce úloh.

Panel s komponentmi je prehľadný, obsahuje ikonu aj slovný popis ku každému komponentu. Paleta komponentov sa prispôsobuje druhu diagramu, ktorý užívateľ modeluje. Je rozdelená na dve časti a to všeobecné anotácie (text, obdĺžnik, elipsa atď.) a komponenty prislúchajúce diagramu. Do všeobecných anotácií bol zaradený aj komponent Note, ktorý je štandardným komponentom v diagramoch. Umiestnenie tohto komponentu v tejto časti môže byť pre užívateľa mätúce. Novo pridané komponenty majú netradičné prednastavené názvy ako napr. Lee, Minkyu.

Práca s komponentmi je štandardná, označenie kliknutím na komponent, presun ťahom myši, zmena názvu dvojklikom. Po kliknutí pravým tlačidlom sa zobrazí menu, prostredníctvom ktorého sa pridávajú atribúty, alebo ďalšie komponenty. Správanie tohto menu mi nevyhovovalo.

Umožňuje napríklad pridať podradenú triedu označenej triede. Po stlačení vybranej voľby sa trieda objaví v ľavom panely s rozpisom komponentov ako súčasť triedy, ale v nákrese diagramu sa nič nestane. Umožňuje napríklad aj pridanie šablónových parametrov, ktoré sa dajú odstrániť iba prostredníctvom ľavého menu s vlastnosťami komponentu a nedajú sa premenovať.

Dizajn komponentov je veľmi jednoduchý. Program umožňuje meniť veľkosť a farbu textu, farbu pozadia a farbu orámovania komponentu.

Prostredníctvom programu je možné vytvoriť výstupný súbor typu StaerUml project file (.uml). Projekt je možné exportovať do XMI (.xml). Jednotlivé diagramy možno exportovať do piatich grafických formátov. Sú to JPG (.jpg), JPEG (.jpeg), Bitmap (.bmp), Enhanced metafile (.emf), Metafile (.wmf).

Program nie je vhodný pre začiatočníkov, ovládanie je v niektorých prípadoch netriviálne. Zložitosť programu zrejme ocení pokročilý používateľ.

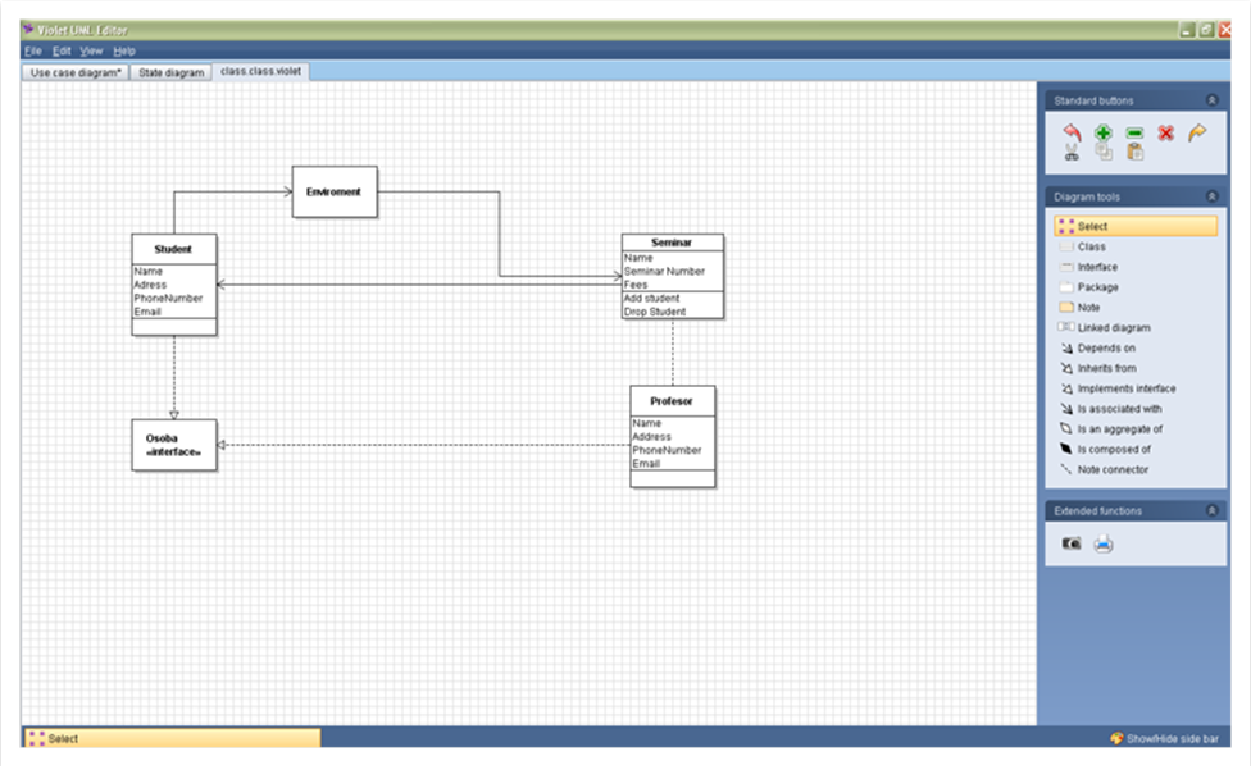
Hodnotiace kritérium	**Popis výsledku hodnotenia**
Inštalácia softvéru	Jednoduchá.
Podporované diagramy	Diagram prípadu použitia, Diagram tried, Sekvenčný diagram, Sekvenčný diagram úloh, Diagram spolupráce, Diagram spolupráce úloh, Stavový diagram, Diagram aktivít, Diagram komponentov, Diagram nasadenia a Diagram zloženej štruktúry
GUI	Prehľadné. Možnosť úpravy podľa potrieb užívateľa
Prehľadnosť pri práci s projektmi	Dobrá.
Intuitívnosť ovládania	Slabá
Grafické zobrazenie diagramov	Priemerné s možnosťou jednoduchých úprav.
Podporované formáty	.uml (StartUML project file), Export do XMI (.xml), Export jednotlivých diagramov do grafických formátov jpg, jpeg, bmp, emf, wmf
Špeciálne +	
Špeciálne -	Program je pre začiatočníka pomerne zložitý.

3.1.3 Violet UML Editor

Je malý veľmi jednoduchý program na tvorbu UML diagramov. Nevyžaduje inštaláciu. Je to jeden jar súbor, ktorý stačí stiahnuť.

Po spustení sa zobrazí horné menu a úvodná obrazovka. Ne tej si užívateľ vyberie či chce vytvárať nový diagram, alebo otvoriť jeden z posledných piatich editovaných súborov (môže otvoriť aj iné súbory, ale musí použiť menu).

Potom výbere program zobrazí kresliacu plochu a v pravo panel s nástrojmi. Nad plochou stále ostáva štandardné menu. Pod kresliacou plochou je vypísaný aktuálne zvolený komponent, ktorý sa po kliknutí myšou pridá na jej súradnice do kresliacej plochy, alebo informácia a tom že momentálne je zvolený selektor.



Panel nástrojov je veľmi jednoduchý. Je rozdelený do troch logických častí. Sú to štandardné funkcie, funkcie diagramu a rozširujúce funkcie. Medzi štandardné funkcie patrí napríklad krok späť, lupa, kopírovanie alebo vymazanie označeného komponentu. Panel s funkciami diagramu sa mení vzhľadom na to aký diagram vytvárame. Vždy obsahuje selektor. Každý komponent je reprezentovaný ikonou aj slovným popisom.

Komponenty sa pridávajú tak, že užívateľ označí druh komponentu, ktorý chce pridať a ak klikne do diagramu na prázdne miesto systém tam vykreslí prázdny komponent vybraného typu. Po pridaní komponentu zostane naposledy pridávaný komponent označený. Ak užívateľ už nechce pridávať komponenty musí vybrať v panely pre prácu s diagramom selektor.

Vlastnosti komponentu sa menia v dialógových oknách, ktoré sa zobrazujú po dvojkliku na označený komponent, alebo po kliknutí pravým tlačidlom myši. Program umožňuje uchovávanie len tých najzákladnejších vlastností komponentov.

Violet UML Editor ukladá diagramy do svojich špecifických formátov podľa druhu diagramu. Sú to informácie zaznamenané v xml štruktúre. Umožňuje aj export do súborov typu Image File.

Tento program je vhodný pre úplných začiatočníkov, ktorý sa učia vytvárať UML diagramy. Nezaťažuje užívateľov s vytváraním zložitých projektov. Pracuje len s úplne základnými stavebnými komponentmi a najpoužívanejšími diagramami. Na rozdiel od predchádzajúcich dvoch programov je graficky príjemný.

Hodnotiace kritérium	**Popis výsledku hodnotenia**
Inštalácia softvéru	Stačí spustiť.
Podporované diagramy	Diagram prípadu použitia, Diagram tried, Diagram objektov, Stavový diagram, Diagram aktivít, Sekvenčný diagram
GUI	Prehľadné s príjemným dizajnom.
Prehľadnosť pri práci s projektmi	Nepodporuje prácu s projektami.
Intuitívnosť ovládania	Dobrá.
Grafické zobrazenie diagramov	Pekné.
Podporované formáty	.class.violet (Class Diagram Files) .usecase.violet (Use Case Diagram Files) .object.violet (Object Diagram Files) .state.violet (State Diagram Files) .activity.violet (Activity Diagram Files)

	.seq.violet (Sequence Diagram Files) Export do súborov typu Image File
Špeciálne +	Veľmi jednoduchý program ideálny na učenie sa základov UML. Graficky príjemné užívateľské prostredie.
Špeciálne -	Nehodí sa na návrh reálneho rozsiahleho systému.

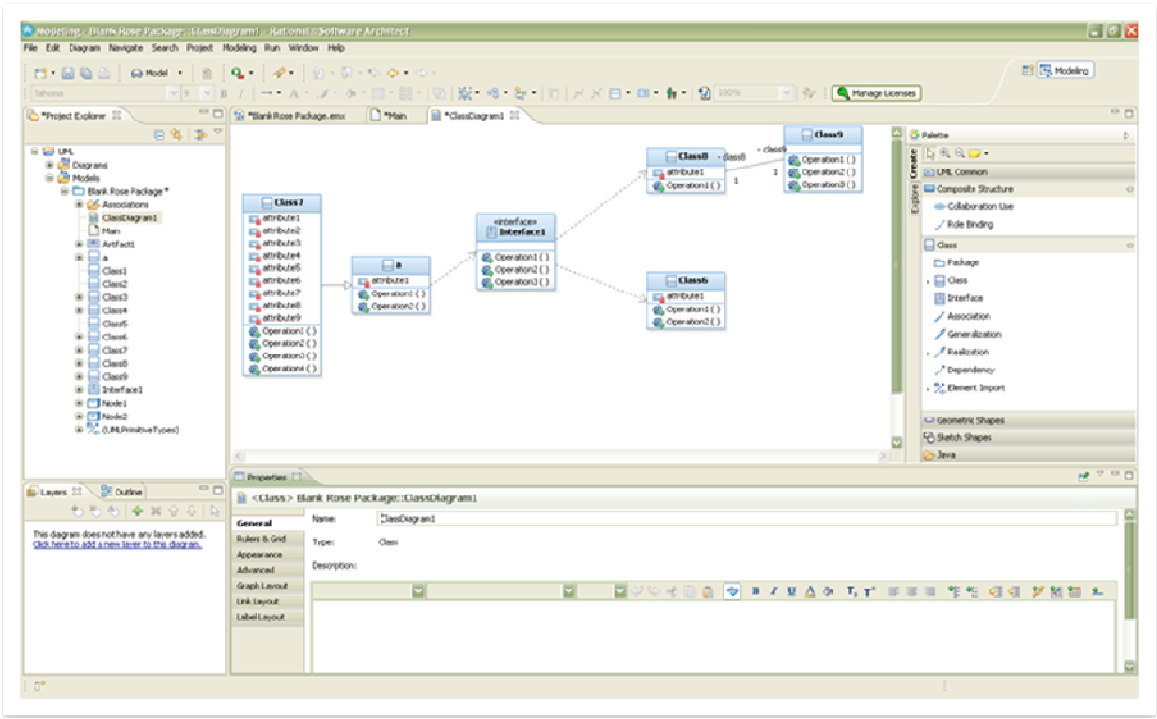
3.2 Komerčné nástroje

Vzhľadom na to, že UML je jazyk, ktorý sa neustále vyvíja je potrebné, aby modelovacie nástroje flexibilne reagovali na tieto zmeny. A to je väčšinou jedna z dôležitých výhod komerčných nástrojov. Sú prevažne používané vo väčších firmách.

3.2.1 Rational Rose

Nástroj Rational Rose je vývojové prostredie navrhnuté firmou IBM, ktorá sa podieľa na vývoji nových štandardov pre UML. Preto možno predpokladať, že práve tento softvér by mal byť jeden z najkvalitnejších nástrojov pre návrh systému v jazyku UML.

Ratiolan Rose je veľmi obsiahly a komplexný program. Vzhľadom k tomu bola inštalácia softvéru náročnejšia ako pri predchádzajúcich softvéroch. Počas inštalácie má používateľ možnosť napríklad nastaviť prepojenia s prostredím na vývoj Java aplikácií Eclipse.



Používateľ, ktorý má predošlé skúsenosti s programom Eclipse po spustení Rational Rose zbadá známe prostredie. V hornej časti sa nachádza štandardné menu a pod ním rýchle menu programu. Pod nimi je ďalšie rýchle menu pre prácu s komponentmi. V ľavom stĺpci pod menu je zobrazená štruktúra projektu. V strednej časti je kresliaca plocha a na pravo je panel prostredníctvom ktorého užívateľ pridáva komponenty. Sú roztriedené do záložiek podľa druhu diagramu. Posledná záložka Java umožňuje užívateľovi pridávať komponenty Java projektov ako Class, Interface a pod. . V ľavej spodnej časti projektu je paleta Layers pre prácu s hladinami. Vedľa nej je panel nástrojov s názvom Properties. Jeho obsah sa mení vzhľadom na to aký komponent je označený. Toto základné rozloženie je editovateľné a užívateľ ho môže meniť podľa svojich potrieb. Po zmene si môže svoje prispôsobenia uložiť a neskôr ich kedykoľvek použiť.

V prvom kroku modelovania je potrebné, aby si užívateľ vytvoril projekt. Program obsahuje dostatok šablón štruktúr projektov zoradených do kategórií. Neskúsený používateľ by mohol mať problém s vybraním správneho typu projektu.

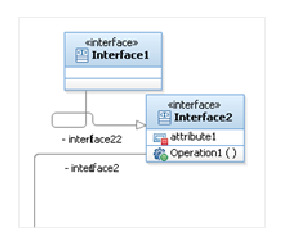
Po vytvorení projektu a pridaní diagramu do projektu používateľ môže začať modelovať diagram. Pre pridanie komponentu si označí v pravom stĺpci jeho typ a klikne do kresliacej plochy. Komponent sa vykreslí na mieste kliknutia. Vlastnosti komponentov sa dajú meniť viacerými spôsobmi. Prejdením myši nad komponent sa zobrazuje nad komponentom dialógová bublina s ikonami na pridanie vlastností. Napríklad pre komponent trieda sú to dve ikony. Jedna pridá novú operáciu a druhá pridáva atribúty. Navyše sa pri komponentoch zobrazujú dve šípky, ktoré umožňujú pridať spojenie medzi komponentmi. Rozšírené vlastnosti komponentu je možné meniť prostredníctvom Properties, pod kresliacou plochou. Obsahuje vlastností špecifikované pre komponent, ale aj mnoho ďalších vlastností týkajúcich sa napríklad dokumentácie komponent, grafického zobrazenia, či nastavení pre pokročilých užívateľov. Posúvanie komponent je realizované ťahom myši.

Editor Rational Rose ukladá projekty do formátov, ktoré sú špecifické pre vybraný typ projektu. Jednotlivé diagramy je možné exportovať do šiestich grafických formátov. Sú to JPG (.jpg), JPEG (.jpeg), Bitmap (.bmp), PNG (.png), GIF (.gif), Scalable Vector Graphics file (.svg). A do formátu Portable Document Format (.pdf).

Tento program je veľmi rozsiahly a zároveň dostatočne prehľadný. Pokročilému užívateľovi umožňuje vytvoriť pomocou preddefinovaných konštrukcií komplexný návrh systému.

Začiatočníci môžu mať problém s výberom vhodného projektu, ak však túto prekážku prekonajú za odmenu dostanú „user-best-friendly“ rozhranie na vytváranie diagramov.

Hodnotiace kritérium	Popis výsledku hodnotenia
Inštalácia softvéru	Zložitá.
Podporované diagramy	Podporuje všetky druhy diagramov, navyše umožňuje vytváranie profilov rozšírení.
GUI	Prehľadné s príjemným dizajnom a možnosťou prispôsobenia.
Prehľadnosť pri práci s projektmi	Dobrá.
Intuitívnosť ovládania	Dobrá. Ovládanie je user-friendly.
Grafické zobrazenie diagramov	Nadštandardné.
Podporované formáty	Špecifické pre rôzne druhy projektov. Export do súborov typu JPEG Image, Bitmap Image, PNG Image, GIF Image, Scalable Vector Graphics file, Portable Document Format
Špeciálne +	Tento program je vhodný na tvorbu profesionálneho návrhu systému.
Špeciálne -	Počas testovania trial-verzia tohto programu viackrát padla.



Aby bolo potvrdené pravidlo, že nikto nie je dokonalý prikladám obrázok s chybou v aplikácii, kde sa mi nedopatrením podarilo posúvaním jedného z dvoch spojených komponentov zauzliť čiaru reprezentujúcu zovšeobecnenie medzi komponentmi. Čiara sa pri pohybe komponentom posúvala automaticky a predpokladám, že toto nie je želaný stav.

3.3 Prídavné moduly do prostredí

Prídavné moduly do prostredí majú tú výhodu, že užívateľ si nemusí zvykať na nové prostredie. Nevýhodou obvykle je, že oproti nástrojom, ktoré sú špecializované na modelovanie, obsahujú len základnú funkcionalitu pre prácu s UML.

3.3.1 NetBeans IDE plugin UML

Inštalácia pluginov pre prostredie NetBeans je vo všeobecnosti jednoduchá. Stačí si v menu aplikácie v záložke Tools vybrať možnosť Plugins. Zobrazí sa okno so záložkou s prístupnými rozšíreniami, kde stačilo vybraný plugin zaškrtnúť a stlačiť tlačidlo Install. Inštalácia trvala niekoľko sekúnd.

Po nainštalovaní vznikla nová možnosť vo výbere typu projektu. Pribudol priečinok UML, ktorý obsahuje tri typy projektov. Platform-Independent Model, ktorý je platformovo nezávislý a neumožňuje generovanie kódu. Java-Platform Model v ktorom má užívateľ možnosť vygenerovať Java kód pre modelovaný diagramy. Reverse Engineered Java-Platform Model ktorý generuje UML diagramy z existujúceho Java projektu.

Pri vytváraní projektu (typu Platform-Independent Model) je v jednom z prvých krokov potrebné zvoliť typ diagramu. Je podporovaných päť typov diagramov. Sú to diagram aktivít, diagram tried, sekvenčný diagram, diagram stavov a diagram prípadu použitia. Do projektu je možné pridať aj ďalšie diagramy, ale až po jeho vytvorení. Celkovo je vytvorenie UML projektu jednoduché a pozostáva z troch krokov.

Po vytvorení diagramu sa v ľavom stĺpci programu zobrazí paleta s komponentmi a pod ňou panel s informáciami o diagrame. Komponenty sú zobrazené ako ikony s popisom. Paleta komponentov sa na rozdiel od iných programov neprispôsobuje typu diagramu. Komponenty vkladáme štandardne označením typu komponentu a kliknutím do kresliacej plochy. Vlastnosti komponentu užívateľ mení kliknutím na komponent pravým tlačidlom myši. Týmto spôsobom je možné napríklad pridať atribút, alebo operáciu. Vlastnosti komponentu, ako názov, viditeľnosť, či stereotyp sa menia v pravom stĺpci programu pod paletou komponentov. Navyše sa po označení komponentu zobrazí na pravej strane malá paleta na rýchle pridanie nového komponentu, alebo závislosti.

Projekty sa ukladajú v špeciálnych NetBeans formátoch. Diagram je možné exportovať do formátov JPEG (.jpg) a PNG (.png).

Plugin UML do prostredia NetBeans sa jednoducho inštaluje a poskytuje užívateľovi výhodu domáce prostredia. Ovládanie je intuitívne. Obsahuje navyše aj malú palu na rýchle pridávanie komponentov. Menšou nevýhodou pre začiatočníkov je, že paleta s nástrojov sa neprispôsobuje typu diagramu. Komponenty sú pekne graficky zobrazené. Pre pokročilých užívateľov je výhoda

tohto prostredia v kvalitnom generovaní kódu, ktorý môžu priamo v tomto prostredí editovať a pracovať s ním.

Hodnotiace kritérium	Popis výsledku hodnotenia
Inštalácia softvéru	Veľmi jednoduchá.
Podporované diagramy	Diagram aktivít, Diagram tried, Sekvenčný diagram, Diagram stavov a Diagram prípadu použitia
GUI	Výhoda domáceho prostredia.
Prehľadnosť pri práci s projektmi	Dobrá.
Intuitívnosť ovládania	Dobrá. Ovládanie je štandardné.
Grafické zobrazenie diagramov	Pekné.
Podporované formáty	Špecifické pre rôzne druhy projektov. Export do grafických formátov JPEG Image a PNG Image.
Špeciálne +	Paleta na rýchle pridanie prvku pri komponentoch.
Špeciálne -	Paleta komponentov sa neprispôsobuje typu modelovaného diagramu.

3.4 Zhrnutie

Každý z testovaných programov mal svoje výhody a nevýhody. Vytvorila som súbor hodnotiacich kritérií, ktoré boli prispôsobené vzhľadom na fakt, že testovanie bolo zamerané na určitú skupinu užívateľov a to začiatočníkov. Celkové vyhodnotenie podľa definovaných kritérií je nasledovné:

- Inštalácia softvéru bola väčšinou jednoduchá a bezproblémová. Dva extrémy tvorili Violet UML editor, ktorý inštaláciu nepotrebuje, stačí ho spustiť a program Rational Rose, ktorého inštalácia je zložitejšia a zdĺhavejšia v porovnaní s ostatnými testovanými nástrojmi.
- Najviac druhov diagramov podporuje program Rational Rose. Druhým najlepším nástrojom čo sa týka počtu podporovaných diagramov bol Start UML Editor. Jeho veľká nevýhoda však bola v tom, že nepracuje s aktuálnou verziou UML 2.0. Najmenej diagramov podporoval plugin do programu NetBeans. Umožňuje modelovať iba päť najpoužívanejších diagramov.

- Najkrajšie grafické rozhranie má program Violet UML Editor, vzhľadom na jeho jednoduchosť nie je ťažké dosiahnuť maximálnu prehľadnosť. Dobré riešenie grafického rozhrania má program Agro UML. Je jednoduchý a prehľadný. Väčšina testovaných nástrojov umožňovala prispôsobenie užívateľského rozhrania.
- Prehľadnosť pri práci s projektmi je dobre riešená v programe Agro UML. Užívateľ má možnosť vyselektovať zo zoznamu komponentov diagramu len tie, ktoré ho zaujímajú. Potom ich môže ďalej zoraďovať.
- Intuitívnosť ovládania bola vo väčšine programov dobrá. Výnimku tvoril program Start UML, kde bolo ovládanie o niečo horšie. Najpohodlnejšie ovládanie majú programy Rational Rose a UML plugin do NetBeans.
- Grafické zobrazenie diagramov vo voľne prístupných editoroch bolo veľmi strohé. Následkom toho sú veľmi strohé aj vyexportované obrázky diagramov. Lepšie vyzerajúce diagramy sa dajú vytvárať v programoch Rational Rose a UML plugin do NetBeans.
- Väčšina editorov si ukladá projekty a diagramy do svojich špecifických typov súborov a umožňujú export do bežne používaných grafických formátov. Programy Agro UML a Start UML umožňujú navyše export do XML. Program Rational Rose vie vyexportovať diagram aj do pdf súboru.

Z programov v tejto kapitole je pre začiatočníka najvhodnejšia voľba program AgroUML. Je to kompromis medzi priveľmi jednoduchými programami ako Violet UML Editor a zložitými programami ako Rational Rose, ktoré obsahujú veľa funkcií, ktoré by používateľ začiatočník pravdepodobne nevyužil. V porovnaní s programom Start UML má podstatne lepšie ovládanie a je intuitívnejší. Plugin UML do prostredia NetBeans je druhou najlepšou voľbou, ktorá je zaujímavá hlavne pre používateľov tohto prostredia.

4 Požiadavky na systém

Program Online UML editor má za úlohu umožniť viacerým používateľom naraz vytvárať diagramy a popri tom im umožniť vzájomnú komunikáciu. Používatelia budú vytvárať viacero typov štandardných UML diagramov. Zmeny, ktoré v diagramoch vykonajú, môžu prostredníctvom aplikácie poslať na server a následne ďalším používateľom. Ďalší používatelia majú možnosť sa k zmene vyjadriť, prípadne ju medzi sebou prediskutovať prostredníctvom mini-chatu.

4.1 Cieľová skupina užívateľov

Program je primárne určený pre informaticky gramotných študentov vysokých, prípadne stredných škôl. Má za úlohu umožniť študentovi efektívne spolupracovať s ďalšími spolužiakmi pri riešení projektov zameraných na tvorbu UML diagramov. Projekt budú môcť takto riešiť študenti z viacerých miest.

Využitie programu môžu nájsť aj profesionálny vývojári. Tejto skupine používateľov poslúži program na sprostredkovanie dynamickej online komunikácie na diaľku. Prostredníctvom programu majú možnosť spoločne načrtnúť diagramy. Pomocou týchto náčrtov môžu ďalej vytvárať diagramy v profesionálnych editoroch, ktoré štandardne používajú.

4.2 Moduly aplikácie

Celá aplikácia je zložená z troch základných častí a to sú prihlasovanie a identifikácia užívateľov, UML editor diagramov a mini-chat. Prihlasovanie a identifikácia užívateľa bude modul, ktorého základom bude databáza s informáciami o užívateľoch. Pomocou modulu UML editora budú používatelia vytvárať diagramy, pridávať do nich prvky, editovať ich, ukladať a posielať zmeny ostatným užívateľom. Mini-chat je určený na komunikáciu užívateľov v skupine.

4.2.1 Prihlasovanie prostredníctvom databázy

Prihlasovanie a identifikácia užívateľa je modul, v ktorom budú uchovávané informácie o používateľoch. Po spustení aplikácie sa bude musieť používateľ prihlásiť. Systém bude pracovať s databázou užívateľov, ktorú vytvorí vyučujúci. Overí prihlasovacie údaje a po úspešnom prihlásení užívateľovi priradí skupinu podľa informácií v databáze.

4.2.2 UML editor

Tento modul je základom aplikácie. Prostredníctvom neho budú mať užívatelia možnosť vytvárať, editovať, ukladať a otvárať diagramy, obdobne ako v bežných editoroch tohto typu. Oproti nim pribudne možnosť posielať zmeny v diagramoch aj ostatným užívateľom.

4.2.2.1 Práca s diagramom

Diagram je objekt, pomocou ktorého definujeme určité vlastnosti vyvíjaných systémov. Rôzne typy diagramov špecifikujú niečo iné a preto sú ich komponenty rozdielne.

Pri vytváraní diagramu si užívateľ určí najprv jeho typ . Podľa toho aký typ diagramu si zvolil dostaneme príslušnú paletu komponentov, ktoré môžeme do diagramu vložiť. Tieto komponenty budú v oddelenom menu. Položky menu budú ikony znázorňujúce jednotlivé komponenty. Každá ikona znázorňujúca komponent bude mať popis (tooltip) pre prípad, že by používateľovi nebolo jasné aký komponent ikona znázorňuje.

Popis diagramov a komponentov, z ktorých sa skladajú :

Typ diagramu	Komponenty
Diagram prípadu použitia (Use case diagram)	Agregácia (Aggregation) Aktér (Actor) Poznámka (Node) Prípad použitia (Use case) Rozšírenie (Extend) Zahrnutie (Include) Závislosť (Dependency) Zovšeobecnenie (Generalization)
Diagram tried (Class diagram)	Agregácia (Aggregation) Asociácia (Association) Balíček (Package) Poznámka (Node) Realizácia (Realization) Rozhranie (Interface) Trieda (Class) Závislosť (Dependency) Zovšeobecnenie (Generalization)

Diagram komponentov (Component diagram)	Asociácia (Association) Balíček (Package) Komponent (Component) Poznámka (Node) Realizácia (Realization) Rozhranie (Interface) Závislosť (Dependency)
Sekvenčný diagram (Sequence diagram)	Návratová hodnota (Return message) Objekt (Object) Poslanie správy (Send message) Poznámka (Node)
Stavový diagram (State diagram)	Koniec scenára (Scenario end) Poznámka (Node) Prechod (Transition) Stav (State) Synchronizácia (Synchronization bar) Štart scenára (Scenario start)
Diagram činností (Activity diagram)	Akcia (Action) Koniec scenára (Scenario end) Podmienka (Decision) Poslaný signál (Signal send) Poznámka (Node) Prechod (Transition) Prijatý signál (Signal accept) Stav objektu (Object flow state) Synchronizácia (Synchronization bar) Štart scenára (Scenario start)

Po vytvorení alebo otvorení diagramu ho bude vedieť používateľ editovať. Vkladať do diagramu nové komponenty, meniť pozície a ďalšie vlastnosti (popisy, mená atď.) už existujúcim komponentom.

Zoznam zložitejších komponentov s ich vlastnosťami, ktoré budú editovateľné :

Komponenty	Editovateľné atribúty
Agregácia (Aggregation)	Začiatočná šípka, Koncová šípka, Štýl čiary
Akcia (Action)	Popis
Aktér (Actor)	Názov, Popis
Asociácia (Association)	Začiatočná šípka, Koncová šípka, Štýl čiary
Balíček (Package)	Názov
Objekt (Object)	Názov objektu, Názov triedy
Podmienka (Decision)	Podmienka
Prechod (Transition)	Začiatočná šípka, Koncová šípka, Štýl čiary
Prípad použitia (Use Case)	Názov, Popis
Stav (State)	Názov, Popis
Trieda (Class)	Názov, Popis, zoznam premenných, zoznam operácií
Závislosť (Dependency)	Začiatočná šípka, Koncová šípka, Štýl čiary
Zovšeobecnenie (Generalization)	Začiatočná šípka, Koncová šípka, Štýl čiary

Zmeny bude možné lokálne ukladať, prípadne posielať ostatným užívateľom. Uložené diagramy sa budú dať pomocou aplikácie opätovne otvárať a ďalej editovať.

4.2.2.2 Online práca s diagramom a prenos udalostí

Aplikácia online UML editor bude umožňovať viacerým používateľom interaktívne vytváranie diagramov. Používateľ po prihlási bude priradený do svojej skupiny, s ktorou potom komunikuje a pracuje na vytváraní diagramu. Zmeny sa ukladajú a ak užívateľ skončí editáciu, vykonané zmeny pošle na server, prostredníctvom ktorého sa dostanú k ostaným užívateľom.

V online editovacom móde môžu meniť diagram viacerí užívatelia naraz. Užívatelia majú tie isté možnosti meniť diagram, ako pri editovaní diagramu na lokálnej úrovni. Užívateľ, ktorý chce diagram meniť, označí objekt v diagrame a pracuje s ním. Ostatní užívatelia vidia zmeny v reálnom čase a majú možnosť medzitým editovať ostatné objekty.
Aby užívatelia zaregistrovali všetky zmeny, budú sa zobrazovať aj v textovej podobe. Každý výpis bude obsahovať meno užívateľa, čas kedy sa zmena stala a popis zmeny. V popise zmeny bude presne určený objekt, ktorý sa zmenil a popis zmeny (napr. Objekt Class „Trieda1“ bol premenovaný na „Class1“ , Objekt Class „Class1“ bol premiestnený atď.) .

4.2.2.3 Synchronizácia

Po akejkoľvek zmene vlastností objektu v diagrame sa budú zmeny posielať v reálnom čase automaticky ostatným užívateľom. Pre dosiahnutie čo najlepšieho času prenosu zmien medzi užívateľmi sa nebude posielať celý diagram, ale iba informácie o zmenách, na základe ktorých sa budú aktualizovať verzie diagramov ostatných užívateľov.

Ak sa prihlási užívateľ do skupiny ktorá vytvára diagram s meškaním, bude sa musieť najprv zosynchronizovať. Server mu po prihlásení automaticky pošle zoznam udalostí, ktoré skupina vykonala na diagrame do aktuálneho času. Systém udalosti vykoná, aktualizuje diagram a následne umožní užívateľovi spolupracovať so svojou skupinou.

4.2.3 MiniChat

Každý užívateľ používajúci aplikáciu bude prihlásený. Po prihlásení ho systém zaradí do skupiny. Užívatelia medzi sebou komunikujú len v rámci svojich skupín. V časti Mini-chat systém zobrazí informácie o skupine, okno s konverzáciu medzi užívateľmi a formulár pre odoslanie správy.

Informácie o skupine budú obsahovať zoznam užívateľov patriacich do skupiny a grafickú informáciu o tom či je užívateľ pripojený, alebo nie.

Do okna s konverzáciou sa štandardne vypisujú správy, ktoré užívatelia napísali a správy, ktoré opisujú udalosti vykonané na diagrame, ktoré generuje systém. Tieto dva druhy správ budú odlíšené, aby každému užívateľovi bolo zrejmé, ktorá správa je od systému a ktorú napísal užívateľ. Užívateľ bude mať možnosť vypisovanie správ generovaných systémom skryť, aby nezahlcovali okno s konverzáciou.

V každej správe bude uvedený čas odoslania, meno užívateľa a text správy. Meno užívateľa bude obsahovať aj generované správy opisujúce zmeny v diagrame, aby bolo jasné, kto ktorú zmenu vykonal.

4.3 Užívateľské rozhranie

Rozhranie aplikácie by malo byť rozložené podobne ako v existujúcich nástrojoch, aby sa v ňom používateľ rýchlo zorientoval.

5 Návrh systému

Táto kapitola obsahuje informácie o technickom návrhu aplikácie Online UML Editor. Poskytuje informácie o použitých technológiách, návrhu riešenia modulu prihlasovania, popisuje objekt diagram a jeho komponenty. Ďalej je v nej popísaná sieťová komunikácia a prenášanie udalostí prostredníctvom server-socked technológie. Posledná časť kapitoly sa zaoberá výstupmi aplikácie.

5.1 Popis použitých technológií

Na vytvorenie aplikácie sa používa objektovo orientovaný programovací jazyk Java. Je to relatívne nový, ale silný programovací jazyk. Prvá verzia bola vydaná v roku 1995. Základ syntaxe Javy vychádza z jazyka C++. Tento jazyk je platformovo nezávislý.

Dôvodom pre výber tohto jazyka bolo to, že poskytuje množstvo štandardne dodávaných knižníc, ktoré uľahčujú prácu programátora. Ten potom nemusí vytvárať rovnaké často používané funkcie. Druhé významné plus je, že tento jazyk podporuje serializáciu ktorá sa využíva pri ukladaní objektov, ale aj pri posielaní objektov po sieti.

5.2 Prihlasovanie

Prihlasovanie bude realizované prostredníctvom okna s prihlasovacím formulárom. Prihlasovací formulár bude obsahovať dva textové editovateľné polia a tlačidlo „Prihlásiť“. Po stlačení tohto tlačidla sa spustí proces overenia údajov. Prihlasovacie meno systém vyhľadá v databáze. Ak existuje overí správnosť hesla, ak nie vypíše užívateľovi upozornenie o nesprávne zadanom prihlasovacom mene alebo hesle.

Na zapamätanie dát o užívateľovi slúži objekt *Uzivatel*. Vytvára sa pri spustení aplikácie. Objekt *Uzivatel* zhromažďuje informácie o užívateľovi v položkách *Meno, Priezvisko, Prihlasovacie meno, Skupina*. Po korektnom prihlásení užívateľa, sa nastavia hodnoty zodpovedajúce prihlásenému užívateľovi načítané z databázy užívateľov.

5.3 Databáza užívateľov

Pre správne fungovanie programu musí databáza, na ktorú sa program bude pripájať obsahovať presne špecifikované stĺpce tabuľky.

Názov	Typ	Dĺžka	Extra	Primary
ID	int	-	AUTO_INCREMENT	áno
Meno	varchar	50	-	-
Priezvisko	varchar	50	-	-
Skupina	-	-	-	-
Login	varchar	50	-	-
Heslo	text	-	-	-

5.4 Objekt Diagram

Vlastnosti vyvíjaných diagramov budú uchovávané v objektoch typu *Diagram*. Tento objekt bude uchovávať informácie o diagrame ako meno diagramu a typ diagramu. Obsah diagramu bude uchovávaný ako zoznam objektov *Komponenty*. Ďalšia úloha objektu *Diagram* je uloženie histórie diagramu. Riešenie tohto problému zabezpečí zoznam objektov Udalosť, ktorý bude obsahovať informácie o všetkých zmenách, ktoré boli na diagrame vykonané. Zaznamenáva všetko od informácie o vytvorení diagramu, cez informácií o pridávaní komponentov až po udalosti, ktoré súvisia so zmenami na jednotlivých komponentoch diagramu.

5.5 Komponenty

Každý typ komponentu v diagrame má množinu spoločných vlastností a operácií, ktoré sa dajú s objektom vykonávať. Medzi takéto vlastnosti patrí identifikátor komponentu, jedinečný pre každý komponent. Ďalej názov komponentu, súradnica komponentu na x-ovej osi, súradnica na y-novej osi, šírka a výška komponentu. Okrem týchto vlastností obsahuje každý komponent špeciálny príznak, ktorý určuje či je objekt označený a deklaráciu abstraktnej funkcie, ktorá objekt vykreslí.

Tieto informácie združuje nadtrieda *Komponenty*. Pre každý typ komponentu existuje ďalej podtrieda, kde sú uložené jeho ďalšie špecifické vlastnosti. Napríklad pre typ komponentu *Trieda* to je zoznam atribútov a zoznam operácií.

5.6 Prenášanie objektov prostredníctvom modelu Klient-Server

Sieťová komunikácia potrebná pre program je realizovaná na základe modelu Klient-Server. Ideálnym riešením by bolo mať neustále k dispozícii bežiaci server, na ktorom by sa mohli ukladať diagramy, ktoré užívatelia vyvíjajú. Problém je v tom, že užívatelia nemusia mať vždy k dispozícii bežiaci server. Riešenie problému spočíva v tom, že sa server aplikácia vytvorí po prihlásení prvého užívateľa. Pri prihlasovaní ostatných užívateľov sa budú vytvárať už len nový

klienti, ktorý sa budú pripájať na existujúci server. Keďže server spracúva požiadavky klientov a zabezpečuje rozposielanie správ medzi klientmi je nutné aby bežal počas celej doby používania programu.

5.7 Prenos správ a udalostí

Na prenos správ a udalostí použijeme Serializáciu. Serializácia je jednou z výhod, ktoré poskytuje jazyk Java. Poskytuje jednoduché riešenie pri ukladaní a posielaní objektov. Implementácia tohto rozhrania na triedu poskytuje možnosť zapisovať a čítať objekty danej triedy.

Pre tento konkrétny prípad sa posielajú ako objekty inštancie triedy popisujúcej správu. Systém rozlišuje dva typy správ. Jeden typ správ popisuje text, ktorý chce užívateľ v rámci komunikácie poslať ostatným užívateľom. Druhým typom správy je reakcia systému na akciu užívateľa vykonanú nad diagramom. Takáto správa popisuje udalosť, ktorú užívateľ vykonal. Informácie o udalosti sú uložené v správe v špeciálnom objekte. Aj na tento objekt musí byť implementované rozhranie na serializáciu.

5.8 Spracovanie udalostí

Pre umožnenie online editovania diagramov viacerými užívateľmi naraz je potrebné vytvoriť funkcionalitu pre spracovanie rôznych druhov udalostí, ktoré sa prenášajú po sieti. V prvom rade treba definovať množinu týchto udalostí a priradiť im systémové názvy.

Systémový názov udalosti	Popis udalosti
PREMIESTNI	Premiestnenie komponentu.
PRIDAJ_ATRIBUT	Pridanie atribútu do komponentov trieda a rozhranie.
PRIDAJ_SPOJENIE	Pridanie určitého druhu spojenia (asociácia, zovšeobecnenie, agregácia atď.) medzi komponentmi.
PRIDAT_OPERACIU	Pridanie operácie do komponentov trieda a rozhranie.
VLOZ_KOMPONENT	Vloženie komponentu do diagramu.
VYTVOR_DIAGRAM	Vytvorenie diagramu.
ZMAZ_KOMPONENT	Vymazanie komponentu.
ZMEN_NAZOV	Zmena názvu komponentu.
ZMEN_POPIS	Zmena popisu komponentu.

Každá z udalostí má špecifický systémový popis. Ten závisí od toho aké informácie je potrebné pre udalosť uchovávať, aby mohla byť zreprodukovaná u ostatných užívateľov. Tento popis má formát :

Typ_udalosti / id_komponentu / premenná1 / premenná2 /...

Typ udalosti definuje druh udalosti, ktorá sa má vykonať. Identifikátor určuje, ktorého komponentu sa udalosť týka. Ďalšie premenné sú špecifické podľa typu udalosti.

Priebeh spracovania udalosti pre komponenty potom prebieha nasledovne :

- Identifikujeme udalosť
- Nájdeme komponent, na ktorý sa udalosť má aplikovať.
- Zmeníme vlastnosti komponentu podľa informácií v premenných

5.9 Synchronizácia diagramov

Synchronizovanie bude prebiehať automaticky po prihlásení používateľa do systému, ak sú už prihlásený iní užívatelia zo skupiny. Server pošle novo prihlásenému užívateľovi zoznam udalostí, ktoré doteraz vykonali prihlásení užívatelia.

V prípade, že nastane konflikt pri editovaní v situácii, keď dvaja užívatelia editujú v rovnakom čase ten istý komponent, bude platná zmena užívateľa, ktorý má nastavenú vyššiu prioritu. Priorita bude jedinečné číslo, ktoré systém priradí každému užívateľovi. Následne systém zistí, ktorú udalosť vykonal užívateľ s vyššou prioritou a tú dodatočne odošle všetkým užívateľom. Takto sa zaručí, že stav diagramov bude rovnaký u všetkých používateľov.

5.10 Ukladanie a export diagramu v rôznych formátoch

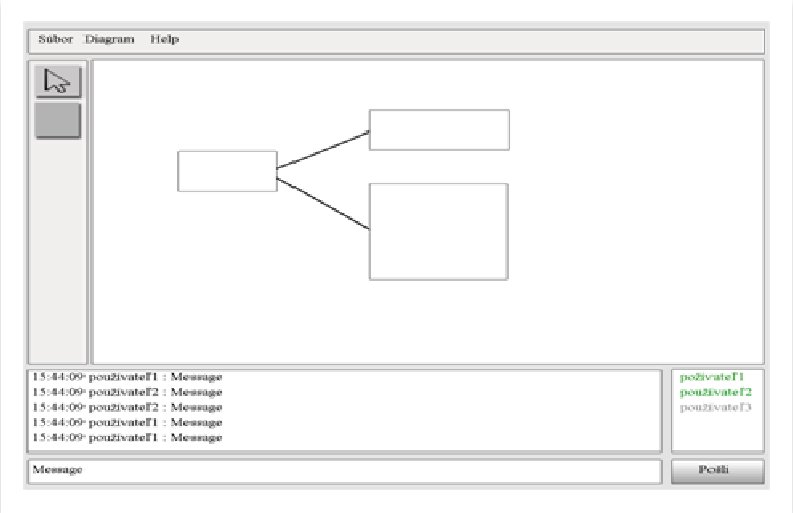
Vytvorené diagramy sa budú ukladať a načítavať pomocou serializácie do súborov s príponou .oud (Online UML Diagram File). V tomto prípade bude súbor obsahovať aj zoznam udalostí.

Druhým výstupom programu budú XML súbory generované podľa noriem organizácie OMG. Výhoda tohto výstupu bude, že diagramy bude užívateľ môcť editovať aj v iných editoroch.

Tretia forma výstupu je export do najpoužívanejších grafických formátov a to JPEG Image, Bitmap Image, PNG Image a GIF Image.

5.11 Návrh užívateľského rozhrania

V hornej časti aplikácie sa bude nachádzať štandardne menu. Pod ním v ľavom stĺpci bude panel s ikonami selektor a komponentmi diagramu. Na pravo od panela sa nachádza kresliaca plocha do ktorej sa vykresľuje diagram. Pod kresliacou plochou sa nachádza okno s konverzáciou, zoznam prihlásených používateľov, editovateľné pole pre zadanie správy a tlačidlo na jej odoslanie.



6 Implementácia

V tejto časti práce opíšem zaujímavé postupy použité pri implementácii a problémy, ktoré vznikli pri programovaní.

6.1 Tenký klient verzus hrubý klient

Program som sa rozhodla po dôkladnom zvážení implementovať formou hrubého klienta. Veľkou nevýhodou tenkého klienta (web aplikácie), by bola rozličná interpretácia kódu a štýlov medzi prehliadačmi. Tento problém by viditeľný pri umiestňovaní komponentov v diagrame, ktoré vyžaduje presnosť. Vykresľovanie čiar a ostatných grafických komponentov je v tomto prostredí oveľa náročnejšie. Väčšina takejto funkcionality sa rieši pomocou Java Scriptu, ktorý môže mať užívateľ vypnutý. Nevýhodou hrubého klienta je o niečo horšia grafika, ale v celkovom zhodnotení je hrubý klient lepšie riešenie.

6.2 Serializácia

Vzhľadom na to, že technológia serializácie je v tomto programe viackrát použitá rozhodla som sa popísať postup ako sa reálne používa.

V prvom kroku je potrebné implementovať rozhranie java.io.Serializable. Toto rozhranie som implementovala na triedy *Sprava*, *Udalost* a *Diagram*. V triedach *Sprava* a *Udalost* sa používa na prenos po sieti a v prípade triedy *Diagram* na jednoduché ukladanie a otváranie. V skutočnosti sa po sieti posiela iba inštancia triedy *Sprava*. Tá však v sebe obsahuje aj premennú typu *Udalost*, preto bolo nutné implementovať toto rozhranie aj triede *Udalost*. Ukážka implementácie rozhrania na triedu :

```
import java.io.Serializable;
public class Sprava implements Serializable{
...
```

Toto rozhranie zabezpečí transformáciu inštancie triedy do bloku bajtov. Pre prácu s týmto blokom sa používajú triedy *ObjectOutputStream* a *ObjectInputStream*. Ukážka použitia funkcií, prostredníctvom ktorých sa ukladajú a čítajú objekty typu *Sprava* :

```
private void readObject(ObjectInputStream in)throws IOException,
ClassNotFoundException{
      in.defaultReadObject();
}

public static Sprava read(InputStream rd){
```

```
      ObjectInputStream is;
      try {
            is = new ObjectInputStream( new BufferedInputStream(rd));
            return (Sprava)is.readObject();
      } catch ...
      ...
}

private void writeObject(ObjectOutputStream out) throws IOException{
      out.defaultWriteObject();
}

public void write(OutputStream wr){
      ObjectOutputStream os;
      try {
            os = new ObjectOutputStream( new
      BufferedOutputStream(wr));
            os.writeObject(this);
            os.flush();
            } catch ...
      ...
}
```

V prípade objektu diagram je ukladanie a otváranie súborov realizované pomocou tried *FileOutputStream* a *FileInputStream*.

```
FileOutputStream out = new FileOutputStream("diagram.oud");
ObjectOutputStream s = new ObjectOutputStream(out);
s.writeObject(diagram);
s.flush();

FileInputStream in = new FileInputStream("diagram.oud");
ObjectInputStream s = new ObjectInputStream(in);
Diagram diagram = (Diagram)s.readObject();
```

6.3 Vzniknuté problémy

Pri programovaní som sa najčastejšie stretla s chybou java.lang.NullPointerException. Táto výnimka sa objavuje v situácii, keď sa volá metóda na objekt, ktorý má hodnotu null. Väčšinou sa dá pomocou debugovacieho módu jednoducho zistiť o ktorú premennú sa jedná a problém vyriešiť. Po naprogramovaní prvej verzie programu sa vyskytli v programe aj vážnejšie chyby a tie sú popísané v nasledujúcich kapitolách.

6.3.1 Oneskorená aktualizácia okien užívateľov pri pridaní komponentu

Oneskorené reakcie na udalosti nastávali ak užívateľ pridal nový komponent do diagramu. Ten sa nevykreslil ostatným užívateľom v skupine, pokiaľ užívateľ nevykonal na komponente nejakú zmenu.

Problém v tomto prípade vznikol kvôli nesprávnemu postupu odchytávania udalostí. Udalosti sa odchytávali prostredníctvom funkcií mousePressed(MouseEvent e) a mouseClicked(MouseEvent e). Sú to funkcie rozhrania *MouseListener*, implementovaného na triedu *Layout*. Prostredníctvom tohto rozhrania sa odchytávajú udalosti myši. Vo funkcii mousePresse(MouseEvent e),ktorá reaguje na stlačenie myši sa kontrolovalo, či užívateľ nie je nad komponentom diagramu. V prípade, že bol, komponent sa označil. Obsah funkcie mouseClicked(MouseEvent e) sa vykoná potom, čo používateľ klikne do plochy. Obsahovala funkcionalitu pre posúvanie, premenovanie objektu a ďalšie zmeny nad objektom. Vytvárali sa v nej udalosti. Problém vznikol vo chvíli, keď je potrebné použiť *MouseListener* v triede *Vypis*. Tu treba kontrolovať či vznikla nejaká udalosť, a ak áno je potrebné ju poslať ostatným užívateľom. Na tento účel bola tiež použitá funkcia mouseClicked(MouseEvent e).

Dôsledok bol ten, že v momente keď používateľ klikol do plochy, kontrolovalo sa či vznikla udalosť, tá však vznikla neskôr ako následok kliknutia. Logickým riešením by bolo použiť na kontrolu, či vznikla udalosť funkciu mouseReleased(MouseEvent), ktorá sa zavolá až po tom čo používateľ pustí myš. Toto riešenie mi nefungovalo, tak som sa rozhodla vyriešiť problém inak. Presunula som funkcionalitu zabezpečujúcu vytváranie udalostí, z metódy mouseClicked(MouseEvent e) do metódy mousePressed(MouseEvent e) a funkcii mouseClicked(MouseEvent e) ostala kontrola toho či vznikla udalosť, ktorú treba poslať.

6.3.2 Blikanie obrazovky pri prekresľovaní

Blikanie obrazovky sa na prvý pohľad nemusí javiť ako vážny problém. U používateľov, ktorý majú rýchlejšie počítače sa nemusí ani prejaviť. Ostaným užívateľom však môže používanie programu značne znechutiť, keďže kresliaca plocha sa prekresľuje po každej udalosti, ktorá je vykonaná nad diagramom.

Tento problém som odstránila pomocou doublebuffering-u. Princíp tohto riešenia spočíva v tom, že nový stav grafickej plochy sa najprv v pamäti vykreslí do obrázka. Tento hotový obrázok sa potom dostatočne rýchlo vykresľuje do grafickej plochy, takže nevznikne neželaný efekt blikania obrazovky.

6.3.3 Udalosť na vytváranie objektu diagram

Udalosti sa vykonávajú na diagrame prostredníctvom funkcie vykonajUdalost(Diagram diagram) . Táto funkcia je súčasťou triedy *Udalost*. Inštancia triedy *Diagram*, s ktorou program pracuje a uchová v nej informácie o diagrame je súčasť triedy *Layout*. Tu vzniká problém, pri spracovaní

udalosti na vytváranie diagramu. Objekt *Diagram*, ktorý je súčasť triedy *Layout*, nie je možné vytvoriť prostredníctvom funkcie triedy *Udalost*.

Riešením bolo upraviť konštruktor a to tak, že sa v ňom nastavia iba hodnoty, ktoré sú rovnaké pre všetky druhy diagramov. Hodnota číselnej premennej, ktorá reprezentuje typ diagramu sa nastaví na číslo -1. Keďže druhy diagramov sú reprezentované len kladnými číslami, bude to v tomto prípade príznak toho, že diagram nemá definovaný typ a budeme ho považovať za nevytvorený. Ďalším krokom bolo vytvoriť funkciu vytvor(int typ), pomocou ktorej sa diagramu definuje typ. V programe sa potom na overenie toho či diagram existuje nepoužíva podmienka diagram != null, ale obmena tejto podmienky diagram.typ != -1 .

7 Nasadenie a použitie

V tejto kapitole sú zhrnuté informácie o tom čo predchádza prvému spusteniu aplikácie a stručný popis používania programu.

7.1 Vytvorenie databázy

Pred spustením programu je potrebné vytvoriť si databázu z ktorej bude program brať informácie o používateľoch. Popis toho čo má táto databáza obsahovať sa nachádza v kapitole 5.3 Databáza užívateľov. V prípade, že existuje databáza užívateľov stačí ju doplniť o spomínané polia.

7.2 Inštalácia programu

Pred prvým spustením programu je potrebné ho nainštalovať. Počas inštalácie, je pre správny chod programu nutné nastaviť databázu používateľov.

7.3 Používanie programu

Po spustení programu je potrebné sa prihlásiť. Prihlásený používateľ bude automaticky zosynchronizovaný s ostatnými používateľmi v skupine. Ak je prihlásený ako prvý môže sa prostredníctvom chatu dohodnúť s ostatnými používateľmi v skupine na druhu diagramu, ktorý vytvoria. Po zosynchronizovaní, alebo vytvorení diagramu používateľ môže pridávať komponenty do diagramu a meniť existujúce komponenty. Zmeny sa v reálnom čase prenášajú ostaným užívateľom a naopak. Po dokončení spolupráce je možné aby si užívateľ diagram uložil a ak sa skupina rozhodne v modelovaní neskôr pokračovať, stačí aby jeden z užívateľov diagram v prostredí otvoril a ostatný pripojený užívatelia v skupine budú automaticky zosynchronizovaný.

8 Možné rozšírenia

Program Online UML editor momentálne poskytuje len základnú funkcionalitu pre prácu s najpoužívanejšími UML diagramami. V budúcnosti by sa mohla funkcionalita doplniť o :

- ďalšie typy diagramov
- aktualizácie s príchodom nových verzií jazyka UML
- prácu s projektmi, ktoré obsahujú množiny diagramov, popisujúce jeden systém
- vylepšenie ovládania
- vytváranie profilov
- krajšiu grafiku

Tieto rozšírenia sú všeobecné. Idea budúcnosti programu je aj v jeho špecializovaní pre vyučovanie. Momentálne uľahčuje prácu študentom, ktorým umožňuje pracovať na vývoji diagramov z rôznych miest. Druhým podstatným rozšírením by malo byť implementovanie nástrojov, ktoré uľahčujú prácu vyučujúcemu.

Návrh systému je vytvorený tak, aby boli v budúcnosti tieto zmeny ľahko doplnené. Doplnenie rozhrania pre učiteľov by malo mať dve časti. Jedno je zamerané na prácu na cvičeniach a druhé na hodnotenie práce jednotlivých členov skupiny.

Pri práci na cvičeniach by učiteľské rozhranie malo poskytovať možnosť pripojiť sa k ľubovoľnej skupine študentov a online sledovať ich prácu. Pre toto rozšírenie by v princípe stačilo pridať do systému špeciálne role, ktoré by užívateľov rozdeľovali do dvoch skupín. Prvá rola by užívateľov obmedzovala na spoluprácu len v rámci svojej skupiny. Druhá rola pre vyučujúceho, by umožňovala pripojiť sa do ľubovoľnej skupiny. Obsahovala by aj možnosť nastaviť viditeľnosť užívateľa v zozname pripojených užívateľov. Takéto rozšírenie by umožňovalo po dohode študentov s vyučujúcim na určitom čase aj interaktívne online konzultácie zamerané na vývoj UML diagramov.

Druhá časť rozhrania zameraná na hodnotenie práce študentov by spracovávala informácie z udalostí, ktoré sú uložené vo výstupe z programu. Aplikácia Online UML editor funguje v princípe na posielaní a spracovávaní udalostí, ktoré boli v diagrame vykonané. Tie obsahujú popis udalosti, ktorá bola na diagrame vykonaná, meno užívateľa a čas, kedy bola vykonaná.

Pomocou zoznamu týchto udalostí by bolo možné vytvoriť presnú simuláciu toho ako bol diagram vytvorený. Stačilo by implementovať funkciu, ktorá by spracovala súbor s udalosťami a to tak že by postupne prechádzala zoznam udalostí a po spracovaní každej udalosti by vytvorila určité čakanie, aby bolo možné zmeny zachytiť. Kontrola toho ktorý užívateľ, vykonal ktorý krok by mohla byť doplnená do funkcie so spracovaním udalosti, ktorá by v prípade simulácie posielala meno užívateľa z udalosti napríklad do textového poľa.

Druhá možnosť, ktorá by mala pomôcť vyučujúcemu pri hodnotení projektu by mala dávať informáciu o tom, ktorý užívateľ sa akou časťou podieľal na vytváraní diagramu. Pri skupinovej práci študentov je veľmi obtiažne zistiť, ktorý člen týmu na projekte pracoval menej, alebo viac. V tomto prostredí nie je problém vytvoriť takéto rozšírenie. Kľúč je opäť v spracovaní udalostí. V prvotnom štádiu by sa podľa celkového počtu udalostí a počtov udalostí, ktoré vykonali jednotlivý študenti dal vypočítať percentuálny podiel práce jednotlivých študentov na vývoji diagramu. V ďalšom vylepšení by sa mohli pridať hodnoty náročnosti k jednotlivým typom udalostí. Prípadne vyvinúť zložitejší a presnejší algoritmus hodnotenia využívajúci informácie o každom kroku, ktorý užívateľ pri vývoji diagramu spravil.

9 Záver

Momentálne je na trhu množstvo UML editorov na vývoj diagramov. Nie je však bežným štandardom, aby tieto editory umožňovali prácu viacerých vzdialených užívateľov naraz na jednom diagrame. Cieľom mojej práce bolo navrhnúť a implementovať špeciálny editor UML diagramov pre vzdialených používateľov, ktorí sú pripojení na internet a je im umožnené súčasne pracovať na vývoji zdieľaného diagramu. Ďalším rozšírením editora je aj Mini-chat, ktorý používateľom slúži na komunikáciu.

V rozsahu bakalárskej práce sa mi podarilo implementovať základné časti návrhu:

- UML editor podporujúci vývoj Diagramu použitia
- prenos základných udalostí na diagrame v reálnom čase
- synchronizáciu neskôr pripojených používateľov k skupine
- Mini-Chat

Tieto štyri časti zachytávajú podstatu navrhovaného programu. Ostatné navrhnuté časti programu sú vecou ďalšieho vývoja Online UML Editora.

10 Zoznam použitej literatúry

[1] CADE Mark – HUMPHREY Sheil 2010. Sun Certified Enterprise Architect for Java EE Study Guid. Second edition. Prentice Hall 2010. ISBN 0131482033

[2] MEILIR PAGE-JONES. 2001. Základy objektově orientovaného návrhu v UML. Praha : Grada, 2001. 368 strán, ISBN 80-247-0210-X

[3] SCHMULLER, Joseph. 2001. Myslíme v jazyku UML . Grada, 2001. ISBN: 8024700298

[4] ŠPIGURA, M. 2007. Generátor kódu z UML diagramov: diplomová práca. Žilina : ŽILINSKÁ UNIVERZITA Fakulta riadenia a informatiky, 2007. 63 s

[5] OBJECT MANAGEMENT GROUP. 1997-2011 [online] <http://www.omg.org>