EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

KOMPARÁCIA POPULÁRNYCH MODELOVACÍCH NÁSTROJOV SÚČASNOSTI

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Kvantitatívne metódy v ekonómii

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Igor Bandurič, PhD.

Bratislava 2017 **Bc. Lukáš Pekný**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Lukáš Pekný
Študijný program:	Informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor:	3.3.24 Kvantitatívne metódy v ekonómii
Typ záverečnej práce:	Inžinierska záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Komparácia populárnych modelovacích nástrojov súčasnosti

Anotácia: Práca je zameraná na stanovenie zoznamu 4 populárnych modelovacích nástrojov (Computer Aided Software Engineering nástrojov) súčasnosti, ich charakteristiku a popis, stanovenie kritérií na porovnanie týchto nástrojov, porovnanie vybraných CASE nástrojov na základe stanovených kritérií a vyvodenie odporúčaní pre využitie príslušných nástrojov pre rôzne typy organizácií a používateľov, pre ktorých má použitie takéhoto nástroja význam. Predpokladá sa znalosť softvérového inžinierstva a viackriteriálneho rozhodovania

Vedúci:	Ing. Igor Bandurič, PhD.
Katedra:	KAI FHI - Katedra aplikovanej informatiky FHI
Vedúci katedry:	Ing. Mgr. Peter Schmidt, PhD.
Dátum zadania:	19.10.2015

Dátum schválenia: 06.11.2015

doc. Ing. Gabriela Kristová, CSc.
vedúci katedry

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: **.....**

(podpis študenta)

Poďakovanie

Touto cestou by som sa chcel poďakovať vedúcemu diplomovej práce Ing. Igorovi Banduričovi PhD., za odbornú pomoc, cenné rady a pripomienky, ktoré mi poskytol pri vypracovaní diplomovej práce.

ABSTRAKT

PEKNÝ, Lukáš: *Komparácia populárnych modelovacích nástrojov súčasnosti*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Igor Bandurič, PhD. – Bratislava: FHI EU, 2017, 68 s.

Predkladaná práca je zameraná na porovnanie troch vybraných populárnych modelovacích nástrojov. Je rozdelená do piatich kapitol a obsahuje dvadsaťtri obrázkov. Prvá kapitola je venovaná modelovaniu a jeho významu, charakteristike diagramov ako aj predstaveniu jednotlivých produktov. Po kapitole, ktorá je zameraná na ciele práce a jej metodiku, je v nasledujúcej kapitole v práci charakterizovaný jazyk UML a popísané modelovanie užívateľského prostredia. V ďalšej časti, sú v práci predstavené a charakterizované modelovacie nástroje súčasnosti. V záverečnej kapitole sú na príklade porovnávané konkrétne tri produkty, na základe niekoľkých vybraných kritérií. Výsledkom práce je zhodnotenie produktov a snaha o odporúčanie vybraných produktov jednotlivým zákazníkom.

Kľúčové slová: modelovanie, UML, Enterprise Architect, Rational Rose, Lucidchart

ABSTRACT

PEKNÝ, Lukáš: University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Thesis Supervisor: Ing. Igor Bandurič, PhD. – Bratislava: FHI EU, 2017, 68 s.

The focus of this thesis is the comparison of 3 chosen modeling tools. It is divided into 5 chapters and contains 23 pictures. The first chapter is about modeling and its importance, diagram characteristics and the presentation of individual products. After the chapter, which is aimed at presenting the goals and used methodics of the thesis, we will continue to describe the UML language and the modeling of user environment. In the next part, our work presents a characterization of the modeling tools of today. At the end, the closing chapter is about the comparison of three specific products on the basis of some chosen criteria. The output of this work is to evaluate these products and the effort to recommend the chosen products to individual customers.

Key words: modeling, UML, Enterprise Architect, Rational Rose, Lucidchart

Obsah

Zoznam obrázkov

Úvod

V dnešnej dobe, sa s veľkým nárastom informačných technológií, rýchlo zvyšujú aj požiadavky používateľov na informačné systémy. Rastú požiadavky na rozvoj informačných systémov, rovnako aj ich vývoj a nasadenie do prevádzky u zákazníka. Jedna z možností pri tvorbe informačného systému je tzv. modelom riadená tvorba informačného systému. Najdôležitejším faktom pri modelovaní informačného systému je samotný modelovací jazyk. V dnešnej dobe je najpoužívanejším modelovacím jazykom jazyk Unified Modeling Language v skratke UML. Rovnako veľký vplyv na tvorbu aplikácie má Model Driven Architecture, ktorej skratka je MDA. Modelom riadená architektúra blízko súvisí s Unified Modeling Language, ale nie sú na seba bezprostredne naviazané.

Cieľom diplomovej práce je porovnať niektoré vybrané modelovacie nástroje. V prvej kapitole sa budem venovať základným pojmom, čo je to presnejšie UML a ďalej charakteristike samotným diagramom, ktoré sa dajú vytvoriť v daných produktoch. V druhej kapitole budú prezentované dané produkty na tvorbu UML diagramov a ich vzájomné rozdiely. V nasledujúcich kapitolách sú popísané rozdiely medzi produktmi a práca s nimi. V závere je porovnanie produktom na základe daných kritérií a následne odporučenie produktu pre daného zákazníka.

1 Súčasný stav riešenej problematiky

1.1 Modelovanie a jeho význam

S narastajúcim rozmerom podpory informačných technológii (IT), pre rôznorodé činnosti podnikov, súčasne narastá množstvo závislostí a väzieb medzi jednotlivými časťami informačného systému (IS). Avšak, zároveň rastú aj požiadavky od používateľov informačných systémov, na rýchlosť zavedenia do praxe a na nové funkcie systému. Pri rozvoji podnikových informačných systémov, dochádza k nárastu nákladov na informačný systém, ak celý tento proces nie je riadený. Nesprávne riadenie môže byť nakoniec veľmi nákladné.

Jedna z možností pri tvorbe informačného systému, je modelom riadený vývoj softvéru. Hlavnými faktormi, modelom riadeného softvéru, sú modelovanie a samotné modely. Modely môžeme definovať ako abstrakcie reálneho sveta a jeho objektov v ňom. Pri použití modelom riadeného vývoja softvéru je dôležité aby bol model formálny, to znamená, že model musí mať definovanú štruktúru a sémantiku a aby bolo možné ho vytvoriť pomocou modelovacích nástrojov. K modelovaniu potrebujeme modelovací jazyk. Z praxe je známe, že najpoužívanejším a najrozšírenejším modelovacím jazyk je Unified Modeling Language (UML). Tento modelovací jazyk ale neposkytuje sémanticky jednoznačný model. Problémom je jeho komplikovaná štruktúra. Pri modelom riadenom vývoji softvéru sa často zamieňajú pojmy diagram a model. Diagram je spôsob zakreslenia modelu a model je abstrakcia entitou. (Arlow, 2006, s.78- 79)

Object Managment Group (OMG) priniesli predstavu pre modelom riadený vývoj softvéru. OMG sa prevažne venovali stanoveniu štandardov pre distribuované objektovo orientované systémy. V súčasnej dobe sa sústreďujú na modelovanie. Avšak nepriamo podporujú ako základ UML a na ňom založené ďalšie modelovacie jazyky. OMG poskytujú len špecifikácie, neposkytujú implementácie. Object Managment Group založilo, v roku 1989, jedenásť firiem, napríklad IBM, HP, Sun Microsystem a ďalšie. Kládli veľký dôraz na tvorbu heterogénnych distribuovaných objektových štandardov.

1.1.1 Modelom riadený vývoj

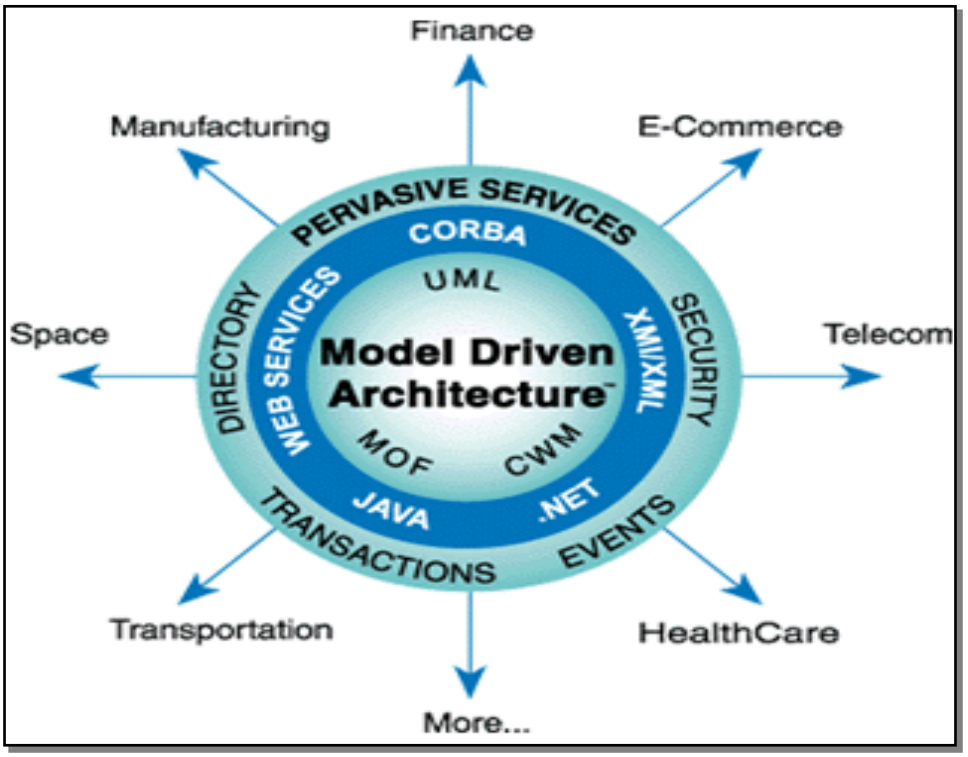
Pred modelom riadeným vývojom softvéru, bolo veľa rôznych myšlienok a spôsobov. Kombináciou a vývojom týchto myšlienok a spôsobov, sa táto technika rozvinula až do stavu, ako ho poznáme dnes. Prvou technikou môžeme označiť Computer Aided Software Engineering (CASE) nástroje. V týchto nástrojoch je možné prostredníctvom diagramov namodelovať celý systém. Z týchto modelov sa následne dala vytvoriť štruktúra softvéru alebo systému. Možnosti boli ale limitované preto postupným vývojom sa tieto nástroje ďalej vyvíjali. Táto modelovacia technika sa začala označovať ako CASE 2.0.

Neskoršou metódou bolo Unified Modeling Language. UML spočívalo v možnostiach využiť každý model a jeho voľný preklad do binárneho kódu.

V roku 2001 bol vydaný Model Driven Architecture (MDA), ktorý mal značný vplyv pri tvorbe aplikácií. V súčasnosti evidujeme viac ako 40 nástrojov, ktoré podporujú minimálne jeden z aspektov Model Driven Architecture. MDA je koncept, ktorý štandardizuje modely, ktoré sú v priebehu tvorby aplikácie vytvárané. Názov by sme mohli preložiť ako modelom riadená architektúra. Model Driven Architecture štandardizuje a následne popisuje spôsob ako sa transformuje jeden model do druhého. Koncept MDA úzko súvisí s UML avšak nie je na tento štandard bezprostredne naviazaný. (Kleppe, 2003, s.6-8)

Zásady MDA sú založené na oddelení technológii vyvíjanej platformy od biznis logiky. Platforma je komplex subsystémov alebo technológií, ktoré zaručujú logickú skupinu vzorov a rozhraní. Tieto vzory a rozhrania môžu využívať aplikácie bez potreby znalosti, ktorá a ako je funkcia poskytovaná platformou. Tým pádom je možné realizovať aplikácie vytvorené pomocou Model Driven Architecture v celej skupine otvorených platforiem ako sú napríklad J2EE, NET alebo COBRA.

Obrázok 1: Model Driven Architecture



Zdroj : http://www.omg.org/mda/

1.2 Unified Modeling Language (UML)

1.2.1 História UML a úvod do jazyka UML

Do roku 1994 bol trh s objektovo orientovanými metódami značne nejednotný. Jeden z prvých pokusov o zjednotenie bola metodika Fusion, ktorej autorom bol v roku 1994 Derek Coleman. Metodika Fusion mala rozsiahlu reklamu a silnú komerčnú podporu, ale k jej vývoju neboli pozvaní páni Grady Booch, ktorý je autorom metódy Booch, Jim Rumbaugh, autor metódy OMT3, a Ivar Jacobson (metodika Objectory). Ako odpoveď na túto iniciatívu sa prví dvaja, vyššie spomenutí páni pripojili k spoločnosti Rational Corporation, ktorá v tom čase pracovala na tvorbe jazyka UML. A presne toto spojenie ukázalo smer v budúcom vývoji a metodika Fusion bola časom zabudnutá. O rok neskôr sa pridal k Rational Corporation aj Ivar Jacobson a boli začaté rozsiahle práce na špecifikácií štandardov UML. Ich výsledkom bol jazyk UML a metodika RUP. V roku 1997 združenie OMG4 prijalo UML 1.1 ako otvorený štandard pre modelovanie vývoja softvéru. Po čase sa pridávali ďalšie prvky a diagramy až v roku 2005 získala platnosť verzia UML 2.0, ktorá priniesla najväčšie zmeny.

Unified Modeling Language je unifikovaný modelovací jazyk. Je možné o ňom povedať, že je to univerzálny jazyk pre vizuálne modelovanie systémov. Najviac je spájaný s modelovaním objektovo orientovaných softvérových systémov ale má aj oveľa rozsiahlejšie využitie, vďaka jeho rozširovacím mechanizmom. UML navrhli, aby spojili, najlepšie už existujúce postupy modelovacích techník a softvérového inžinierstva. Sám o sebe je navrhnutý takým spôsobom, aby mohol byť využitý všetkými CASE nástrojmi (Computer-Aided Software Engineering). Táto koncepcia vychádza z reality, že veľké softvérové systémy sa zvyčajne bez CASE nástrojov nezaobídu. V jazyku UML vytvorené diagramy sú ľahko pochopiteľné pre ľudí, dokonca ich vedia ľahko interpretovať aj CASE programy. Je veľmi podstatné vedieť, že UML neposkytuje žiadne druhy metodík modelovania. Avšak poniektoré aspekty metodiky je možné nájsť v každom z prvkov, z ktorých sa celý model UML skladá. Tento jazyk poskytuje iba vizuálnu syntax, ktorou si je možné pomôcť pri tvorbe vlastných modelov.

Unified Process (UP) je metodikou na rozdiel od UML. UP vie odporučiť akých zamestnancov na danú činnosť máme použiť, aké činnosti by sme mali vykonať a ktoré modely vypracovať aby sme dokázali vytvoriť funkčný model softvérového systému. (Kanisová, 2004)

Výhodou UML je, že nie je viazaný na žiadny životný cyklus alebo špecifickú metodiku. Je možné ho využiť spoločne so všetkými existujúcimi metódami. Unified Process je metodika, ktorá využíva UML jazyk, ako vlastnú syntax pri vizuálnom modelovaní. Tým pádom je možné považovať UP za uprednostňovanú metodiku pri používaní UML jazyka z dôvodu, že tento jazyk je pre ňu najlepšie prispôsobený. UML vie poskytnúť, a aj poskytuje, podporu pre vizuálne modelovanie ale aj pre iné metódy resp. metodiky. Cieľom jazyka UML a metodiky UP je podpora všetkých najlepších známych postupov a metód pri softvérovom inžinierstve, z overených skúseností za posledné roky. Kvôli tomu boli v UML jazyku ale aj v metodike UP unifikované všetky pokusy o tvorbu jazykov pre vizuálne modelovanie a proces softvérového inžinierstva.

Je možné povedať že UML je súhrn, hlavne, grafických notácií k vyjadreniu návrhových a analytických modelov. Rovnako umožňuje modelovať jednoduché ale aj zložité aplikácie pomocou rovnakej formálnej syntaxe a tým pádom je možné výsledky práce zdieľať s kolegami či inými návrhármi. Tieto modely sú ľahko pochopiteľné aj pre obstarávateľa aplikácie. To dovoľuje kvalitné pochopenie požiadaviek užívateľa na vytváraný systém. Ani jeden diagram nezobrazuje navrhovaný systém ako celok, ale vždy sa sústreďuje len na jeden pohľad na daný systém. UML je taktiež jazykom pre vizualizácie, špecifikácie, dokumentáciu a stavbu softvérových systémov. V dnešnej dobe je známych veľa metodických postupov, ktoré vychádzajú práve z modelovacích techník UML a rozširujú tieto modelovacie techniky o ich vlastné spoľahlivé postupy, techniky a diagramy (procesné modelovanie, špecifikácia požiadaviek). Jedným z najznámejších, ktoré je možné uviesť, je skupina metodík RUP od firmy Rational a jej voľne šíriteľná verzia UP – Unified Process alebo metodika od firmy Select Business Solutions – Select Perspective. (Ambler, 2005)

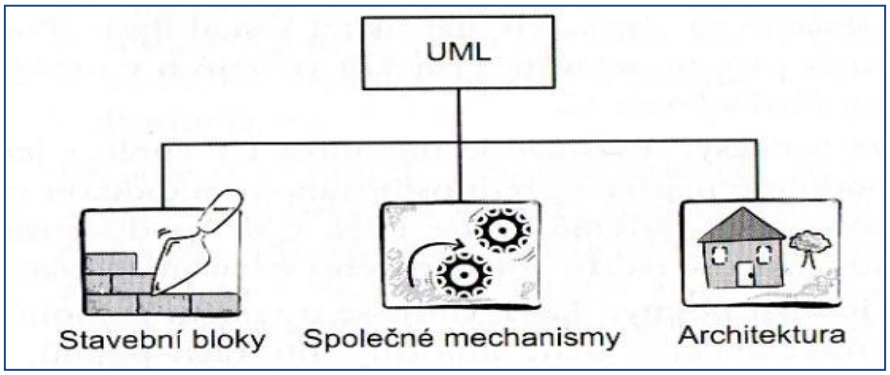
V minulosti súperilo viacero metodík a jazykov pre vizuálne modelovanie, viac ako polovicu trhu v tom čase si rozdeľovali metóda OMT (Object Modeling Technique Rumbauch) a metóda Booch. Veľmi veľa autorov v tej dobe prehlasovalo, že oni majú svoje vlastné „metódy“, no pritom mali len syntax pre vizuálne modelovanie. Až v roku 1996 vytvorila skupina Object Managment Group špecifikáciu Request For Proposal (RFP), ktorá je postupom pre vytváranie štandardov v sieti internet a súčasne požiadaviek na označenie pre takto vytvorený štandard. Unified Modeling Language v ňom vzniklo ako štandard, ale až neskôr v roku 1997 Object Managment Group prijalo jazyk UML. Jazyk UML bol v roku 2000 značne rozšírený o sémantiku akcií. Sémantika slúži

k charakteristike chovania množín primitívnych akcií. Akčné jazyky spoločne so sémantikou nám dovoľuje podrobnejší opis prvkov, ktoré súvisia s chovaním UML modelov priamo v danom modeli UML. (Fowler, 2009)

1.3 Štruktúra jazyka UML

Keď sa pozrieme na štruktúru jazyka UML zistíme že ma veľa funkcií. UML štruktúru je možné rozdeliť do troch častí. Prvou časťou sú stavebné bloky, kde patria diagramy, relácie a všetky typy modelov. Druhou sú spoločné mechanizmy, vďaka ktorým sú v jazyku UML dosiahnuté dané ciele. Poslednou časťou je architektúra, ktorou sa rozumie pohľad v jazyku UML na architektúru navrhovaného systému. Štruktúru jazyka UML je zobrazená na obrázku 2.

Obrázok 2: UML štruktúra



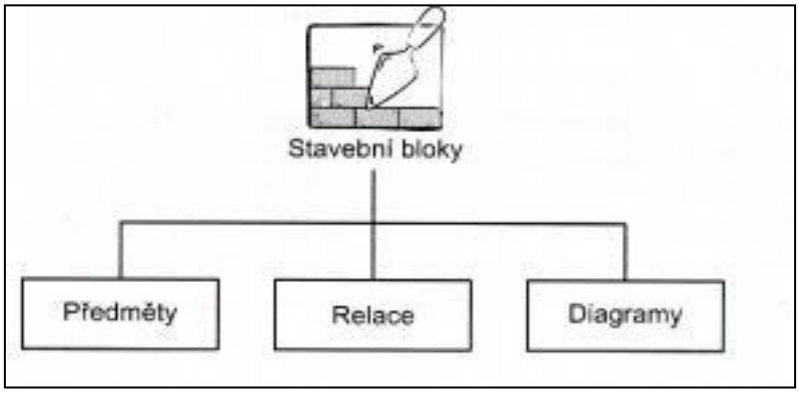
Zdroj: (Arlow, 2007)

UML jazyk sa skladá z viacerých stavebných blokov, ktorými sú :

- Predmety (Things) – sú definované ako samotné prvky modelu
- Relácie (Relationship) – spájajú predmety a definujú, ako spolu dva alebo viac predmetov súvisí
- Diagramy (Diagrams) – zobrazujú čo bude daný systém robiť (analytické diagramy), a ako to bude robiť (návrhové diagramy), je to pohľad na model v UML (The Unified Modeling Language User Guide) (Ambler, 2005, str.12)

Stavebné bloky UML sú zobrazené na Obrázku 3.

Obrázok 3: Stavebné bloky UML



Zdroj: (Arlow, 2007)

1.3.1 Predmety

Predmet sa skladá z elementov daného modelu. Elementy sú ďalej členené do viacerých navzájom odlišných podskupín.

Prvým typom skupiny sú takzvané štrukturálne abstrakcie UML modelu, ako napríklad programové triedy, objektové rozhranie, aplikácie, uzly, komponenty či prípady použitia. V UML diagramoch sú zobrazované štrukturálne abstrakcie ako rôzne, zvyčajne ploché útvary. Tieto útvary sú vždy uzavreté. Ide napríklad o obdĺžniky, kružnice, elipsy či dokonca aj jednoduché na prvý pohľad vyzerajúce trojrozmerné útvary ako sú kocky alebo iné kvádre. Keď má byť diagram UML zmysluplný, mal by obsahovať minimálne dve štrukturálne abstrakcie. Pri použití iba jednej abstrakcie stráca celé modelovanie svoj hlavný význam. (Arlow, 2007)

Medzi predmety, okrem štrukturálnych abstrakcií, patria aj „správania“. Tieto správania v diagrame UML prezentujú interakcie, ktoré definujú komunikáciu medzi jednotlivými objektmi. S využitím správania je možné modelovať stavový automat, pri ktorom sa jednotlivé stavy definujú pomocou udalostí, prechodov a samotných aktivít. (Arlow, 2007)

V diagramoch UML sa správanie zvyčajne znázorňuje pomocou rôzne vytvorených a rôzne zakončených šípok alebo spojovacích čiar. K týmto predmetom patrí aj takzvané zoskupenie, ktoré podľa potrieb modelu graficky zoskupí diagramové časti na nižšej úrovni. Zvyčajne hovoríme o takzvaných balíkov, ktoré majú tvar kancelárskej zložky, ktorá ma popis umiestnený v ľavej hornej časti obdĺžnika (toto zobrazenie zoskupenia sa ale môže v rôznych prostrediach líšiť). Zoskupenie nie je v súčasnosti v niektorých ďalej

popisovaných aplikáciách použiteľné (nie je možné vytvoriť hierarchické diagramy), a to spôsobuje tvorcom rozsiahlejším diagramom značné obmedzenia. (Arlow, 2007)

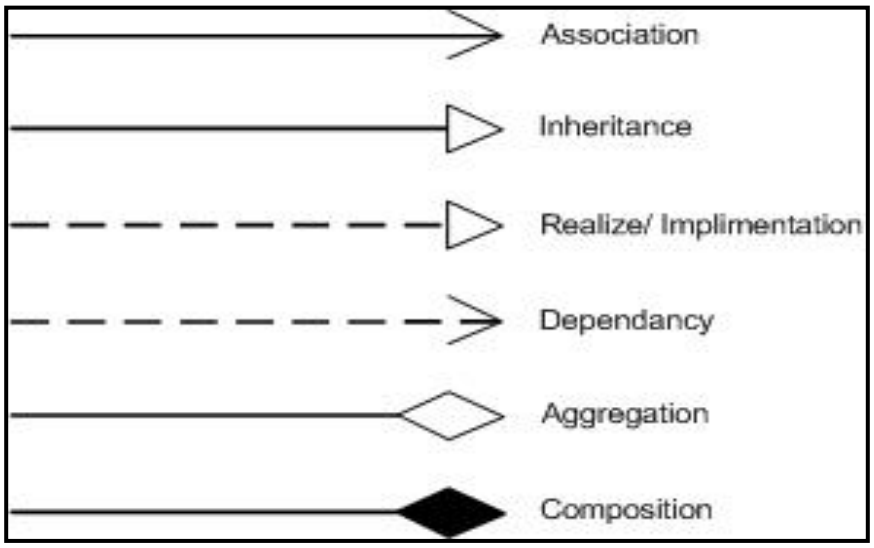
Jednoduchým, a aj posledným typom predmetov, je poznámka. Poznámka bližšie definuje správanie a vlastnosti elementov v UML diagrame. Poznámky sú graficky na všetkých typoch UML diagramov zvyčajne vyznačené ako žlto vyplnený obdĺžnik, ktorý ma zahnutý roh. (Arlow, 2007)

1.3.2 Relácie

Z dôvodu že v UML diagramoch je nutnosťou dané predmety spolu poprepájať rôznymi spôsobmi, sú v jazyku UML definované aj relácie. Je možné povedať že relácie sú vzťahy medzi predmetmi. Ďalšou podstatnou časťou modelovania v UML jazyku je pochopenie presnej sémantiky v rôznych typoch relácií (Obrázok 4). V jazyku UML rozoznávame tieto štyri podtypy relácií:

- Asociácie – Vďaka združeniu sa modeluje všeobecný vzájomný vzťah, ktorý je však v UML diagrame presne zadefinovaný. Poznáme ale aj špeciálne asociácie takzvané agregácie a kompozície, ktoré sa prevažne používajú v objektovo orientovaných databázach a jazykoch.
- Závislosť – Závislosť sa používa ak v danom predmete zmena spôsobí aj zmenu v inom predmete, alebo danému predmetu poskytne určitú informáciu
- Generalizácia – S použitím generalizácie vieme modelovať stav kde sa nachádza predmet ktorý je špecializáciou iného predmetu. Generalizácia je používaná pri objektovo orientovaných jazykoch. Táto relácia je zvyčajne implementovaná pomocou dedičnosti.
- Realizácia – Tento pojem predstavuje druh vzťahu pri ktorom daný predmet predstavuje dohodu za ktorej splnenie je zodpovedný iný predmet. Realizácia sa vytvára pri objektovo orientovaných jazykoch vďaka rozhraniu (interface). Ale len za predpokladu že objektovo orientovaný jazyk tieto rozhrania aj podporuje. (Schmuller, 2004)

Obrázok 4: Grafické znázornenie relácií UML



Zdroj: https://sites.google.com/site/sureshdevang/uml

1.3.3 Diagramy UML

Computer Aided Software Engineering (CASE) nástroje, sú to nástroje, ktoré sú založené na jazyku UML. V CASE nástrojoch je každá novo vytvorená relácia alebo predmet automatický vkladaný do tvoreného modelu. Modelom teda rozumieme repozitár všetkých predmetov a k ním vytvorených relácií k tomu, aby bolo definované správanie systému. Diagram nie je model, je to pohľad na systém. Relácie ale aj predmety vieme z diagramov vymazať no v samotnom modeli môžu aj naďalej existovať.

Poznáme trinásť odlišných typov diagramov. Je možné ich rozdeliť na diagramy ktoré majú statickú štruktúru, takzvané statické modely, a na tie ktoré majú dynamickú štruktúru, čiže dynamické modely. Dynamické modely zachytávajú spôsob, ako predmety na seba vzájomne pôsobia. Statické modely zachytávajú asociácie medzi predmetmi a samotné predmety. Nie je žiadne vopred stanovené poradie, ktoré by určovalo v akom poradí by sa mali dané diagramy tvoriť. Zvyčajne sa začína diagramom prípadov použitia (Use Case), ktorý definuje rozsiahlosť platnosti systému, ktorý navrhujeme. V reálnom svete často pracujeme súčasne s viacerými odlišnými diagramami naraz. Diagram nie je len pohľad na model ale je aj základným mechanizmom pre určovanie nových informácií do už vytvoreného modelu.

Využitie UML diagramov vo vývojových fázach je dané vývojovou metodikou. Je možné povedať, že v rámci analýzy sú využívané diagram aktivít, modely rokovania, diagram tried, scenár činnosti a stavový diagram. Pri návrhových fázach je používaný

diagram spolupráce, diagram tried, diagram aktivít, diagram nasadenia a diagram komponentov. V implementačnej časti je používaný diagram tried, diagram nasadenia a diagram komponentov.

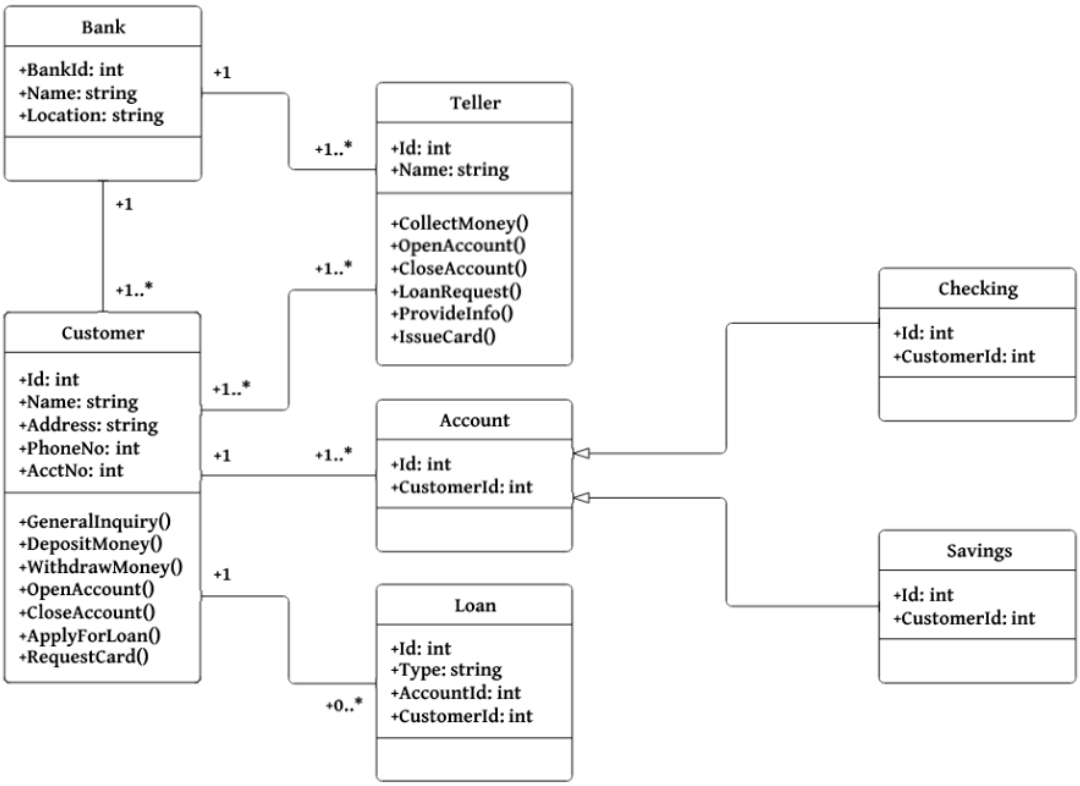
Najčastejšie je využívaných týchto osem UML diagramov: diagram objektov a tried, model rokovaní, scenáre činností, diagram spoluprác, stavové diagramy, diagram aktivít, diagram komponentov a diagram nasadenia.

Diagram objektov a tried (Object Diagram, Class Diagram), opisuje statickú štruktúru systému, stvárňuje dátový model od konceptu až po jeho implementáciu. Tento diagram je zobrazený na obrázku 5.

Model rokovaní (Diagram prípadov použitia – Use Case Diagram), dokumentuje rôzne prípady použitia vytváraného systému – sú to udalosti, na ktoré by mal systém reagovať. Taktiež nám slúži pre základné určenie hraníc medzi systémom a jeho okolím, príklad znázornený na Obrázku 6. Model má tieto komponenty:

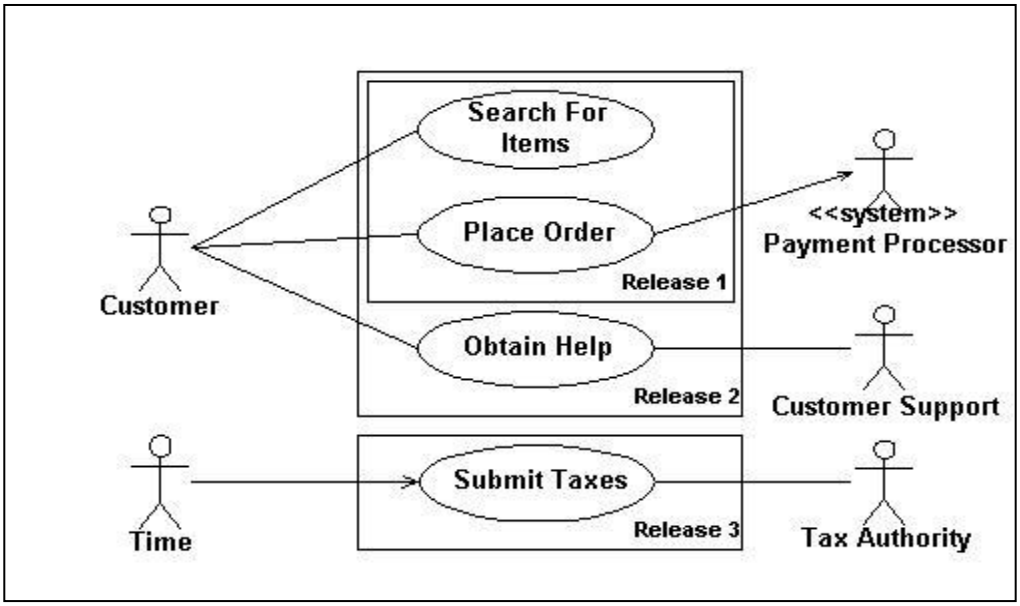
- Aktér (Actor) – môže to byť spolupracujúci systém alebo používateľská rola
- Hranice systému (System boundry) – definovanie hraníc systému
- Prípady použitia (Use Case) – popísané udalosti, na ktoré by mal systém reagovať
- Komunikácia (Communication)– väzba medzi prípadmi použitia a aktérmi (aktéri komunikujú so systémom na danom prípade použitia). (Schmuller,2004)

Obrázok 5: Class Diagram



Zdroj: https://www.lucidchart.com/pages/uml/class-diagram

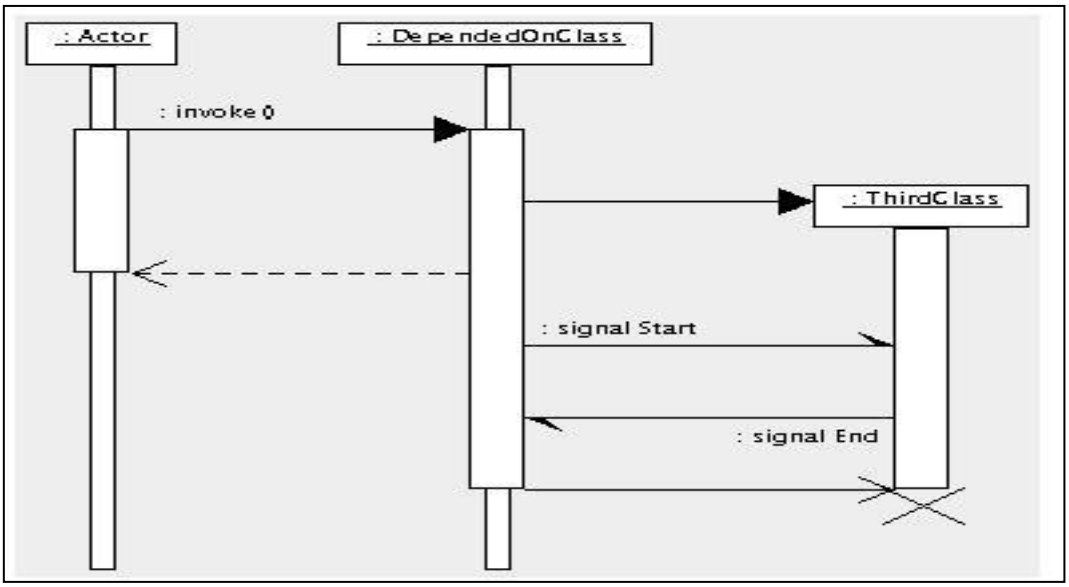
Obrázok 6: Use Case Diagrams



Zdroj: http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

Scenáre činností (Sequence Diagram – Diagram postupností/sekvenčný diagram) definuje scenár priebehu danej činnosti v určitom systéme. Zapisuje, ako spolupracujú účastníci na scenári činnosti. Kladie sa pritom ale dôraz na časový aspekt komunikácie. Dokumentujú sa správy a objekty, ktoré si dané objekty posielajú pri riešení scenára. Tieto diagramy sú vhodné pri popise scenára komunikácie s používateľmi. Príklad je zobrazený na Obrázku 7.

Obrázok 7: Sequence Diagram

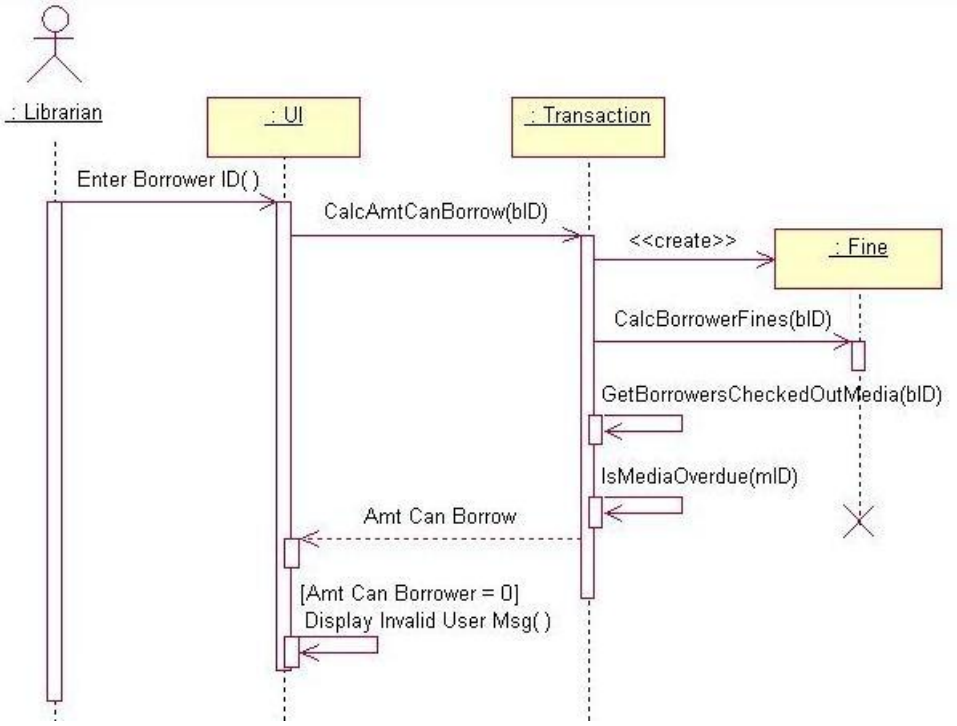


Zdroj: http://argouml-stats.tigris.org/documentation/manual-0.22/ch19.html

Diagram spoluprác (Collaboration Diagram) zbiera komunikáciu medzi objektmi, ktoré spolupracujú. Možné je to prirovnať k scenárom činností, ktoré dokumentujú spoluprácu objektov. Rozdiel je pri pohľade na komunikačný aspekt, na ktorý je kladený oveľa väčší dôraz (čas je vyjadrený číslami). Popisujú správy a objekty, ktoré si objekty posielajú pri určitom riešení danej problematiky. Diagramy spoluprác sú dobre využiteľné pri popisoch spolupráce objektov pri návrhu spolupráce.

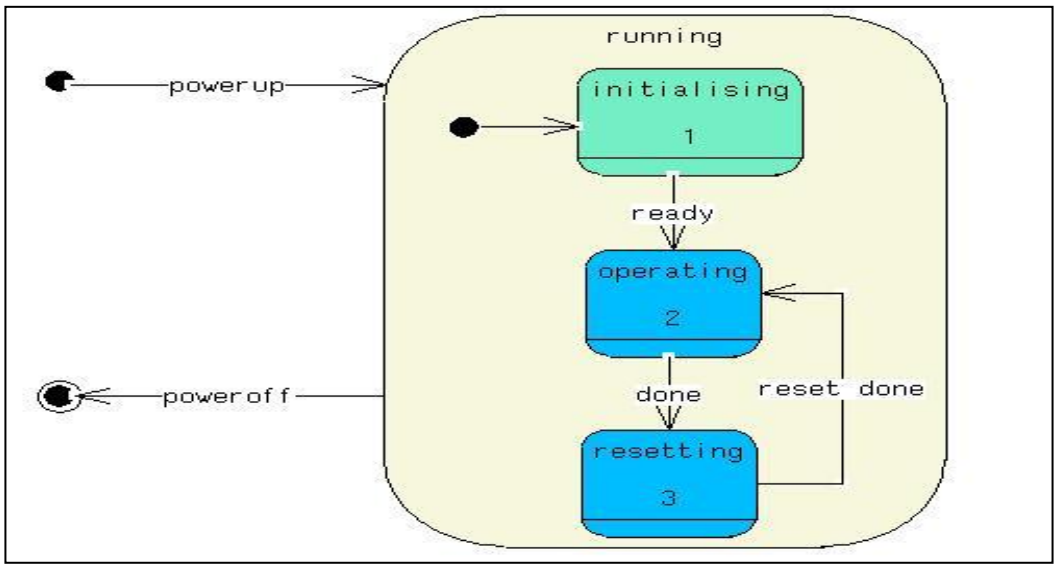
Stavové diagramy (Statechart Diagrams) charakterizujú dynamické správanie systémov alebo objektov. Stavový diagram slúži hlavne k popisu dynamiky systému. Popisuje možné stavy a prechody medzi stavmi, udalosti ktoré sú iniciátormi týchto prechodov, podmienky prechodov a všetko čo s prechodmi súvisí. Tieto diagramy môžeme využiť na charakteristiku dynamiky objektu, za predpokladu že daný objekt má rozpoznateľné stavy. Ukážka je zobrazená na Obrázku 9. (Russell,2006)

Obrázok 8: Collaboration Diagram



Zdroj: http://www.codemag.com/Article/0205051

Obrázok 9: Statechart Diagram



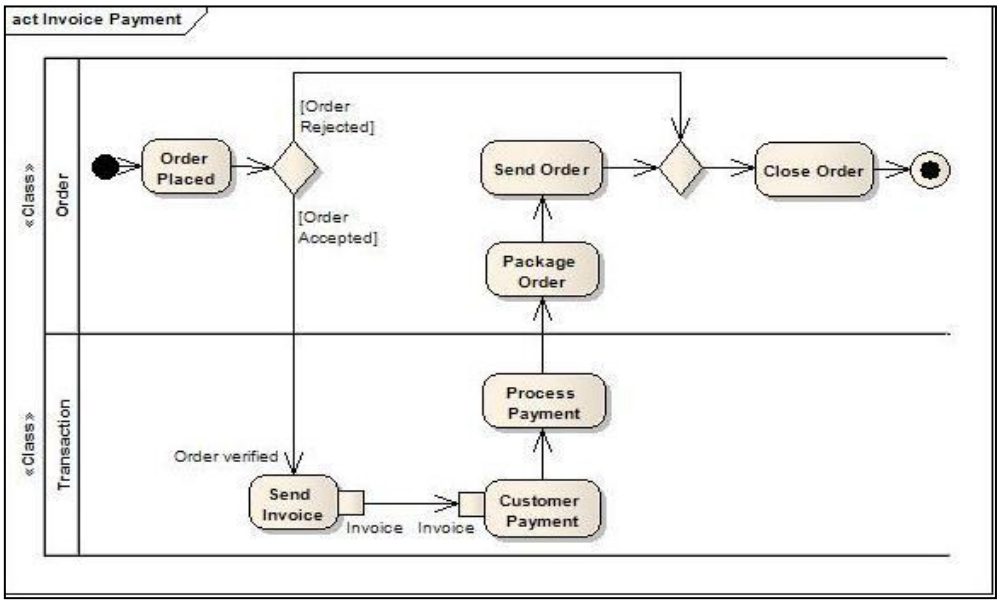
Zdroj: http://www.threesl.com/pages/reference/diagrams/state-chart-diagram.php

Diagram aktivít (Activity Diagram) popisuje celý priebeh aktivít činností alebo procesov. Pri tomto diagrame na rozdiel od stavového diagramu prechod iniciovaný ukončením danej aktivity. Prechody sú ukončene až po ukončení akcie, takže sú synchrónne. Tieto diagramy používame pri popisoch metód, tried alebo aj prípadov použitia. Čiastočne zastupujú za neexistujúce diagramy dátových tokov v UML. Sú špecifické nakoľko môžu obsahovať symbol „*rozhodnutie*“. Príklad je zobrazený na obrázku 10.

Diagram komponentov (Component Diagram) rozdeľuje systém na komponenty (celky) a popisuje význam jednotlivých celkov. Ďalej sa popisujú aj typy celkov, ktoré sú definované v diagramoch nasadenia. Celky môžeme mať vnorené do ďalších komponentov. Pri definovaní vzťahov medzi celkami môžeme používať rozhranie (Interface). (Schmuller,2004)

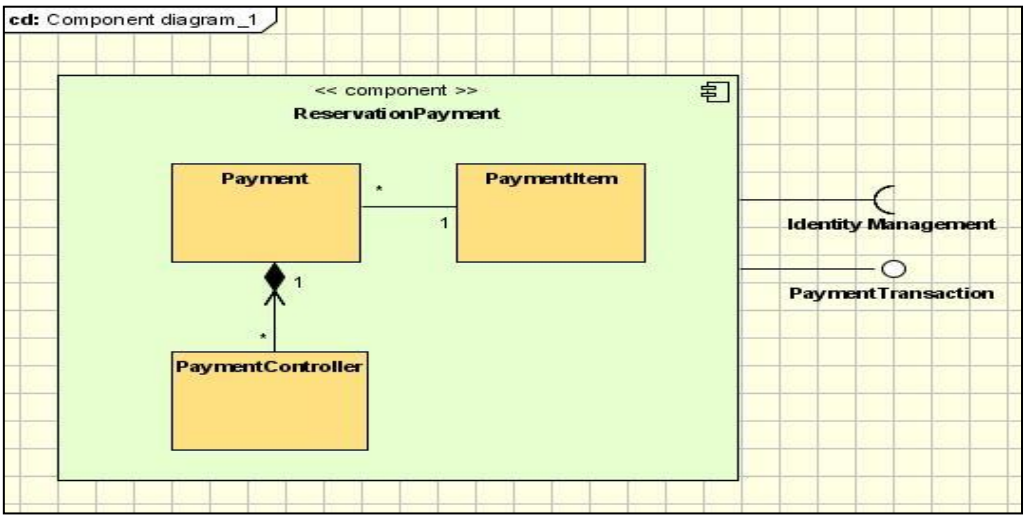
Diagram nasadenia (Deployment Diagram) definuje presné umiestnenie komponentov na jednotlivé výpočtové uzly systému. Nájdeme v ňom popisy fyzického rozmiestnenia elementov informačného systému. Elementy spoločne s uzlami sú označené podobne ako triedy a objekty. Definujú všetky potrebné väzby medzi uzlami, prípadne použitý interface. Sú v ňom obsiahnuté len komponenty potrebné pre funkčnosť aplikácie, ostatné komponenty pre zostavenie a preklad sú už uvedené v komponentovom diagrame. Príklad je zobrazený na Obrázku 12. (Arlow,2007)

Obrázok 10: Activity diagram



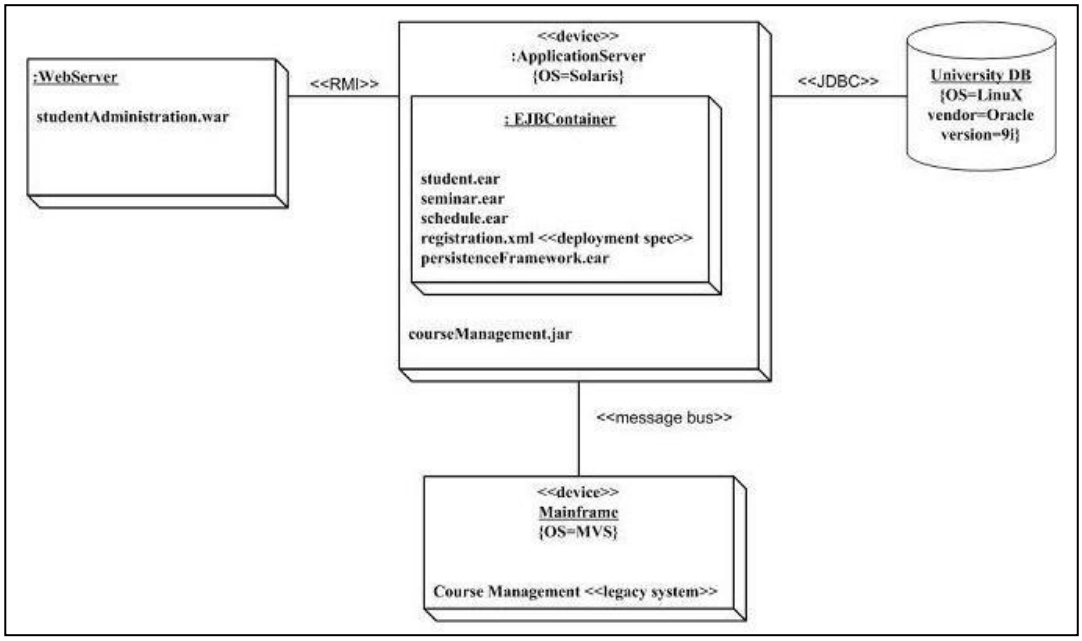
Zdroj: http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/353/Enterprise-Architect-for-Business-Analysts.aspx

Obrázok 11: Component Diagram



Zdroj:
http://www.gentleware.com/fileadmin/media/archives/userguides/poseidon_users_guide/componentdiagram.html

Obrázok 12: Deployment Diagram



Zdroj:http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

Je možné povedať že Unified Modeling Language je všestrannou notáciou pre univerzálne využitie pri obchodného modelovania až po detailné návrhy informačných systémov. Notácie, ktoré by vyčerpávajúcimi spôsobmi pokryli tak veľké spektrum potrieb

by boli príliš komplikované. Preto jazyk UML zadefinoval len základnú časť (UML core) a tým pádom dovoľuje rozšíriť notácie podľa potreby. Základnú sémantiku jazyka UML je možné doplniť štandardným obmedzením, ktoré ale bude dovoľovať zmeny interpretácie elementov:

- Príznak (Tag) – príznaky nám umožňujú k prvkom v diagramoch pridávať ďalšie atribúty
- Obmedzenie (Constraint) – obmedzenia umožňujú popisovať obmedzenia pri atribútoch
- Stereotyp (Stereotyp) – stereotypy dovoľujú triedenie elementov (Ambler, 2005)

1.4 Programové nástroje s podporou modelovania v UML

Keď si chceme vybrať vhodný nástroj pre modelovanie v jazyku UML, je nutné zohľadniť množstvo nástrojov. Každý z ponúkaných nástrojov má svoje určité výhody a nevýhody, prípadne sú vhodné len na určitú oblasť použitia. Môžeme sa stretnúť s nástrojmi, ktoré dokonca nemajú implementovaný support UML 2 alebo bol ich vývoj zastavený. Z týchto dôvodov je uvedený popis niektorých nástrojov, s ktorými je možné sa stretnúť pri modelovaní.

1.4.1 ARIS UML Designer

Verzia: ARIS Expes 2.0
Autor: IDS Scheer AG
Downolad: <http://download.ariscommunity.com/express.jnlp>

Tento nástroj je prevažne zameraný na poskytovanie supportu procesom. Aris UML Designer spojuje modelovanie biznis procesov a vývoj softvéru, za účelom zjednodušenia spolupráce medzi určitými oddeleniami. Výsledkom je integrovaný, priebežný prístup k vytváraniu podnikových aplikácií, všetko vrátane analýzy až po celkový návrh systému.

Práve Aris UML Designer je prvým nástrojom, ktorý je možné použiť na modelovanie bussines procesov a zároveň pri vytváraní softvéru. Aris podporuje komplet celý proces vývoja softvéru. Tým pádom nie sú potrebné akékoľvek aktivity mimo tohto procesu. Aris umožňuje všetkým procesným a UML modelárom prácu s jedným integrovaným softvérovým produktom. Používatelia dostanú prístup k informáciám z procesného modelu. A obsah UML cez ľubovoľný webový prehliadač.

Vlastnosti Aris UML Desiger sú:

- Dokáže integrovať biznis procesy a UML do projektu tvorby softvéru.
- Jedna rozsiahla implementácia procesov vývoja obchodne orientovaného softvéru.
- Jednoduché vytvorenie a komunikácia vývojovej dokumentácie.
- Prepája orientovaný návrh na objekt a generovanie kódu.
- Podporuje všetky druhy UML diagramov.

1.4.2 ArgoUML

Verzia: v0.28
Autor: University of California
Project leader: Linus Tolke
Downolad: <http://argouml.tigris.org/>

Tento nástroj je možné využiť na veľmi jednoduché grafické návrhové prostredie, ktoré podporuje vývoj, návrh a dokumentáciu objektovo orientovaných aplikácií. Argo UML spadá pod licenciu General Public Licence (GPL). Je vyvíjaný širokou komunitou vývojárov. Tento nástroj je celý napísaný v jazyku Java, vďaka čomu je Argo multiplatformovým nástrojom, čiže ho môžeme používať na ľubovoľnom počítači a operačnom systéme. Nevýhodou je, že nepodporuje generovanie kódu z dynamických diagramov, ale vieme vygenerovať kód z diagramov tried.
Argo podporuje profily Java, C++, UML 1.4 a tieto diagramy:

- Diagram tried
- Diagram stavov
- Diagram aktivít
- Diagram prípadov použitia
- Diagram Spolupráce
- Diagram Nasadenia
- Sekvenčný Diagram

(tigris.org, 2010)

1.4.3 Rational Modeler

Verzia: 7.6
Autor: Rational Software
Downolad: <http://www.ibm.com/developerworks/downloads/r/modeler/>

Je bezplatným nástrojom na návrh softvéru, založený na jazyku UML 2.1. Rational Modeler implementuje modelovanie na základe UML na vizualizáciu, špecifikáciu a dokumentáciu návrhov softvéru. Za základné vydanie produktu Rational Modeler sa neplatí.

Vlastnosti uvádzané výrobcom:

- Zlepšuje produktivitu a urýchľuje vývojové cykly pomocou vizuálneho prostredia návrhu.
- Pomáha pri eliminácii chýb vďaka automatickým dokumentačným funkciám, ktoré sú možné spustiť jediným tlačidlom.
- Dokáže pracovať s návrhmi v štandardizovanom prostredí UML 2.1.
- Sprostredkováva ciele pri návrhu prostredníctvom spoločného grafického jazyka.
- Poskytuje diagramy na zostavovanie jednotných modelov, ktoré vám pomôžu vytvárať komplexné a správne aplikácie.
- Poskytuje výkonné a prispôsobiteľné prostredie návrhu, ktoré poskytuje používateľom sústrediť sa na svoje odborné sféry pomocou špecifickej terminológie a zápisu.
- Možnosť jednoduchej aktualizácie na modelmi riadené vývojové prostredie IBM Rational Rhapsody.
- Verzia Modeler Corporate Edition sa integruje spolu s nástrojmi na správu verziií, vrátane IBM Rational Synergy a IBM Rational ClearCase.
- Podporuje operačný systém Windows. (IBM, 2010)

1.4.4 Toad Data Modeler

Verzia: 2.25

Autor: Quest Software

Downolad: <http://www.quest.com/toad-data-modeler/>

Toad Data Modeler je produkt, ktorý sa zameriava na oblasť kvalitného reverzného inžinierstva a návrh databáz. Keď navrhujeme databázu, musíme zohľadniť určité špecifiká danej databázy ako sú napríklad domény, integrita, triggre a podobne. Tento nástroj umožňuje použiť reverzné inžinierstvo. To znamená pri existujúcich databázach generovať SQL/DDL skripty, alebo podrobné HTML reporty. Toad Data Modeler podporuje:

- fyzický aj logický model,
- konverziu z fyzického modelu do logického a opačne,
- verifikáciu modelu,
- generovanie SQL,
- generovanie skriptov pre Oracle 10g, 9 a MS SQL Server 2008 a 2005,
- porovnávanie modelov, ich aktualizáciu a spájanie HTML s RTF reportmi.

2 Ciele a metodika práce

2.1 Ciele práce

Hlavným cieľom tejto práce je porovnanie produktov Enterprise Architect, Rational Rose a Lucidchart, na základe vopred stanovených kritérií.

Pre splnenie hlavného cieľa je potrebné si určiť niekoľko čiastkových cieľov. Prvým čiastkovým cieľom je získanie licenčných zmlúv a inštalácia vybraných produktov. Ďalším čiastkovým cieľom, ktorý výrazne dopomôže k splneniu hlavného cieľa, je oboznámenie sa s jednotlivými produktmi a ich praktické použitie.

Posledným cieľom práce je po subjektívnej skúsenosti s použitím daných produktov snaha o odporúčanie správneho produktu pre zvolené modelové príklady.

2.2 Metodika práce

Po zadaní témy a po určení hlavných cieľov, bolo dôležité oboznámiť sa s danou problematikou. Preto bolo prvým krokom zozbieranie potrebnej literatúry, ktorá sa venuje UML a modelovacím nástrojom. Štúdiom tejto literatúry, bol nadobudnutý teoretický základ o danej problematike. Pre splnenie zadaných cieľov bolo jedným z kľúčovým krokom aj pochopenie významu samotného modelovania.

Ďalším veľmi dôležitým krokom bolo zaobstaranie licenčných zmlúv na používanie vybraných produktov. Pri Enterprise Architecte mi bola licenčná zmluva poskytnutá Ekonomickou univerzitou v Bratislave. Pri ďalšom produkte, ktorým je Rational Rose, bolo získanie licenčnej zmluvy zložitejšie. Kvôli zaobstaraniu tejto licenčnej zmluvy, som sa skontaktoval so spoločnosťou IBM, od ktorej je tento produkt. Po objasnení dôvodov využitia licenčnej zmluvy ich produktu, mi spoločnosť IBM poskytla licenciu na jeden mesiac bezplatne. Tretím produktom je Lucidchart. V tomto prípade nebolo potrebné získavanie licenčnej zmluvy, nakoľko je tento produkt po registrácií voľne dostupný. Registráciou na stránke spoločnosti je poskytnutá iba základná verzia, rozšírené verzie sú mesačne spoplatnené.

Po inštalácií vybraných produktov bolo ďalším krokom oboznámenie sa s princípmi fungovania a používania týchto produktov. Tento krok bol dosiahnutý štúdiom popisov produktov. Pre bližšie porovnanie vybraných produktov som v každom z nich vytvoril class diagram, ktorý znázorňuje fungovanie letiska. Postup tvorby diagramov

v jednotlivých produktov som vyhodnotil. Posledným krokom bolo celkové zhodnotenie a porovnanie produktov. Pre lepšie porovnávanie týchto troch produktov, bolo zvolených päť kritérií, na základe ktorých bolo možné vyhodnotiť využitie pre tri vybrané modelové príklady. Medzi vybrané kritéria porovnávania patrí aj cena týchto produktov. Práve cena je jedným z rozhodujúcich faktorov pri výbere produktov. Ďalšími vybranými kritériami sú dostupnosť, kompatibilita s operačnými systémami, funkcionalita a na koniec celkový user experience.

Záver práce je venovaný celkovému zhodnoteniu vybraných produktov, ktorý bol podmienený subjektívnymi skúsenosťami pri práci s nimi a vybranými piatimi kritériami, ako aj snahe odporučiť jednotlivé produkty vybraným modelovým príkladom.

3 Modelovanie používateľského rozhrania

Po zoznámení s diagramami, ktoré UML podporuje je jasné, že diagramy ktoré umožňujú priame modelovanie user interface (používateľské rozhranie) nie sú v jazyku UML podporované. V predošlej časti, kde sme si predstavili niekoľko nástrojov, ktoré podporujú UML sme si mohli všimnúť, že medzi podporovanými vlastnosťami sa rovnako nevyskytuje podpora modelovania user interface. Preto by sa zdalo že jazyk UML je pre modelovanie user interface nedostačujúci. Avšak nástroje s podporou jazyka UML obsahujú aj niektoré neštandardné diagramy. Medzi takéto diagramy môžeme zaradiť aj diagramy používateľského rozhrania.

3.1 Požiadavky na používateľské rozhranie

Pri vývoji používateľského rozhrania, tak ako pri každej inej programovanej aplikácii, je dobre začať s vopred definovanými požiadavkami. Požiadavky vieme rozdeliť do dvoch skupín. Prvou sú funkčné požiadavky, ktoré špecifikujú požiadavky na funkčnosť systému. Nefunkčné požiadavky, špecifikujú isté vlastnosti systému, prípadne podmienky obmedzujúce fungovanie systému (Kanisová, a kol., 2007)

Na zaznačenie funkčných požiadaviek sa v jazyku UML používajú diagramy prípadov použitia. Tieto diagramy dovoľujú zachytiť, ako užívateľ bude daný systém využívať a pracovať s ním. Formou textu vieme popísať nefunkčné požiadavky, nakoľko pre tieto požiadavky nemáme v jazyku UML žiadnu grafickú notáciu, ale je určená pre poznámky, čo sa zvyčajne používa aj na popis nefunkčných požiadaviek. Ako ukážkový príklad pre overenie možností si určíme zdanlivý informačný systém pre sklad. Na jeho user interface budú kladené nasledujúce požiadavky:

- zabezpečené pripojenie klient server,
- klient spustiteľný v ľubovoľnom internetovom prehliadači,
- server pripojený v lokálnej sieti,
- prihlásenie skladníka,
- prihlásenie pre obchodníka,
- vytváranie dokumentov a následne ich tlač a export do súboru PDF,
- príjem tovaru,
- výdaj tovaru,
- objednávka tovaru,

- vytvorenie nového tovaru,
- používateľské nastavenie,
- odhlásenie.

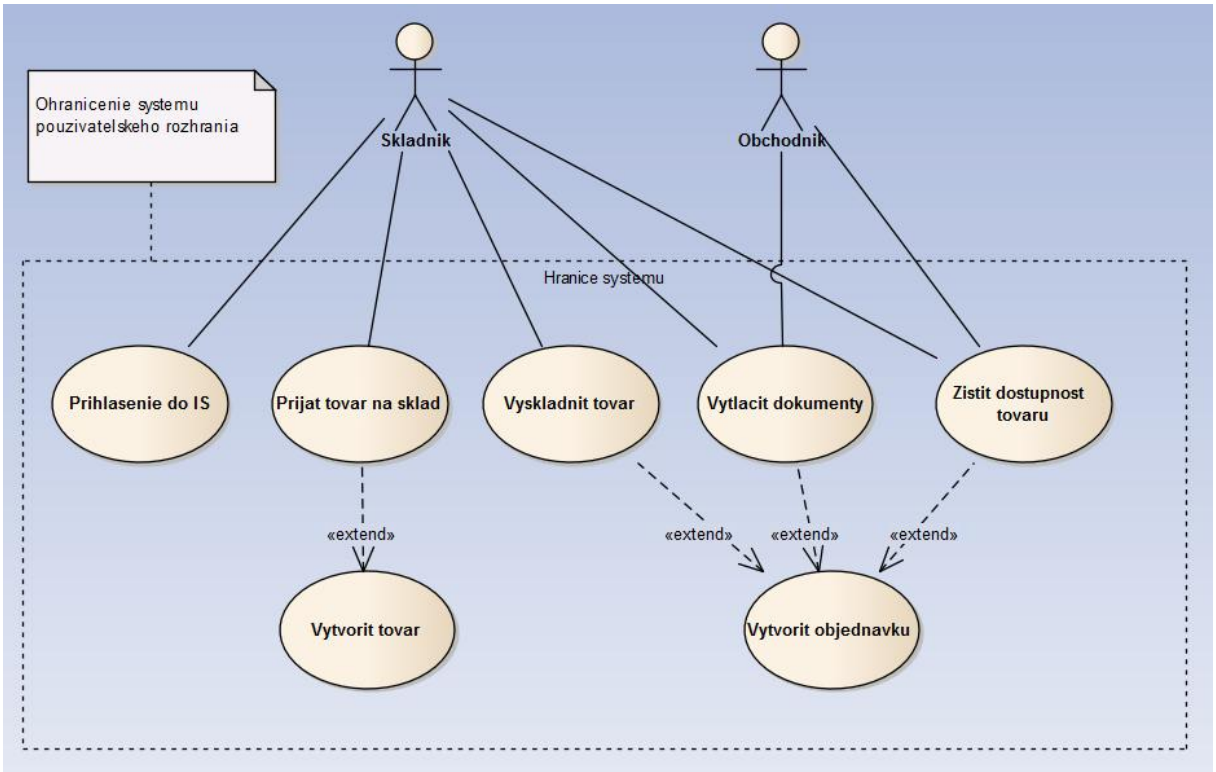
Funkčné požiadavky:

- prihlásenie skladníka do IS
- tvorba dokumentov, ich tlač a export
- práca s tovarom
- tvorba objednávok

Nefunkčné požiadavky:

- zabezpečiť pripojenie klient server
- klient spustiteľný v internetovom prehliadači
- server pripojený do lokálnej siete

Obrázok 13: Diagram prípadov použitia užívateľského rozhrania



Zdroj: vlastné spracovanie

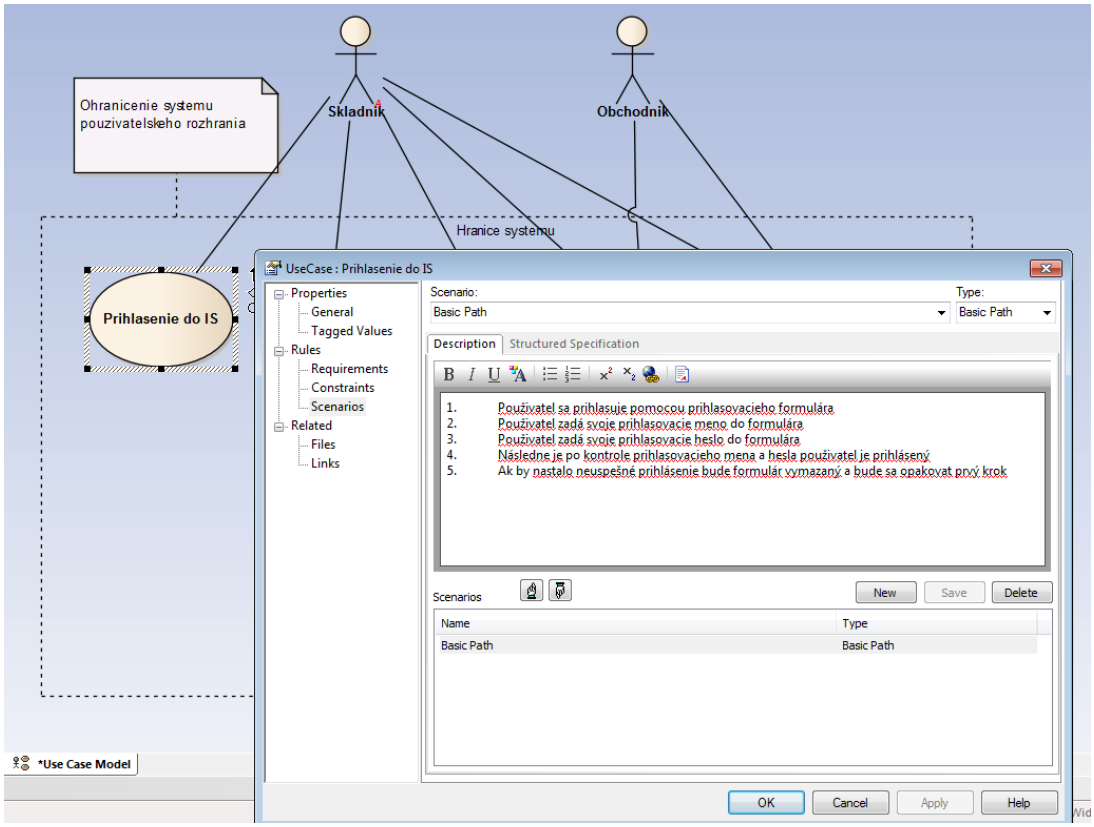
Prípady použitia predstavujú spôsoby použitia rozhrania systému. Kvôli tomu je ideálne spísať scenár pre každý jeden prípad použitia. V Enterprise Architecte je to možné napísať pomocou vloženia textu s pripraveným scenárom pre každý jeden prípad použitia.

Túto možnosť Enterprise Architect ponúka v záložke v kontextovej ponuke. V tomto prípade by scenár mohol byť nasledovný:

1. Používateľ sa prihlasuje pomocou prihlasovacieho formulára.
2. Používateľ zadá svoje prihlasovacie meno do formulára.
3. Používateľ zadá svoje prihlasovacie heslo do formulára.
4. Následne je po kontrole prihlasovacieho mena a hesla používateľ je prihlásený.
5. Ak by nastalo neúspešné prihlásenie bude formulár vymazaný a bude sa opakovať prvý krok.

Týmto spôsobom poskytuje scenár použitia dobrý prehľad o prihlasovacích operáciách. Nie je ale možné definovať každú jednu požiadavku, napríklad obmedzenie pri zadaní troch zlých prihlasovacích pokusoch. Preto je dobré použiť možnosti, ktoré diagramy UML poskytujú, napríklad stavové diagramy, diagramy aktivít alebo sekvenčné diagramy.

Obrázok 14: Vloženie scenára použitia v Enterprise Architecte



Zdroj: vlastné spracovanie

4 Modelovacie nástroje a ich použitie

4.1Enterprise Architect

V nasledujúcich krokoch je opísaný postup, ako vytvoriť projekt v Enterprise Architect. Project sa skladá z jedného súboru alebo slúži ako úložisko pre jeden alebo viacero modelov.

Prvým krokom je buď otvoriť už existujúci projekt alebo vytvoriť úplne nový projekt. V tomto príklade vytvoríme projekt založený na jednom súbore a pridáme niektoré modely, ktoré sú založené na šablónach. Vytvoríme jednoduchý Use Case Diagram, ktorý budeme ďalej prispôsobovať našim požiadavkám. Kliknutím na projekt v prehliadači súborov, ho môžeme kedykoľvek znova otvoriť. Objaví sa nám zoznam súčasných projektov (Recent Projects) na úvodnej stránke (Start Page).

Vytvorenie nového projektu:

- Zapneme Enterprise Architect. Zobrazí sa nám dialóg „Open Project“ (Otvorenie projektu). Ak sa nezobrazí, môžeme stlačiť Ctrl+O, aby sme dané zobrazenie iniciovali.
- Klikneme na možnosť *„New Project“* (Nový projekt) a vyberieme si meno a miesto kde sa nám projekt dodatočne uloží. Otvorí sa klasické dialógové okno prehliadača súborov Windows.
- Vyhľadáme vhodnú zložku pre náš projekt a zadáme názov projektu. Klikneme na možnosť *„Save“* (Uložiť). Enterprise Architect následne vytvorí nový súbor projektu a uloží ho na nami vybranom mieste. Projekt sa automaticky otvorí a zobrazí sa *„New Model Wizard“* (Sprievodca novým modelom).
- V ľavom stĺpci sa presvedčíme, že je zvolená voľba „Basic UML 2 Technology“ a na pravej strane si vyberieme *„Use Case“* diagram odkliknutím. Model Wizard automaticky vytvorí pre náš nový Use Case model s začiatočným diagramom, vedomosťami a prvkami, ktoré nám môžu pomôcť.
- Klikneme na tlačidlo *„OK“* a tým si uložíme zmeny do súboru.

Po vytvorení projektu, ktorý obsahuje aspoň jeden model, môžeme pridať zobrazenie tohto modelu. Potom pridáme balíček s prvkami diagramu a konektory (vzťahy). Pohľad – pohľady poskytujú určitý pohľad na rozpracovaný model, napríklad z pohľadu obchodných procesov, logických operácií, dynamického pohľadu atď. Nimi rozširujeme model v závislostí na konkrétnych požiadavkách na modelovanie.

4.1.1 Popis produktu Enterprise Architect (Sparx Systems)

Spoločnosť Sparx Systems sídli v Austrálií a má bohatú históriu inovácií a vývoja na modelovacom/UML trhu. Sparx Systems sú prispievajúcim členom Object Management Group (OMG), ktorý je osobne zodpovedný za dodržiavanie noriem v rámci definovania a udržiavania UML a súvisiacich špecifikácií.

Enterprise Architect je Computer Aided Software Engineering (CASE) nástroj. Tento nástroj slúži na konštruovanie, projektovanie a vývoj softvérových systémov, modelovanie obchodných procesov a na iné všeobecné modelovacie účely. Produkt Enterprise Architect je založený na najnovšej verzií UML 2.1. Enterprise Architect je progresívny nástroj, ktorý zvláda všetky aspekty vývojového cyklu a umožňuje sledovať požiadavky od zberu, cez analýzu, design a implementáciu až po nasadenie a údržbu. Enterprise Architect poskytuje dokonca aj podporu na testovanie či kontrolu zmien. Od iných UML nástrojov sa odlišuje v týchto bodoch:

- Celková podpora UML 2.1
- Zabudovaná podpora pre zber a manažment požiadaviek
- Rozsiahla podpora projektového riadenia, vrátane zdrojov, metriky a testovania
- Podpora testovania: testovacie úlohy, podpora JUnit a NUnit
- Flexibilné možnosti voľby dokumentov: HTML a Rich-Text (RTF) pre tvorcov reportov
- Podpora generovania kódu z modelu a spätného inžinierstva na generovanie modelu z existujúceho kódu, ktorá je priamou súčasťou produktu
- V produkte je integrovaný nástroj Debug Workbench, ktorý umožňuje profilovať JAVA a .NET aplikácie, vytvárať run-time inštancie tried priamo z modelu a vytvárať sekvenčné diagramy na základe aktuálneho stavu zásobníka vykonávanej aplikácie
- Rozšíriteľné modelovacie prostredie, do ktorého je možné pridávať používateľské profily a technológie
- Rýchlosť: Enterprise Architect je výnimočne výkonný nástroj
- Škálovateľnosť: Enterprise Architect vie jednoducho spracovať veľmi veľké modely a veľa súbežných používateľov
- Cena: Cena je prispôsobená tak, aby Enterprise Architect mohol slúžiť kompletne celému tímu, čo robí tímový vývoj softvéru reálnou skutočnosťou.

Enterprise Architect predal viac ako 150 000 licencií a tým dosiahol mimoriadne uplatnenie v širokom spektre priemyselných odvetví a je využívaný tisícami spoločností po celom svete. Enterprise Architect sa stal UML modelovacím nástrojom a alternatívou pre vývojárov, konzultantov a analytikov vo viac ako 60 krajinách a v spoločnostiach počnúc od veľkých, dobre známych, nadnárodných organizácií, až po malých, nezávislých podnikateľov a konzultantov. Sparx softvér je používaný na vyvíjanie mnohých softvérových systémov v širokom spektre priemyselných odvetví, ako je napríklad letectvo, bankový sektor, web vývoj, inžinierska činnosť, medicína, armáda, výskum, akademická obec, doprava, maloobchod, verejnoprospešné služby (ako plynárenský priemysel, elektrárenský priemysel) a elektrotechnické inžinierstvo. Je tiež efektívne používaný na UML výučbu a výučbu obchodnej architektúry na mnohých prominentných akadémiách, v školiacich strediskách a na univerzitách po celom svete. Aktuálne implementácie sa nasadzujú v praxi od jednoužívateľských projektov, až po veľké distribuované projekty, na ktorých pracuje vyše 600 pracovníkov.

Produkt Enterprise Architect pomáha veľkým spoločnostiam ale aj skupinám a jednotlivcom pri návrhu a riadení komplexných informácií. To často súvisí so softvérovým vývojom a dizajnom IT systémov a ich implementáciou, ale môže sa tiež spájať s obchodnými analýzami a modelovaním obchodných procesov. Enterprise Architect integruje a spája rozsiahlu oblasť štrukturálnych informácii a informácii týkajúcich sa správania, napomáhajúc budovať súvislý a dôveryhodný architektonický model, ktorý je, alebo ktorý bude. Ponúka nástroje pre riadenie verzií, hľadaných rozdielov, revidovaných zmien a uskutočňovanie projektového vývoja s pomocou kontroly bezpečnosti a presadzovaním dodržiavania štandardov.

Umožňuje celkové sledovanie požiadaviek počas celého cyklu a tým umožňuje vytvorenie systému presne podľa požiadaviek zákazníkov.

Škálovateľné, ľahko naladiteľné, viac užívateľské prostredie, to všetko ponúka Enterprise Architect pri spájaní členov tímu zo všetkých sekcií a všetkých fáz produktového (alebo systémového) vývoja a údržby životného cyklu, pričom poskytuje rozsiahlu vstavanú podporu pre spoluprácu a zdieľanie informácii medzi jednotlivými členmi tímu. Ponúka spoločný ukladací priestor pre obchodných analytikov, softvérových architektov, vývojárov, projektových manažérov, testerov, predvádzacieho a podporného personálu a tiež jednotný pohľad na komplexný systém, majúci mnoho východiskových bodov a veľa možných podsystémov.

UML 2.1 je otvorený štandard, ktorý poskytuje bohatý jazyk na popisovanie, dokumentovanie a návrh softvéru, obchodných a IT systémov vo všeobecnosti. Produkt Enterprise Architect dovoľuje používateľom využiť plný potenciál UML 2.1 na modelovanie, dizajn a tvorbu rôznych systémov v otvorenom a dobre zrozumiteľnom štýle, ďalej na generovanie kódu, databázových štruktúr, dokumentácie a metriky, transformovanie modelov, špecifikovanie správania a štruktúry, ako bázy pre zmluvné dohody.

Softvér je komplexný a často ťažko zrozumiteľný. Enterprise Architect umožňuje pomocou reverzného inžinierstva načítať veľké množstvo zdrojových kódov a následne analyzovať ich štruktúru. Po získaní základného obrazu o štruktúre je možné využiť špecifické, vstavané, profilovacie a debugovacie nástroje pre analýzu run-time správania aplikácie. Na integráciu štruktúr už existujúcich databáz do modelu je reverzné inžinierstvo podporované pre veľký rozsah databázových systémov.

Enterprise Architect dokáže získať a sledovať informácie o modelovacích jednotkách, ktoré sú dôležité pre výsledok, napríklad testovanie, projektové riadenie a detaily pre údržbu. A dokáže použiť tieto informácie na vedenie a sledovanie vývoja produktu a implementáciu.

Produkt Enterprise Architect podporuje množstvo mechanizmov na exportovanie a importovanie modelov, využitím priemyselných štandardov XMI. Čo umožňuje modelárom využívať informácie vytvorené v iných nástrojoch, kopírovať informácie medzi modelmi produktu Enterprise Architect, a dokonca aj naprogramovať a používať vlastné nástroje, ktoré vedia spracovať XMI.

Model Driven Architecture (MDA) je otvorený štandard podporujúci RAD platformovo nezávislým spôsobom. MDA model sa vytvorí na najvyššej úrovni abstrakcie a následne je možné ho transformovať do platformovo závislého modelu a zdrojového kódu. Enterprise Architect podporuje rozsiahlu sadu MDA nástrojov.

Sumár funkcií produktu Enterprise Architect.

- Modelovať komplexné informačné, softvérové a hardvérové systémy s využitím notácie kompatibilnej s UML
- Modelovať, manažovať a v už nasadených systémoch aj sledovať požiadavky
- Tvoriť detailnú a kvalitnú dokumentáciu v RTF a HTML formátoch

- Využiť framework Enterprice Architect Framework, ktorý je priemyselným štandardom.
- Generovať a pomocou reverzného inžinierstva analyzovať kód vo viac ako 10 programovacích jazykoch
- Modelovať databázy, vytvárať DDL skripty, a prostredníctvom reverzného inžinierstva analyzovať databázové schémy prístupne cez ODBC
- Manažovať verzie s podporou mergovania
- Centralizovať celopodnikovú dokumentáciu procesov a informačných systémov
- Modelovať závislosti medzi jednotlivými elementami, správaním a stavmi systému
- Modelovať hierarchie tried a ďalších komponentov
- Spravovať slovník projektu, úlohy a pripomienky k projektu
- Priradiť jednotlivým elementom modelu zdroje a porovnávať plánované a reálne využitie týchto zdrojov
- Zdieľať modely použitím najnovšieho XMI 2.1 formátu. (Sú podporované aj predchádzajúce verzie). • Importovať modely v XMI formáte z iných nástrojov
- Sledovať verzie Version Control využitím technológie XMI spolu so SCC, CVS, alebo Subversion konfiguráciami
- Použiť UML profily UML Profiles na tvorbu užívateľských rozšírení pre doménovo-špecifické modelovanie
- Uložiť kompletné diagramy ako UML vzory UML patterns a tieto neskôr použiť ako šablónu
- Analyzovať a zaznamenávať vzťahy medzi elementmi, použijúc tabuľkovú vzťahovú maticu Relationship Matrix
- Skriptovať a automatizovať úlohy použitím detailného rozhrania Automation Interface
- Pripojiť sa do zdieľaných databázových úložných priestorov použitím MS SQL servra, MySQL, Oracle a iných
- Replikovať zmeny v distribuovanom prostredí pomocou balíčkov Controlled XMI Packages
- Uskutočňovať transformácie modelu do modelu použitím Model Driven Architecture (MDA)
- Pomocou Model Views je možné vytvárať a zdieľať dynamické pohľady na jednotlivé elementy a diagramy modelu

- Zaznamenávať Mind Maps, vytvárať Business Process Models a Data Flow Diagrams použitím UML
- Vizualizovať vykonávané aplikácie použitím technológie Debug a Profiling Workbench

Enterprise Architect podporuje všetky UML 2.1 modely a diagramy. Môžete modelovať obchodné procesy, web stránky, užívateľské rozhrania, siete, hardvérové nastavenia, odkazy a mnohé ďalšie elementy vášho vývoja. Čitatelia neoboznámení s UML môžu nájsť krátku príručku na nasledujúcej URL*:* http://www.sparxsystems.com/uml-tutorial.html Enterprise Architect bol prvý UML nástroj na zavedenie úplnej podpory UML 2.1 v apríli 2004, pokračuje vo vylepšovaní a aktualizovaní UML 2.1 podpory a podporuje všetkých 13 typov diagramov z UML 2.1

Program Enterprise Architect poskytuje doplnkové typy diagramov, ktoré rozširujú základné UML diagramy o modelovanie obchodných procesov, mind mapping, špecifikácie formálnych požiadaviek, data-flow diagramy a ďalšie doménovo špecifické modely. Prostredie produktu tiež poskytuje množstvo alternatívnych pohľadov, ktoré robia prácu s UML diagramami viac intuitívnou a efektívnou. Jedným príkladom je tabuľkový editor State Table, ktorý predkladá štandardný UML State Machine diagram ako editovateľnú tabuľku.

Tvorba dokumentácie je podstatná na pochopenie celkovej užitočnosti produktu Enterprise Architect. Enterprise Architect vytvára dokumentáciu vysokej kvality v RTF alebo HTML formáte. Môžete modifikovať RTF formátovanie priamo v šablóne RTF Style templates, a tým meniť vzhľad výstupu. Použitím Microsoft® Word® môžete miešať a spájať výstupy z modelu s dokumentmi, a tým ďalej vylepšiť dokumentáciu. Je tam veľa spôsobov, ako špecifikovať možnosti produktu Enterprise Architect na zdokumentovanie vlastného obsahu. Pri tvorbe dokumentácie je možné:

- Vytvoriť dokumentáciu pre obsah balíčka a jeho pod-balíčkov, tým že balíček vyberiete a spustíte dokumentačný nástroj
- Pri rekurzívnom dokumentovaní balíčka je možné niektoré z pod-balíčkov z procesu vylúčiť
- K balíčku je možné priradiť RTF šablónu, čo značne zjednodušuje vytváranie konzistentných typov dokumentácie (napr. Tlač Use Case modelu)
- Môžete vybrať spolu ´group´ a ´order´ balíčky, a to odlišným spôsobom od projektového zobrazenia, vytvorením ´virtuálnych´ dokumentov.

RTF Style Template Editor - umožňuje vytvárať a editovať vlastné RTF šablóny, vďaka čomu je možné definovať ľubovoľný formát RTF dokumentácie. Style Template Editor umožňuje pre jednotlivé elementy modelu definovať, ktoré ich položky budú súčasťou dokumentácie. Prostredníctvom nástroja Style Editor je možné definovať formátovanie a do dokumentu je možné vkladať prvky, ako napr. obsah a nadpisy.

Enterprise Architect umožňuje exportovať model, alebo jeho časť do HTML. HTML formát poskytuje ľahko použiteľný strom diagramov s jednoduchou navigáciou pomocou hypertextových odkazov. Dokumentácia je tvorená na základe HTML šablón, takže je možné doladiť stránky tak, aby zodpovedali firemným štandardom.

Enterprise Architect vám umožňuje spájať Rich Text dokumenty s akýmikoľvek elementami v modely. Zložené dokumenty sú vytvárané z upravovateľných šablón a sú zahrnuté do generovaných web a Word-based reportov.

Väčšinou je prvým krokom vo vývoji riešenia zozbierať požiadavky. Či už sa jedná o vývoj softvérovej aplikácie, alebo mapovanie obchodného procesu. Požiadavky sú v podstate o tom, čo systém potrebuje vedieť robiť. Funkcie vstavanej podpory na zber a manažment požiadaviek produktu Enterprise Architect môžu byť použité na:

- Definovanie organizovaného a hierarchizovaného modelu požiadaviek
- Spájanie a sledovanie implementácie systémových požiadaviek do modelovacích elementov
- Vyhľadávanie a reprtovanie požiadaviek a vykonávanie impaktných analýz pri zohľadňovaní a rešpektovaní zmien požiadaviek.

Existuje mnoho prístupov k modelovaniu obchodného procesu (Business Process Modeling - BPM) pri použití UML ako základného modelovacieho jazyka. Predovšetkým diagramy aktivít (Activity Diagrams), objektové diagramy (Object Diagrams) a užívateľské profily (User Profiles) poskytujú hojnosť použiteľných, modelovacích možností pre BPM analytikov. Enterprise Architect kombinuje vlastnosti jazyka UML 2.1 s vlastnými prvkami na účely analýzy, spracovania požiadaviek a práce s procesmi (napr. prvky change, feature a issue).

Jedným z populárnych prístupov do obchodného modelovania je Business Process Modeling Notation (BPMN) profil. Táto notácia je špecializovaná pre modelovanie obchodných procesov, pričom pomocou BPMN profilov je možné ju priamo mapovať do UML. Sparx Systems poskytujú vstavaný UML profil pre BPMN modelovanie v produkte Enterprise Architect.

Model je možné validovať voči pravidlám jazyka UML a tiež i voči obmedzeniam, ktoré sú definované v modely pomocou jazyka Object Constraint Language (OCL). Validovať je možné jeden element modelu, diagram, alebo celý balíček.

Užívateľské rozhranie, nástroje a bustre produktivity

Enterprise Architect sa dodáva s množstvom preddefinovaných vzorov (Model Patterns), na základe ktorých je možné vytvárať projekty a modely. Každý vzor obsahuje užitočné poznámky, referencie a spúšťacie prvky a poskytuje základy, na ktorých je možné vytvoriť model.

Užívateľské rozhranie produktu Enterprise Architect sa skladá zo spektra kvalitne prepracovaných okien high-impact windows, ponúk menu a líšt nástrojov toolbars, ktoré môžu zostať skryté, ak nie sú potrebné, alebo môžu zostať zobrazené v akejkoľvek polohe na obrazovke, vhodne umiestnené podľa metód práce. Z týchto medzi kľúčové patria:

- Prieskumník projektu (Project Browser), ktorý zobrazuje kompletný obsah vášho modelu alebo projektu v hierarchickom formáte (s možnosťou výberu úrovne číslovania) a ktorý vám umožňuje pridať, vybrať, prerobiť alebo vymazať balíčky, diagramy alebo elementy kdekoľvek v projekte.
- UML lišta nástrojov (UML Toolbar) produktu Enterprise Architect, ktorá je obsahovo citlivá na vytvárané diagramy, a poskytuje rýchle a výkonné spôsoby výberu a tvorby vhodných modelovacích elementov alebo konektorov, ktoré sú buď UML, z rozšírených diagramov alebo importovaných technológií.
- Diagramové zobrazenie (Diagram View), ktoré vám umožňuje zobrazovať a vyvíjať diagramy vybrané z prieskumníka projektu (Project Browser); diagramové pozadia, konektory a elementy môžu byť zafarbené s odstupňovaním alebo bez odstupňovania farieb pre účely programovania farieb alebo lepšieho zobrazenia a prezentácie.
- Kontextové ponuky (Context menus), ktoré poskytujú alternatívy špecifické podľa typu objektu a jeho priameho prostredia.

Enterprise Architect umožňuje rýchlu editáciu vlastností ľubovoľného elementu priamo v diagrame. Takto je možné rýchle pridávať a meniť atribúty, operácie a parametre bez potreby opustiť editor diagramu.

Rýchly zostavovací program poskytuje rýchly a vhodne umiestnený mechanizmus na vytváranie nových elementov a konektorov v diagrame. Jeho na kontext citlivé

výberové ponuky pomáhajú viesť vytváranie korektných modelov, šetriac čas užívateľov a zlepšujúc celkovú produktivitu.

Medzi ostatné diagramové funkcie patrí:

- Exportovanie diagramov do spektra zobrazovacích formátov (.bmp, .jpg, .png, .gif, .emf a .wmf)
- Swimlanes umožňujú logické segmentovanie diagramov
- Pan a Zoom okien zabezpečujú jednoduchú navigáciu a prehliadanie komplexných diagramov
- Uzamknutie (Lock) diagramov na ochranu pred náhodnými modifikáciami
- Naformátované texty (Shape Scripts) produktu Enterprise Architect vám umožňujú špecifikovať užívateľské formáty použitím textového jazyka, pričom každý formát je zakódovaný podľa šablón; tieto užívateľské šablóny sú predkreslené namiesto štandardnej UML notácie pre každý element toho istého typu
- Môžte tiež priložiť alternatívne zobrazenia (alternative images) – ako metasúbory (metafiles) – v diagramoch a elementoch, na zmenu štandardného vzhľadu (standard image)
- Vystopovateľnosť (Traceability)
- Auditovanie (Auditing View)

Kontrolné funkcie produktu Enterprise Architect umožňujú sledovať a zaznamenávať zmeny, ktoré časom vzniknú v modely. Umožnením tejto voľby môžu modelovací administrátori prezerať rozsah informácií s ohľadom na zmeny, ako napríklad:

- Kto zmenil element
- Koľko elementov bolo zmenených
- Kedy boli dáta zmenené
- Aké boli predchádzajúce hodnoty
- Aký typ elementov bol zmenený

Kontrolné zobrazenie (Audit view) môže byť prispôsobené pre zobrazovanie zmien špecifických typov (vrátane zmien v nastavení auditovania), pre špecifické oblasti alebo úrovne modelu, nad špecifickými časovými periódami a podľa každého užívateľa. Kontrolné zobrazenie (Audit view) môže byť zosynchronizované s prieskumníkom projektu (Project Browser) a zoznamom elementov (Element list) (viď. nižšie) na overenie zmien podľa toho, ako prehodnocujete elementy, a tietio zmeny môžu byť automaticky

zobrazované v histórii kontroly (Audit History) vo výstupnom okne programu Enterprise Architect (Output window).

Zoznam elementov (Element list) je tabuľkové, prepisovateľné zobrazenie elementov, ktoré môže byť umiestnené v hlavnom pracovnom priestore. Môžete používať zoznam elementov na zjednodušenie procesu tvorby a aktualizácie elementov v balíčku alebo diagramoch vybraných z prieskumníka projektu (Project Browser). Toto môže byť čiastočne použiteľné pre analytikov na tvorbu a nastavenie formálnych požiadaviek a definícií vo vnútri modelu. Tiež môžete tlačiť zoznam alebo generovať RTF dokument priamo zo záznamov na zozname elementov.

Produkt Enterprise Architect uľahčuje vyhľadávanie a zobrazovanie používaných elementov. Funkcia zobraziť použitie (Show Useage) v diagrame, prieskumník projektu (Project Browser) alebo zoznam elementov (Element list) zobrazujú všetky výskyty daného elementu v celom modely, a umožňujú vám ľahšie vojsť do akéhokoľvek výskytu.

Hierarchické okno zobrazuje mini obrázok súčasnej elementárnej kompozície s rešpektom voči ostatným elementom. Táto informácia je odvodená zo vzťahu s dieťaťom alebo pokrvným príbuzenstvom. V hierarchii zobrazené vzťahy zahrňujú agregáciu, zabezpečenie a závislosť; vložené elementy sú tiež prítomné. To pomáha rozšíriť zobrazenie miest, na ktorých sa element v modely vyskytuje .

Vzťahová matrica vám (Relationship Matrix) umožňuje študovať vzťahy medzi modelovacími elementmi v tabuľkovom zobrazení. Taktiež vám umožňuje vytvárať, modifikovať a mazať vzťahy medzi elementmi jednoduchým kliknutím myšky.

Enterprise Architect môže automaticky ošetriť diagram všetkými elementmi, ktoré sa vzťahujú k danému elementu. Môžete filtrovať vložené elementy založené na type, druhu a hĺbke vzťahu. Funkcia vloženie príbuzného elementu (Insert Related Element) poskytuje rýchlu a silnú cestu, ako vytvoriť prehľady špecifického vzťahu pre framework alebo reverzný inžiniersky zdrojový kód.

Model Search generuje report, ktorý môžete prezerať v hlavnom pracovnom priestore. Vymenováva každý element v modely, ktorý spĺňa vysoko univerzálne kritériá zadefinované vami, vrátane podmienok vyhľadávania a jeho typu. Elementy uvedené vo výsledkoch vyhľadávania sa dajú vybrať na tlačenie, reporting, upravovanie, pridávanie do dokumentácie, a na vkladanie tém z diskusného fóra.

Okno modelového zobrazenia (Model Views) produktu Enterprise Architect poskytuje filtrované zobrazenie elementov zo základu modelovej hierarchie. Použitím

Model Views ste schopní usporiadať elementy podľa kritérií vyhľadávania (napríklad najčastejšie spúšťané vyhľadávania), preferované elementy a diagramy alebo technologicky špecifikované informácie, ako napríklad elementy zodpovedajúce hľadisku čiastkovej štruktúry. Model Views môžu byť ukladané lokálne pre používanie jednotlivcami, importované a exportované medzi užívateľmi alebo zahrnuté v spoločnej schránke pre archiváciu zoznamov so vzájomnou súvislosťou.

Enterprise Architect podporuje množstvo osožných a sebestačných reportov, napr.:

- Zdrojové detaily a detaily zadania (Resource and Task Details)
- Projektové výsledky (Project Issues)
- Projektový slovník (Project Glossary) • Projektové štatistiky (Project – size – Statistics)
- Závislostné a implementačné detaily (Dependency and Implementation Details)
- Testovacie detaily (Testing Details)

Enterprise Architect ponúka špecifickú funkčnosť pre zdieľanie projektov v tímovom prostredí a distribuovanom vývojovom prostredí. Projekty môžu byť zdielané použitím spoločných ukladacích priestorov dostupných prostredníctvom siete, použitím replikácie, voľby XMI Imprt/Export, kontroly verzie (Version Control), kontroly balíka (Package Control) a užívateľskej bezpečnosti (User Security). Podpora pre veľké modely a mnoho súbežných užívateľov. Corporate edícia produktu Enterprise Architect umožňuje namiesto do EAP súborov uložiť model do vyhradeného serverového DBMS. Enterprise Architect podporuje nasledujúce DBMS:

- MS SQL Server
- MySQL
- Oracle
- PostgreSQL
- Progress OpenEdge
- MSDE Server
- Adaptive Server Anywhere

Enterprise Architect podporuje formát XML Metadata Interchange (XMI), ktorý umožňuje vývojárom vymieňať si celé modely, alebo ich časti. XMI umožňuje exportovať balíček, alebo celý model do XML súboru, ktorý môže byť importovaný do iného modelu, alebo byť uložený do systému pre správu verzií.

Zabezpečenie užívateľov je k dispozícii v Corporate edícii programu Enterprise Architect a umožňuje definovať, kto môže robiť aké zmeny v modely. Elementy môžu byť blokované podľa užívateľa alebo skupiny; ak je zabezpečenie užívateľov aktivované, tak je pred prístupom k modelu potrebné zadať heslo. Bezpečnosť v programe Enterprise Architect nie je navrhnutá tak, aby predchádzala neautorizovanému prístupu; skôr je zameraná na podporu kolektívneho designu a vývoja, tým že umožňuje predchádzať konfliktom pri zmenách a redukuje možnosť neúmyselnej zmeny modelu užívateľmi, ktorí nepatria medzi autorov modelu.

Projektové diskusné fórum umožňuje užívateľom hovoriť o vývoji a postupe projektu. Členovia tímu môžu prezerať a posielať správy vo vnútri modelovacieho prostredia a môžu priradiť svoje pripomienky priamo k elementom modelu. V distribuovanom prostredí je možné použiť diskusné fórum na vzdialenom serveri.

Sparx Systems podporuje niekoľko architektonických framework-ov. Tieto tvoria vo svojej triede priemyselný štandard a sú založené na UML a príbuzných technológiách. Toto umožňuje jednotnosť architektúry a výmenu informácií prostredníctvom štandartných technológií ako je napr. XMI. Enterprise Architect podporuje prostredníctvom rozšírení (plug-in) nasledujúce framework-y:

- Zachman Framework (viď. http://www.sparxsystems.com/zachman)
- DoDAF (viď. http://www.sparxsystems.com/dodaf-modaf)
- MODAF (viď. http://www.sparxsystems.com/dodaf-modaf)
- Open Group's TOGAF (viď. http://www.sparxsystems.com/togaf)

Taktiež podporuje Federal Enterprise Architecture Framework (FEAF) referenčný model

Enterprise Architect podporuje dve kľúčové technológie W3C XML, kritické pre plnú podporu SOA: XML Schema (XSD) aWeb Service Definition Language (WSDL). Spojenie UML 2.1 a XML ponúka prirodzený spôsob, ako špecifikovať, vytvárať a umiestňovať jednotlivé SOA komponenty v rámci organizácie.

XML schémy sa modelujú ako UML class diagramy, pričom sa používa toolbox XML Schema, kde sú vstavané nástroje na podporu XSD. To umožňuje generovať XSD súbor z abstraktného UML class diagramu.

Enterprise Architect dokáže generovať a načítať WSDL dokumenty a obsahuje WSDL toolbox, pomocou ktorého je možné pohodlne spracovávať WSDL dokumenty. WSDL dokumenty sú v modely reprezentované ako komponenty so stereotypom WSDL,

ktorý je uložený v hierarchii balíčkov. WSDL dokumenty reprezentujú cieľový WSDL namespace a k nemu prislúchajúce XSD typy, správy, typy portov, väzby a služby.

Enterprise Architect umožňuje automatické generovanie zdrojového kódu z UML modelu, načítanie existujúceho kódu do modelu pomocou reverzného inžinierstva a tiež synchronizáciu kódu a modelu. Je k dispozícii iba v edíciách Professional a Corporate. Enterprise Architect má priamu vstavanú podporu pre viac ako 10 najpoužívanejších jazykov:

- ActionScript (Macromedia Flash vývojársky jazyk)
- C
- C# (pre oboje .NET 1.1 a .NET 2.0)
- C++ (standard plus .NET managed C++ extensions)
- Delphi
- Java (vrátane Java 1.5, Aspects and Generics)
- PHP
- Python
- Visual Basic
- Visual Basic .NET
- Importovanie .jar súborov a .NET Assemblies

Enterprise Architect dokáže pomocou reverzného inžinierstva načítať nasledujúce typy binárnych modulov:

- Java Archive (.jar)
- Net PE file (.exe, .dll)*
- Intermediate Language (.il).
- Natívne Windows DLL a Exe súbory nie sú podporované, iba PE súbory obsahujúce .Net dáta v assembly.

Enterprise Arcitect obsahuje Code Template Framework (CTF) a umožňuje špecifikovať šablóny na generovanie kódu, v rámci ktorých sa definuje spôsob transformácie jednotlivých UML elementov do elementov daného jazyka. Code Template Framework umožňuje:

- Generovať zdrojový kód z UML modelov
- Upravovať spôsob, ktorým produkt Enterprise Architect generuje zdrojový kód
- Generovať kód aj v jazykoch, ktoré nie sú v produkte Enterprise Architect priamo podporované

Enterprise Architect podporuje priebežné generovanie kódu Live Code Generation, ktoré automaticky aktualizuje váš zdrojový kód, ihneď ako robíte zmeny vo vašom modely. Napríklad, keď vytvárate nové operácie a vlastnosti pre triedy v modely, tieto sú ihneď zapísané do zdrojového súboru.

V rámci prostredia je možné prezerať a modifikovať zdrojový kód. Ak vyberiete element, ku ktorému je priradený kód, tento kód sa zobrazí v editore. Zobrazí sa tiež navigovateľná osnova štruktúry kódu a na kód je aplikovaný syntax highlighting. K dispozícii je tiež toolbar pre rýchle generovanie kódu a pre synchronizáciu kódu s modelom.

Edície Professional a Corporate dokážu priamo vo svojom prostredí buildovať, testovať, debugovať a spúšťať skripty, čo umožňuje plne integrovať modelovanie v UML s vývojárskymi nástrojmi pre prácu s kódom. Pomocou MDA transformácií je možné generovať triedy pre NUnit a JUnit testy priamo z tried v modely, čo umožňuje implementovať testovací proces priamo v prostredí Enterprise Architect, vďaka čomu je modelovanie v UML plne integrované do procesu vývoja aplikácie. Okrem podpory pre testovanie Enterprise Architect podporuje aj debugovanie pre Javu, .NET a natívne MS jazyky (C++, C a VB). Tieto debugre sú špeciálne navrhnuté tak, aby dokázali analyzovať stack pozastaveného vlákna bežiacej aplikácie. Dokáže konvertovať stack trace do sekvenčných diagramov, čím vizualizuje vykonávaný kód do modelu. Takto je možné analyzovať komplexný kód a kontrolovať, či sa kód vykonáva tak, ako bol navrhnutý architektom a vývojármi.

Enterprise Architect umožňuje špecifikovať, akým spôsobom bude buildovaný a vykonaný kód spojený s balíkom, pričom balík môže pre tieto činnosti mať priradených aj viac konfigurácií, čo uľahčuje prácu s viacerými verziami. Tiež je podporované spracovanie výstupu kompilátora s následným zvýraznením prípadných chýb.

Debugovanie je podporované priamo v rámci prostredia Enterprise Architect, pričom sa vyberá debugovacie rozhranie vhodné pre daný jazyk. To sa zadefinuje priradením debugovacích skriptov jednotlivým balíčkom. V rámci debugovania je možné spracovať stack trace a priamo z neho generovať sekvenčné diagramy Sequence diagrams, čo predstavuje veľmi efektívny spôsob ako preskúmať, čo presne sa deje počas behu aplikácie.

Zo zaznamenaných výsledkov debugovania je možné jednoducho generovať detailné a úplné sekvenčné diagramy. Je možné buď manuálne prechádzať určité časti

kódu, alebo nechať produkt Enterprise Architect, aby sám prešiel celý kód. Z takto zaznamenaných výsledkov je možné generovať sekvenčné diagramy.

Enterprise Architect podporuje unit testy pomocou Nunit a Junit. Testovacie triedy a metódy je možné generovať priamo z modelu. Výsledkom týchto transformácií sú prázdne procedúry, do ktorých stačí doplniť testovací kód. Výsledok je možné ručne buildovať a spustiť, alebo je možné balíčku priradiť test script, ktorý spustí testovací program, pričom Enterpris Architect zobrazí výsledok testovacieho programu priamo vo svojom IDE.

Jedným z kľúčových princípov pri používaní unit testov je, že testy treba napísať ako prvé. Enterprise Architect tento prístup plne podporuje. Pri pridaní novej metódy do triedy sa prostredníctvom transformácií vytvorí aj príslušná testovacia metóda, ktorú môžete implementovať neskôr, čo je možné urobiť ešte pred tým, ako je vytvorený kód, ktorý sa bude testovať.

Užívatelia Enterprise Architect môžu potvrdiť, že v prostredí Enterprise Architect sa pracuje oveľa rýchlejšie ako vo väčšine momentálne dostupných nástrojov, lebo je bez problémov škálovateľný a zvláda spracovávať extrémne veľké modely bez významnejšej straty výkonu.

Enterprise Architect umožňuje kontrolu verzií Version Control na úrovni balíčka, alebo jeho podbalíčkov a v rámci centrálneho ukladacieho priestoru. Tento priestor je spravovaný užívateľom prostredníctvom definovaného systému pre kontrolu verzií, čo má nasledujúce dve kľúčové výhody:

- poskytuje možnosť koordinovať zdieľanie balíčkov medzi užívateľmi
- ukladá históriu zmien do balíčkov produktu Enterprise Architect a umožňuje získať späť predchádzajúce verzie

Enterprise Architect podporuje nasledovné aplikácie kontroly verzií:

- Akýkoľvek produkt kontroly verzií, ktorý je v zhode s Microsoft Commno Code Control standard, verziou 1.1 alebo vyššou. (Napríklad Visual Source Save alebo Clear Case)
- Microsoft Team Foundation System
- Subersion, ktorá je sprístupnená zo stránky http://subversion.tigris.org

Corporate edícia Enterprise Architect umožňuje v stanovený čas určiť aktuálny stav balíčka ako baseline. Neskôr je možné pomocou nástroja Enterprise Architect Compare utility (diff) vizuálne sledovať rozdiely medzi base-line a aktuálnym stavom balíčka. Tiež

je možné vrátiť sa v ľubovoľnom bode vývoja do stavu, ktorý je uložený ako base-line, a tým zrušiť vykonané zmeny. Ďalej sa dá mergovať do modelu zmeny, čo umožňuje, aby na jednom balíčku pracovalo off-line viacero užívateľov naraz, pričom ich zmeny je možné neskôr bezpečne spojiť. Okrem porovnávania a mergovania zmien oproti base-line modelu je tiež možné porovnávať balíček a:

- súbor na disku, vytvorený použitím XMI exportovacej funkčnosti produktu Enterprise Architect v balíčku
- verziovo kontrolovaný XMI súboru pre vybraný balíček
- akúkoľvek základnú linku balíčka v externom modely, do ktorého máte prístup

Enterprise Architect poskytuje množstvo mechanizmov na integrovanie vášho modelu do nástrojov tretej strany, ponúka tiež programovateľné API, add-ins framework a niekoľko vstavaných MDG riešení.

Automatizovateľné rozhranie vám umožňuje vstúpiť do vnútra modelov produktu Enterprise Architect. Tu sú príklady úloh, ktoré môžete vykonávať použitím automatizovateľného rozhrania:

- vykonávať opakujúce sa úlohy, ako napríklad aktualizovanie čísla verzie pre všetky elementy v modely
- generovať kód zo stanoveného diagramu
- tvoriť užívateľské reporty
- vykonať účelové vyhľadávania v modely (Ad hoc Queries)

Ľubovoľné vývojárske prostredie, ktoré podporuje ActiveX, by malo byť schopné využívať automatizovateľné rozhranie.

Schopnosti produktu Enterprise Architect je možné rozširovať pomocou add-ins. Add-ins majú niekoľko výhod v porovnaní s aplikáciami, ktoré automatizujú produkt Enterprise Architect pomocou

ActiveX, ako je napr. možnosť vytvoriť v prostredí Enterprise Architect nové menu a definovať reakcie na rozličné udalosti, napr. na výber príslušného príkazu z menu.

Sparx vyvinul množstvo MDG produktov poskytujúcich súčinnosť s ostatnými nástrojmi. Enterprise Architect priamo podporuje spoluprácu s nástrojmi MS Visual Studio a Eclipse pomocou MDG Link for Visual Studio a MDG Link for Eclipse.

MDG Integrovanie tesne integruje produkt Enterprise Architect do Microsoft® Visual Studio® 2005 a 2008 a Eclipse vývojárskych zariadení. Tento produkt umožňuje užívateľom skúmať a upravovať UML model vo vnútri Visual Studio alebo Eclipse a tiež poskytuje mnoho kľúčových funkcií produktu Enterprise Architect priamo v rámci týchto IDEs, obsahujúcich rich text a web tvorbu dokumentov, MDA transformácie, Baseline manažment a inžinierstvo kľúčových XML technológií.

Enterprise Architect podporuje vykonávanie MDA transformácií a plne konfigurovateľných konverzií elementov, alebo fragmentov modelu z jednej domény do druhej. Väčšinou zahŕňa konverziu modelov Platform-Independent Model (PIM) do Platform Specific Model (PSM). Jeden element PIM modelu môže byť transformovaný do mnochých elementov PSM modelu z viacerých domén. Použitie takýchto transformácií znamená obrovské zvýšenie produktivity a znižuje potrebu tvorby tried a elementov pre konkrétne domény. Napríklad, databázové tabuľky môžu byť automaticky generované z PIM modelu perzistentných tried.

MDA kapacity produktu Enterprise Architect vám umožňujú :

- s využitím zabudovaných transformácií generovať: o Data Models (DDL) o Code Models, vrátane C# a Java o XML models, ako XSD a WSDL o Test Model pre JUnit and NUnit
- definovať nové transformácie použitím výkonného a na šablónach založeného nástroja
- opakovať transformácie pre zabezpečenie konzistencie medzi zdrojovým a cieľovým modelom tak, ako sa časom menia

Enterprise Architect obsahuje Data Modeling profile, ktorý rozširuje UML a poskytuje intuitívne mapovanie medzi databázovými konceptmi tabuliek a relácií a UML konceptmi tried a asociácií. Tieto rozšírenia umožňujú modelovať kľúče, trigre, constrainty, RI a ďalšie databázové koncepty. Pri modelovaní a návrhu databázy budete väčšinou:

- tvoriť modelovací diagram (Data Model diagram)
- tvoriť tabuľku
- nastavovať vlastnosti tabuľky
- tvoriť stĺpce, primárne kľúče, cudzie kľúče
- tvoriť procedúry, ktoré sa zálohujú
- tvoriť indexy, sekvencie, funkcie a trigre
- generovať DDL pre tabuľky a balíky
- konvertovať dátové typy pre tabuľku, balíček alebo celé DBMS
- importovať databázovú schému z ODBC dátového zdroja
- tvoriť prehľady

Enterprise Architect podporuje dátové modelovanie databázovej schémy z nasledujúcich databáz:

- DB2
- InterBase
- Informix
- Ingres
- MS Access
- MySQL
- Oracle 9i a 10g
- PostgreSQL
- MS SQL Server 2000 a 2005
- SQL Server7
- Sybase Adaptive Server Anywhere
- Sybase Adaptive Server Enterprise
- Firebird.

Enterprise Architect je dostupný v štyroch edíciách: Corporate (Floating), Corporate (Standalone), Professional and Desktop. Každá edícia ponúka spektrum funkcií na podporu požiadaviek rôznych skupín užívateľov, od projektov jednotlivca až po projekty tímov veľkých podnikov. Zmluvný súhlas Corporate (Floating) edície je obzvlášť vhodný pre spoločnosti, ktoré potrebujú riadiť centrálnu pamäť licencovaných kľúčov. Corporate (Floating) kľúče môžu byť používané v priebehu času rôznymi zamestnancami na dočasnej alebo trvalej báze. Viac informácií o edíciách programu Enterprise Architect je dostupných na stránke: http://sparxsystems.com/products/ea_editions.html

UML je jazyk, nie proces. Stanovuje elementy modelovacieho jazyka a ako tieto elementy môžu byť spájané do výsledkov v realite. Nestanovuje ako máte použiť tieto elementy v priebehu času na vytváranie nových softvérových systémov. Ako UML, Enterprise Architect je neutrálny proces v tom zmysle, že obsahuje všetky vybavenia a funkcionality potrebné na implementovanie vybraného procesu, ale nepredpisuje aký tento proces má byť, alebo ako má byť implementovaný. Mnohí užívatelia produktu Enterprise Architect zavádzajú vysoko štruktúrované procesy, ako RUP, pokým iné

používajú viac flexibilné a menej zaťažujúce procesy. Bez ohľadu na stupeň riadenia procesu, ktorý požadujete, Enterprise Architect má nástroje a funkcie potrebné na podporu procesu softvérového vývoja.

4.2 Rational Rose

Nástroj Rational Rose (Rational Object Oriented Software Engineering) nám ponúka viacero modelovacích nástrojov, pre vývoj objektovo orientovaných softvérov. Rational Rose využíva grafické metódy jazyka UML pre používateľov, ktorí chcú modelovať podnikové procesy bez programovania rovnako ako programátori modelujú aplikačnú logiku. Modelovanie s Rational Rose môže byť užitočné v ľubovoľnom bode pri procese vývoja aplikácií.

Počiatočné práce (Analýza požiadaviek a definícií)

- Use Cases
- Class Diagrams
- Sequence Diagram

Spresnenia skorých modelov (System and Software Desing) Rational Rose obsahuje nástroje pre takzvané reverzné inžinierstvo, ako aj forwardové inžinierstvo tried

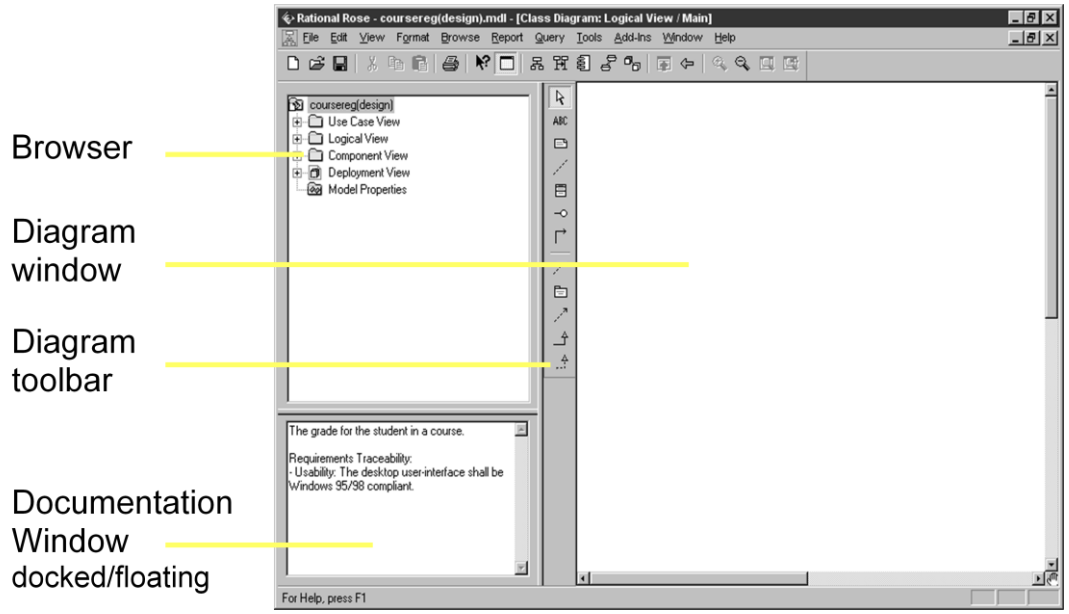
a komponentov architektúry. Môžeme získať cenný náhľad do reálnej architektúry a určite odchýlky od originálneho dizajnu. Rational Rose ponúka rýchly spôsob pre klientov a nových používateľov ako sa oboznámili so systémovými časťami.

Rational Rose nám dovoľuje uložiť model v rozličných formátoch, môžeme si vybrať medzi možnosťami z Save As Type zoznamu:

- Models*.mdl (súčasna verzia Rational Rose)
- Petal*.ptl
- Rose 6.1/6.5 Model
- Rose 4.5/6.5 Model
- Rose 4.0 Model
- Rose 3.0 Model

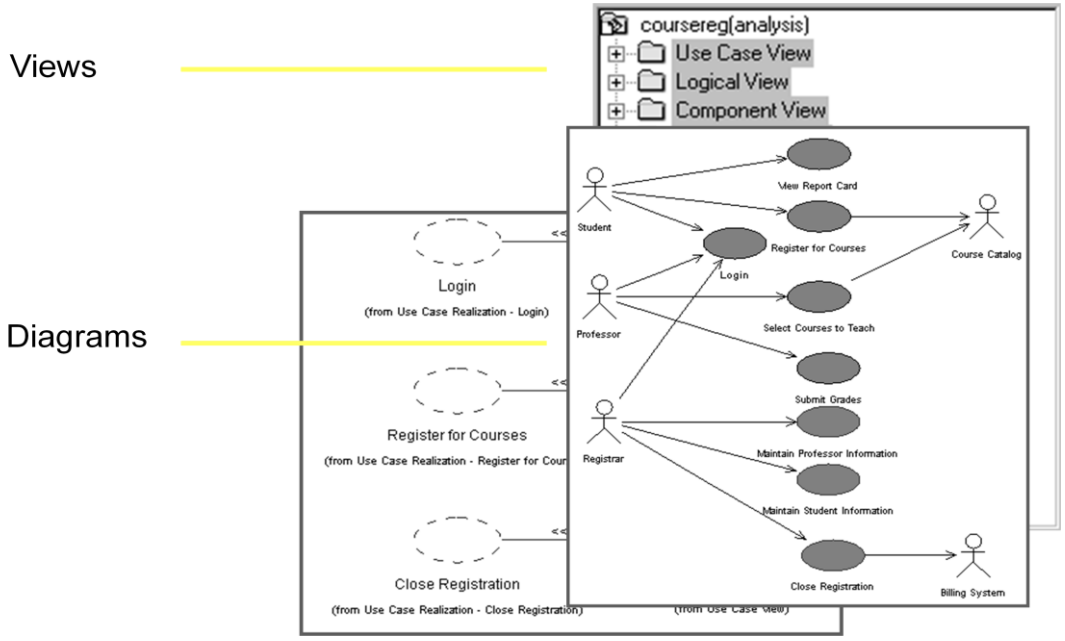
Ak máme jeden formát ktorý preferujeme môžeme modifikovať rose.ini súbor aby ukladal vždy v danom formáte a tým eliminujeme potrebuje používať variantu Save As.

Obrázok 15: Rational Rose Interface



Zdroj: http://www.philadelphia.edu.jo/academics/aghool/uploads/Rational%20ROSE.ppt

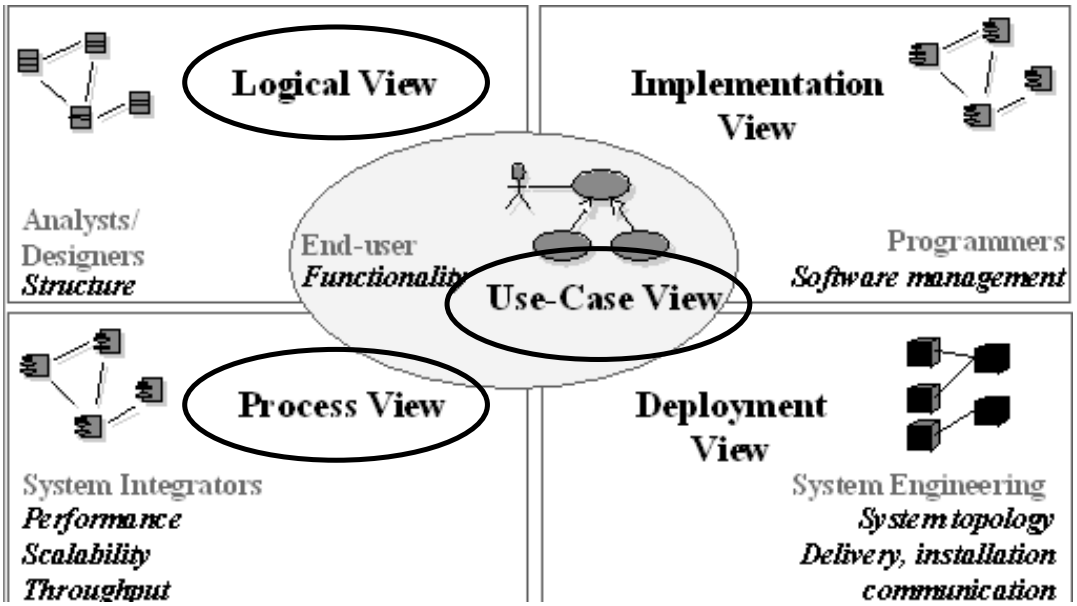
Obrázok 16: Views and Diagrams



Zdroj:

http://www.philadelphia.edu.jo/academics/aghool/uploads/Rational%20ROSE.ppt

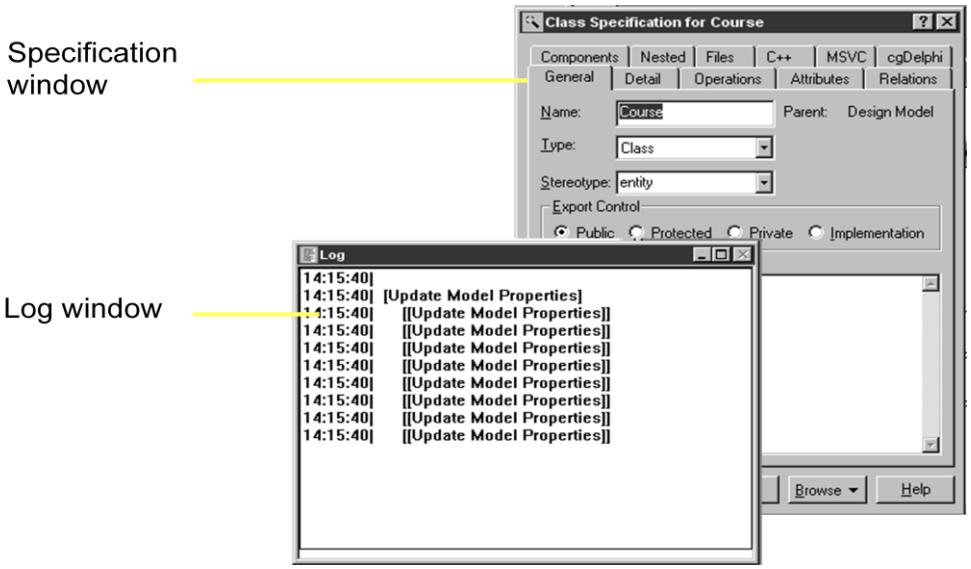
Obrázok 17: The different Views



Zdroj:

http://www.philadelphia.edu.jo/academics/aghool/uploads/Rational%20ROSE.ppt

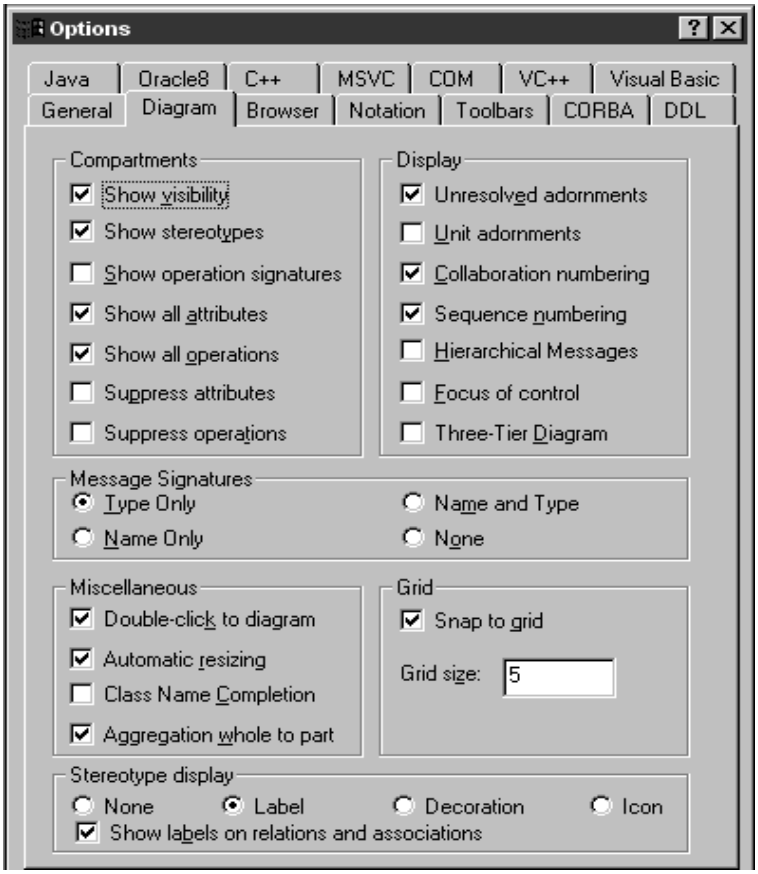
Obrázok 18: Rational Rose Interface



Zdroj:

http://www.philadelphia.edu.jo/academics/aghool/uploads/Rational%20ROSE.ppt

Obrázok 19: Options window



Zdroj:

http://www.philadelphia.edu.jo/academics/aghool/uploads/Rational%20ROSE.ppt

Add IN Manager – Rozšírenie Rational rose umožňuje rýchlo a presne prispôsobiť prostredia v závislosti od vývoja

- Môžeme nainštalovať jazyky napríklad Visual Basic, Visual Java atď.
- Rovnako môžeme nainštalovať nástroje ako Microsoft Project

Doplnky, ktoré môžeme inštalovať:

- Menus (.mnu file)
- Help files (.hlp file)
- Contents tab file (.cnt file)
- Properties (.ptv file)
- Executables (.exe)
- Script files (.ebx script, .ebx kompilovaný script)
- OLE servers (.dll file)

Ako využiť Rational Rose modelovanie v reálnom živote: Obstaráme si Business Process Model. Zmapujeme Use Casel model pre Business Process Model, aby sme si zadefinovali presnú funkcionalitu. Spresníme Use Cases vrátane požiadaviek, poznámok, scenárov, obmedzení a celkové hodnotenie. Zo vstupov a výstupov Business Process modelu a z podrobností o Use Cases, začíname budovať model domény (High level business objects) sekvenčné diagramy, diagramy spolupráce a modely používateľského rozhrania. Od modelu domény, model používateľského rozhrania a scenáre, musíme vytvoriť Class Model. Jedná sa o presné vymedzenie objektov v systéme, ich dáta alebo atribúty a ich správanie alebo operácie. Tým ako sa Class model vyvíja, môže byť rozdelený do oddelených balíčkov a komponentov. Takže Class model a Component Model je postavený aby definoval logické spojenie tried. Súbežne s prácou by sme mali mať dodatočné požiadavky, ktoré by mali byť zachytené a zdokumentované. Deployment model definuje fyzickú architektúru systému.

Budovanie systému sa uskutočňuje vzatím samostatného kúska modelu a priradením ho k jednému alebo viacerým vývojárom. V Use Case prebieha budovanie systému nasledovne. Vezmite samostatných kusov modelu a priradiť k jednému alebo viacerým vývojárov. Use Case potrebujeme prideliť do vývojového tímu, ktorý vybuduje obrazovky, podnikateľské objekty, databázové tabuľky a súvisiace komponenty potrebné na spustenie Use Case.

4.3 Lucidchart

Lucidchart vyvinula spoločnosť Lucid Software Inc. Ide hlavne o cloudový software, čo znamená, že práca s ním je praktizovaná pomocou ľubovoľného internetového prehliadača a pracovné prostredie nám pripomína webovú stránku. Toto riešenie skrýva hneď prvú veľkú výhodu. Touto výhodou je fakt, že používateľ sa nemusí venovať žiadnemu inštalačnému procesu. Jedine čo musí používateľ na začiatku spraviť je zaregistrovať sa a neskôr sa už len prihlasuje. Samozrejme že k práci s Lucidchart je potrebné internetové pripojenie. Pri registrovaní sa nám vytvorí osobný účet, ku ktorému sa vieme neskôr prihlásiť z rôznych počítačov, tabletov či dokonca aj mobilných zariadení. Ako on-line nástroj je Lucidchart schopný pracovať aj na rôznych operačných systémoch ako je Windows, iOS či Linux.

Lucidchart môžeme používať aj offline, teda bez pripojenia na internet a to v prípade, že si ho nainštalujeme ako plugin do prehliadača Google Chrome. Ako on-line nástroj však poskytuje dôležitú výhodu, ktorou je práca viacerých používateľov na jednom projekte v reálnom čase. Túto možnosť majú samozrejme v dnešnej dobe viaceré nástroje. Lucidchart nám všetko ponúka vo veľmi peknej prehľadnej, používateľsky intuitívnej a jednoduchej podobe. Pre využitie na školách je ideálnou voľbou, vďaka jeho dostupnosti.

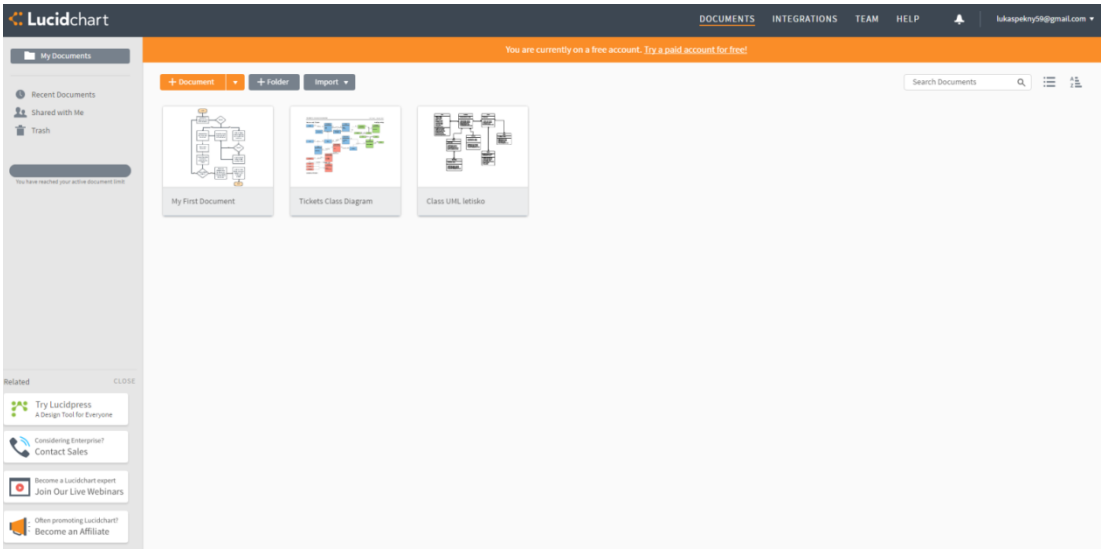
Lucidchart nám poskytuje viacero verzií. Tieto verzie sú Basic, Pro, Team a Free. Mimo verzie Free sú všetky ostatné platené. Ako býva zvykom čím drahšia verzia, tým viac získame možností pre prácu. Dôležitým faktom je, že mimo verzie Team sú všetky verzie jednopoužívateľské. Chýba im tam možnosť multipoužívateľského prístupu k tvorbe projektu. Tvorcovia Lucidchartu mysleli aj na školy a vzdelávacie inštitúcie. Pre tie, sú v prípade záujmu o licenciu verzie Team, zadarmo. Registrácia používateľov, čiže učitelia a študenti týchto školských licencií je možno realizovať iba z jedinečnej URL adresy. Registrovanie z internetovej stránky s takou URL adresou sa automaticky overí, odkial dná škola je. Žiaľ, škola dostáva len obmedzený počet licencií. Ďalšia výhoda spojená so školským účtom je možnosť používať novú aplikáciu Lucidpress pre tvorbu interaktívnych dokumentov.

4.3.1 Možnosti Lucidchartu

Pre tvorbu najrôznejších diagramov sa stačí prihlásiť, čím sa zobrazí *„User interface“* (používateľské rozhranie), v ktorom sa nachádzajú na ľavej strane možnosti,

ktoré ponúkajú tvorbu či import nového projektu. Pod týmto menu je umiestnený zoznam zložiek, kde nájdeme nami vytvorené alebo s nami zdieľané dokumenty, a nechýba ani „odpadový kôs“. Obsah zložiek je ľubovoľne rozšíriteľný kliknutím na možnosť *„Create“*. V strednej časti menu sú k dispozícií konkrétne vytvorené či zdieľané dokumenty – projekty. Na pravej strane sú umiestnené informácie o danej zložke či dokumentu a nachádza sa tu ponuka rôznych operácii, napríklad editovanie, mazanie, kopírovanie či zdieľanie a nastavenie používateľských práv. Ďalšou časťou úvodného prostredia je horná lišta s odkazmi na vybrané dokumenty, fóra a tutoriály, help centrum a nastavenie účtu.

Obrázok 20: Úvodné používateľské rozhranie Lucidchart



Zdroj: vlastné spracovanie

Nový dokument vytvoríme pomocou možnosti *„Create“* s následnou voľbou *„New Document“*. Po tejto akcii sa nám zobrazí najskôr ponuka, kde si zvolíme z už vopred pripravených šablón, ktoré sú rozdelené do rôznych kategórií podľa podstaty projektu, alebo si môžeme vytvoriť svoj vlastný dokument. Po výbere šablóny sa v novom okne prehliadača otvorí už pracovné prostredie pre tvorbu nášho dokumentu. Štandardne je na ľavej strane ponuka rôznych objektov, ktoré môžeme spôsobom „drag and drop“ umiestňovať do samotnej pracovnej plochy. Rovnakým spôsobom je možné objekty prepojovať vytiahnutím šípok z okrajových bodov objektu. Objekty môžeme klasicky upravovať po označení myškou, vpisovať do nich text alebo ich ľubovoľne presúvať. Pre ich ďalšiu úpravu slúži lišta nad pracovnou plochou, kde nájdeme možnosti úpravy textu, farebné nastavenie, pridávanie odkazov či zamykanie/odomykanie objektov. Na pravej strane pracovného prostredia je umiestnený panel s nastavením farebných tém, vlastnosti

pracovnej plochy alebo vlastnosti textu. Ďalej sa tam nachádzajú aj funkcie *„Navigator“* , ktorá ponúka navigáciu v rozsiahlej pracovnej ploche, *„Metrics“*, určená pre prácu s rozmermi objektu, *„History“*, ktorá slúži na zobrazenie histórie a *„Layers“* určená pre prácu s vrstvami. Ďalšou výhodou Lucidchartu je práca na viacerých stránkach, rovnako ako Excel listy, kde sa zobrazujú ako záložky nad pracovnou plochou v rámci jedného projektu, pridávanie komentárov k projektu priamo pri tvorbe, chat s ostatnými spolupracovníkmi, tvorba „hotspotov“ (interaktívny objekt, ktorý na základe udalosti napríklad kliknutie s myškou vykoná určitú akciu, napríklad zobrazí skrytý obsah) alebo nastavenia prepojenia Lucidchat účtu s Google diskom pre ukladanie našich projektov. Pre zdieľanie projektov je v hornej lište umiestnená voľba *„Share“*, ktorá ponúka ďalšie nastavenia. V nastaveniach sú k dispozícií tri položky. Prvá je klasické zdieľanie projektu *„Share“* prostredníctvom mailovej pozvánky či poslaním vygenerovaného URL odkazu. Súčasťou zdieľania je nastavenie práv pre jednotlivých používateľov (písanie komentárov, editovanie). Druhou položkou je *„Publish“*, ktorá umožňuje publikovanie niekoľkými rôznymi spôsobmi (webová stránka, PDF dokument, šablóna, obrázok). Tretia možnosť poskytuje HTML kód pre prípadne publikovanie v rámci webovej stránky (Iframe Objekt). Samozrejme nechýba ani možnosť zdieľať naše projekty na sociálnych sieťach ako Google+, Facebook či Twitter.

4.4 Vytvorenie diagramu

Pri práci s danými produktmi bol zvolený class diagram, teda diagram tried. Tento diagram bol vybraný vďaka svojim vlastnostiam. Predstavuje statický pohľad na modelovaný systém, zobrazuje štruktúru objektových tried a ich názvy, vzájomné vzťahy v systéme. Trieda je definovaná svojimi atribútmi, operáciami a vzťahmi k ostatným triedam, ktoré sú spoločne pre určitú množinu objektov rovnakého typu. Podľa úrovne abstrakcie rozlišujeme tri úrovne class diagramov:

- konceptuálny,
- designový,
- implementačný.

Úlohou konceptuálneho modelu je zmapovať triedy zastupujúce objekty z problémovej oblasti bez implementačných detailov. Ide o implementačne nezávislý model. Pri designovom modeli, sa pôvodné analytické triedy a ich väzby spresňujú a rozširujú. Často bývajú pridávané ďalšie triedy. Designový model obsahuje už niektoré

implementačné charakteristiky. Implementačný model tried obsahuje všetky implementačné charakteristiky pre prípadné generovanie zdrojového kódu vo vybranom programovacom jazyku.

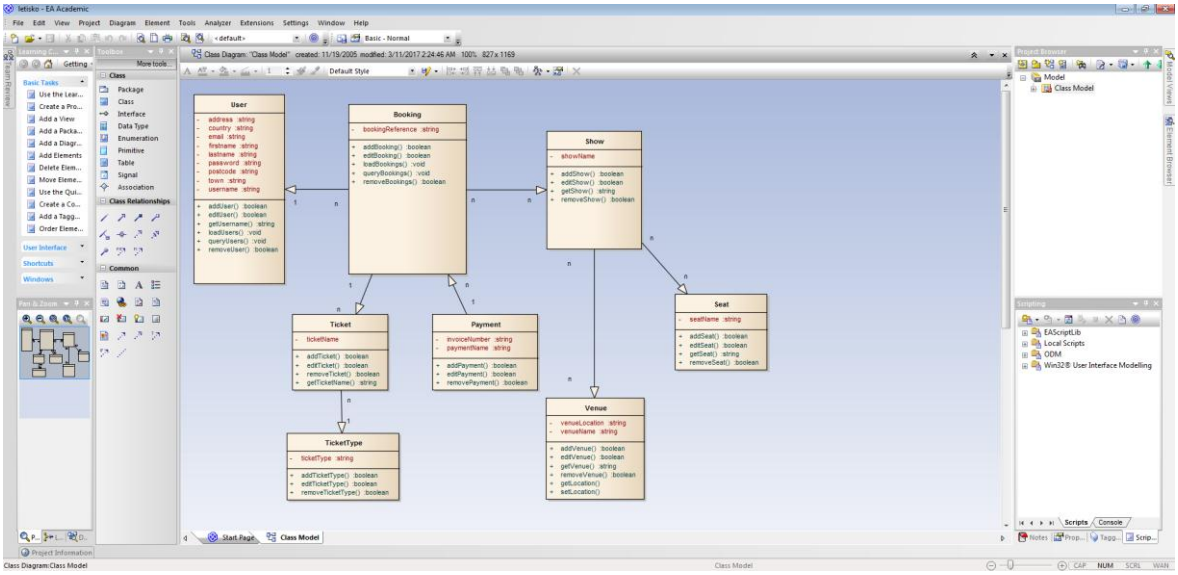
Vytvorenie diagramov v jednotlivých produktoch pozostávalo z niekoľkých krokov. Po otvorení konkrétneho programu som si zvolil typ diagramu, ktorý chcem kresliť a príslušnú technológiu, ktorú budem používať. Po zadaní názvu novo vytvoreného projektu som začal pracovať na tvorbe class diagramu. Výsledné diagramy z jednotlivých produktov sú zobrazené na obrázkoch 21, 22 a 23. Na diagramoch je znázornené fungovanie letiska.

Práca na diagrame v produkte Enterprise Architect bola nenáročná. Počas tvorby diagramu som sa nestretol so žiadnym vážnym problémom. S produktom som bol veľmi spokojný a nechýbala mu žiadna funkcionalita, ktorú som k práci na diagrame potreboval.

Pri práci s produktom Lucidchart som bol milo prekvapený. Po spustení produktu a zvolení konkrétneho diagramu v tomto prípade class diagramu sa používateľovi zobrazí viacero šablón a vzorov rôznych class diagramov. Práca v tomto produkte bola jednoduchá najmä vďaka jeho prehľadnosti a rôznym ponúkaným pomôckam.

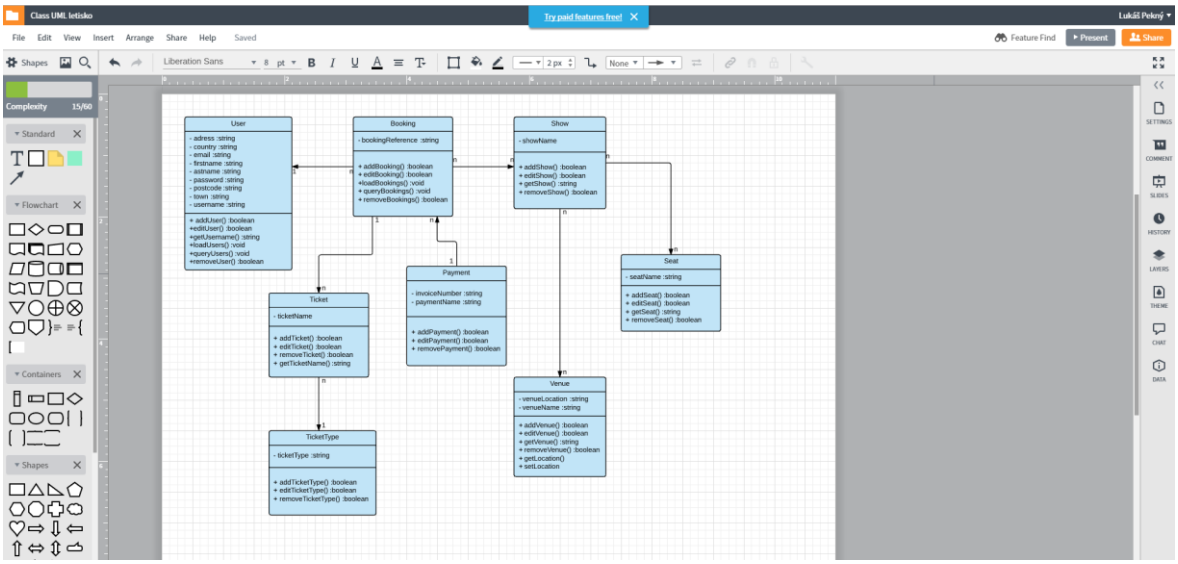
Práca s Rational Rose bola čiastočne neprehľadná, nakoľko produkt prišiel na trh pred viac ako dvadsiatimi rokmi, v roku 1994. S produktom som bol menej spokojný ako pri produktoch Enterprise Architect a Lucidchart.

Obrázok 21: Class diagram v Enterprise Architect



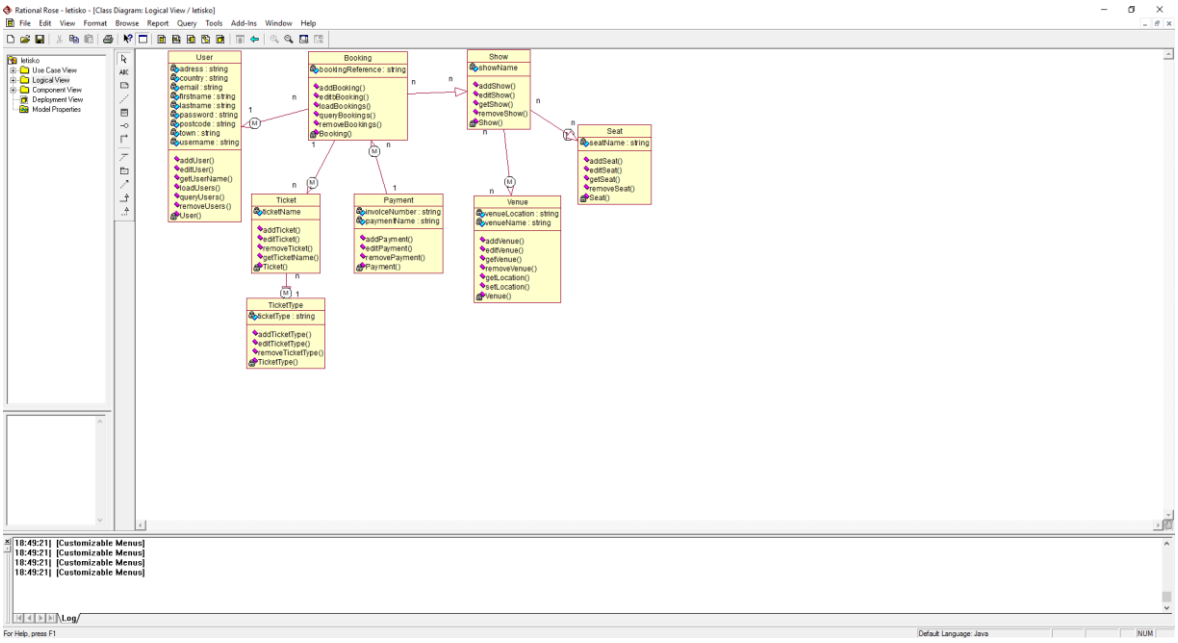
Zdroj: vlastné spracovanie

Obrázok 22: Class diagram v Lucidchart



Zdroj: vlastné spracovanie

Obrázok 23: Class diagram v Rational Rose



Zdroj: vlastné spracovanie

4.5 Technické požiadavky

Technické požiadavky sú na vybrané produkty vo všeobecnosti nízke. V dnešnej dobe by nemal mať žiadny používateľ problém s inštaláciou a spustením týchto produktov, nakoľko súčasné počítače spĺňajú všetky tieto požiadavky.

4.5.1 Technické požiadavky na Enterprise Architect

- Windows – Microsoft Windows10, 8.1, 7, Vista, 2008 Server, 2003 Server alebo XP service pack 2 (verzie 32bit aj 64bit sú podporované)
- Hardware – 2GB pamäte RAM, 300MB miesta na HDD a 1280*720 rozlíšenie na diplayi alebo viac
- Linux – Linux Operating System (kernel 2.4 alebo novší), Wine**1.8 (minimum), 1.9x alebo novší (odporúča sa), Microsoft Data Access Components (MDAC) 2.8
- Hardware – 2GB pamäte RAM, 300MB miesta na HDD a 1280*720 rozlíšenie na diplayi alebo viac
- Mac OS X - Mac OS 10.8 alebo novší, Wine 1.8(minimum), Microsoft Data Access Components (MDAC) 2.8
- Hardware – Intel processor, 2GB pamäte RAM, 300MB miesta na HDD a 1280*720 rozlíšenie na diplayi alebo viac

4.5.2 Technické požiadavky na Rational Rose

- Operačné systémy: Microsoft Windows 2000 Service Pack 4, Microsoft Windows XP Pro Service Pack 1 alebo 2, Windows 2003, Windows XP Service Pack 3, Vista Bussiness Service Pack 1, Windows 7, Solaris 7/8/9/10 a Linux x86 Red hat 8.0 a 9.0
- Hardware – Processer:600 MHZ, RAM:512MB, miesto na pevnom disku 720MB a diplay s rozlíšením 1024x768

4.5.3 Technické požiadavky na Lucidchart

Pri technických požiadavkách produktu Lucidchart je to o niečo jednoduchšie. Pre prácu s týmto produktom potrebujeme mať k dispozícii iba ľubovoľný internetový prehliadač.

5 Výsledky a modelový príklad

Pre správne porovnanie produktov, je nutné si vybrať kritéria, ktoré nám pomôžu v správnom zhodnotení produktov a ich následnom porovnaní. V prvom rade porovnáme kvalitu. Kvalita softvéru je najdôležitejšia. Pri zaobstaraní nekvalitného softvéru sa môžeme stretnúť s vysokými dodatočnými nákladmi na doplnkové inštalácie, servis a podobne. V dnešnom svete sa ale väčšina ľudí rozhoduje hlavne podľa ceny produktu. Je preto dôležité aby pomer cena/kvalita bol v našom porovnaní vyzdvihnutý. Porovnávať budeme tri celé modelovacie produkty. Danými produktmi sú Enterprise Architect od Sparx Systems, Rational Rose od firmy IBM a posledný produkt je Lucidchart.
Produkty budeme porovnávať podľa týchto kritérií:

- cena
- dostupnosť
- kompatibilita s operačnými systémami
- možnosti, ktoré nám daný produkt poskytuje
- UX (User experience)

V tomto odseku budú opísané skúsenosti s týmito tromi produktmi. Ako prvý bol vyskúšaný EA od Sparx Systems. Licencia produktu nám bola poskytnutá univerzitou. Inštalácia softvéru je veľmi jednoduchá, celou inštaláciou nás prevedie installation wizard. Po pár kliknutiach a zadaní licenčného kódu máme program k dispozícii. Interface je veľmi priateľský, pri tvorbe nového projektu nám program dá na výber čomu sa chceme venovať a akú technológiu chceme použiť a kde chceme náš novy projekt uložiť (umiestnenie na disku). Na ľavom paneli máme položky a časti diagramu s ktorými môžeme daný diagram vytvoriť. Pri použití forwardového inžinierstva sa nevyskytli problémy, vybrali sme si v akom jazyku chceme vygenerovaný kód a kam ho chceme uložiť.

Práca s Rational Rose bola zložitejšia. Trial licencia bola poskytnutá spoločnosťou IBM. Celkový interface pôsobil zastarane a práca s bežnými úkonmi bola zbytočne komplikovaná. Rovnako ako pri EA aj v Rational Rose máme pri tvorbe nového projektu možnosť vybrať názov projektu a jeho miesto kde bude projekt uložený avšak tvorba samotného projektu voči EA je zložitejšia pri EA hneď pri spustení programu sa náš softvér spýtal či chceme otvoriť už existujúci projekt alebo chceme vytvoriť nový. Pri Rational Rose nič také sa nám nestane. Tvorbu nového projektu musí inicializovať

samotný používateľ. Ďalší postup je podobný ako pri EA na ľavej strane máme panel ktorý nám ponúka položky pre tvorbu nášho diagramu. Rozdiel pri porovnaní generovania kódu v EA a Rational Rose je, že EA pre každú časť diagramu, triedu či tabuľku generuje vlastný kód a Rational Rose vygeneruje len jeden súbor s kódom. Čiže pri diagrame kde máme 8 tried EA vygeneruje 8 súborov s kódom. Čo je pri kontrole kódu alebo jeho dopĺňaní výhodou, nakoľko je tento spôsob prehľadnejší.

Práca s Lucidchartom bola najjednoduchšia stačilo sa zadarmo registrovať len zadaním e-mailovej adresy. Po prihlásení máte veľký výber šablón a pomôcok ako vytvoriť vlastný nový diagram. Nevýhodou tohto softvéru je, že neposkytuje forwardové ani reverzné inžinierstvo, čiže aj po veľmi ľahkom vytvorení class diagramu, nie je poskytnutá možnosť vygenerovať kód. Lucidchart je milým prekvapením. Je to výborný nástroj pri ktorom sa nestratí ani úplný počítačový laik. Ideálny softvér ak chceme pekne namodelovať UML diagram. Naopak, Rational Rose prekvapil v zlom. Nie moc príťažlivý a prehľadný user interface. Zbytočné komplikácie s jednoduchými úkonmi a celkovo čo sa týka vizuálnej stránky je na tom z pomedzi týchto troch softvérov najslabší. Od Enterprise Architectu môžeme čakať veľmi dobré možnosti pri tvorbe UML diagramov, avšak chýbajú šablóny a vzory aké ponúka Lucidchart. EA je ale určený už pre zručnejších používateľov.

5.1 Modelový príklad

Odporučíme produkt pre dané tri skupiny pričom zohľadníme všetky vyššie uvedené kritéria. Skupina A predstavuje školský sektor kde budú produkt využívať na výučbu modelovania študentov. Skupina B predstavuje stredne veľkú firmu, ktorá ma okolo 50 zamestnancov. Skupina C bude predstavovať veľkú spoločnosť, v ktorej pracuje viac ako 200 zamestnancov.

Pre prvú skupinu, ktorá predstavuje školský sektor, by sme mohli odporučil produkt Lucidchart nakoľko je veľmi ľahko dostupný (verzia free je zadarmo), no ak by mal používateľ záujem základná verzia stojí 4,95$ mesačne a poskytuje veľmi dobré možnosti pri tvorbe UML diagramov. Z týchto troch produktov má Lucidchart najlepší User Experience (UX), poskytuje veľké množstvo vzorových ukážok a návrhov, ktoré nám vedia pomôcť. Na druhej strane, nevýhodou Lucidchartu je, že neposkytuje reverzné a forwardové inžinierstvo. Tým pádom je možné povedať, že Lucidchart by bol nepostačujúcim v školách, ktoré ponúkajú študijné odbory zamerané na programovanie.

Pre tieto odborné školy by bol vhodnejšou voľbou Enterprise Architect. Síce nie je bezplatne dostupný, na rozdiel od Lucidchart, ale okrem výborných možností v tvorbe UML diagramov a priateľskému UX poskytuje aj už spomínané reverzné a forwardové inžinierstvo.

Pre druhú skupinu, ktorá predstavuje stredne veľkú firmu, by bol ideálny softvér Enterprise Architect. Professional verzia tohto produktu stojí podľa cenníka od spoločnosti Sparx Systems 199$. Táto verzia podporuje plné vybavenie pre tvorbu UML diagramov pre pracovné skupiny, vývojárov a analytikov.

Enterprise Architect by bol vhodný aj pre skupinu C, čiže veľkú spoločnosť, ktorý zamestnáva väčší počet zamestnancov. V tomto prípade by bola vhodná Suite edícia softvéru, ktorá stoji 699$. Poskytuje kompletný Enterprise Architect cez rôzne domény. Keby sme si v tomto prípade chceli zvoliť Rational Rose, náklady na nákup licencií by boli vysoké. EA je celosvetovo na trhu, čo sa týka Architecture Management, číslo jedna s viac ako 580 000 predanými licenciami, podporovaný viac ako 230 partnermi v 160 krajinách a s priemerným hodnotením zo strany užívateľov 7.8. Rational Rose je na desiatej priečke v Architecture Management a neuvádzajú ani počet predaných licencií, ani hodnotenia zo strany svojich používateľov.

Veľkú váhu pri výbere softvéru majú ceny. Tu je nutné spomenúť aj skvelý marketing zo strany Sparx Systems. Pri kúpe viacerých licencií, cena jednotlivých licencií klesá. Naopak, pri kúpe licencie Rational Rose od spoločnosti IBM, získame len nonstop podporu cez e-mail po dobu jedného roka.

Ďalšia vlastnosť, ktorá by nás mala zaujímať, je kompatibilita s operačnými systémami a hardvérové požiadavky. Tieto produkty majú natoľko nízke hardwarové požiadavky, že na súčasných počítačoch nie je problém s ich spustením. Na spustenie Lucidchartu potrebujeme len ľubovoľný internetový prehliadač.

Záver

V tejto diplomovej práci som mal za úlohu porovnať modelovacie nástroje súčasnosti. Pre túto úlohu som si vybral Enterprise Architect ako jednotku na trhu, Rational Rose od firmy IBM a Lucidchart ako modelovací nástroj založený na cloud princípe. Preto som v teoretickej časti popísal dôležité fakty ohľadom jazyka UML, ktoré som v praktickej časti využíval.

Výsledkom mojej práce je prehľad jazyka UML, ktorý som čerpal z uvedených kníh, charakteristika a následne porovnanie vybraných produktov podľa vybraných kritérií s cieľom vedieť odporučiť optimálny produkt danému zákazníkovi. Produkty som porovnával z pohľadu ceny, dostupnosti, požiadaviek, funkcionality a celkového dojmu (User Experience). Pri práci s danými produktmi som vytváral rôzne diagramy, skúšal grafické znázornenia a tvorbu zdrojového kódu. Po dokončení porovnania som zhodnotil, že Enterprise Architect je výborná voľba pre veľké ale aj stredne veľké podniky. Rational Rose by som osobne neodporúčal nikomu a Lucidchart je výborná voľba pre školy alebo malé podniky, ktoré potrebujú rýchlo a pekne nakresliť návrhy, diagramy a podobne. Osobne mi táto práca priniesla veľmi cenné skúsenosti z oblasti práce s modelovacími nástrojmi a plánujem sa tejto problematike venovať aj v budúcnosti.

Zoznam použitej literatúry:

AMBLER, S. – NALBONE, J. – VIZDOS, M., (2006). *The Enterprise Unified Process: Extending the Rational Unified Process*. Prentice Hall PTR, 2006. 408 s. ISBN: 0-13-191451-0

AMBLER, S., (2005). *Elements of UML 2.0 style*. Cambridge University Press, 2005. 200 s. ISBN: 0-521-61678-6

ARLOW, A. J. – DUFFY, C. J. – MCDERMID, J. A., (2006). *Safety specification of the active traffic management control system for English motorways*. [online] In: System Safety, 2006. The First Institution of Engineering and Technology International Conference on. IET, 2006. p. 10 pp. 54-63. [cit. 8.4.2017] Dostupné na: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.483.8439&rep=rep1&type=pdf>

ARLOW, J. – NEUSTAD, I., (2007). *UML 2 a unifikovaný proces vývoje aplikací. Objektově orientovaná analýza a návrh prakticky*. Computer Press, 2007. 568 s. ISBN: 978-80-251-1503-9

BOGER, M. – GRAß, E. – KÖSTER, M., (2000 – 2007). Poseidon for UML. *[online] [cit. 13.4.2017] Dostupné na:* <http://www.gentleware.com/fileadmin/media/archives/userguides/poseidon_users_guide/userguide.html>

FOWLER, M. *Destilované UML*. Grada, Praha, 2009. 176 s. ISBN: 978-80-247-2062-3

KANISOVA, H. – MULLER, M. (2006). *UML srozumitelně*. Computer Press, 2006. 176 s. ISBN: 80-251-1083-4

KANISOVA, H. – MULLER, M., (2007). *UML srozumitelně.* 2.vydání.Computer Press, 2007. 176 s. ISBN: 80-251-1083-4

KLEPPE, A. – WARMER, J. – BAST, W., (2003). *MDA Explained: The Modern Driven Architecture: Practice and Promise.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.171 s. ISBN:032119442X

MCNEISH, K. (1993 – 2017). *UML Collaboration Diagrams* [online] [cit. 13.4.2017] Dostupné na: http://www.codemag.com/Article/0205051

MUKERJI J., – MILLER, J. (2003). *MDA Guide V1.0.*, [online] [cit. 3.4.2017] Dostupné na: < http://www.omg.org/cgi-bin/doc?omg/03-06-01>

QUATRANI, T., (1998). *Visual modeling with Rational Rose and UM.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 144 s., ISBN:0-201-31016-3

RUSSELL, M., (2006)*Learning UML 2.0*. RepKover, 2006. 288 s. ISBN: 0-596-00982-8

SCHMULLER, J. (2004) *Sams Teach Yourself UML in 24 Hours, Complete Starter Kit* . 3rd Edition, Sams publishing, 2004. 504 s. ISBN 0-672-32640-X

Internetové zdroje:

Class Diagram, (2017). [online] [cit. 2.5.2017] Dostupné na: <https://www.lucidchart.com/pages/uml/class-diagram>

Enterprise Architect for Business Analysts (2006-2015) [online] [cit. 20.4.2017] Dostupné na: <http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/353/Enterprise-Architect-for-Business-Analysts.aspx>

Foundational UML Reference Implementation, 2017. [online] [cit. 9.4.2017] Dostupné na:<http://portal.modeldriven.org/content/foundational-uml-reference-implementation>

MDA - The Architecture Of Choice For A Changing World, (1997 – 2017). [online] [cit. 21.3.2017] Dostupné na: <http://www.omg.org/mda/>

Rational Software Architect Standard Edition, 2017. [online] [cit. 14.3.2017] Dostupné na: <http://www-142.ibm.com/software/products/sk/sk/swarchitect-standard>

Sequence Diagram, (2001 – 2009). [online] [cit. 19.4.2017] Dostupné na: <http://argouml-stats.tigris.org/documentation/manual-0.22/ch19.html>

UML 2 Deployment Diagrams, (2003-2014). [online] [cit. 19.4.2017] Dostupné na: <http://www.agilemodeling.com/artifacts/deploymentDiagram.htm>

UML 2 Use Case Diagrams,, (2003-2014). [online] [cit. 19.4.2017] Dostupné na: <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>

UML, (2009) [online] [cit. 2.5.2017] Dostupné na: <https://sites.google.com/site/sureshdevang/uml>