Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
študijný program: POČÍTAČOVÉ SYSTÉMY A SIETE

Podpora vzdelávania v predmete Špecifikačné a opisné jazyky

Dokumentácia k predmetu:
Tímový projekt I - zimný semester

Členovia tímu: Bc. Peter Koine, Bc. Šimon Hupka, Bc. Ján Turoň, Bc. Michal Choleva, Bc. Emil Moťovský
Vedúca tímového projektu: Ing. Katarína Jelemenská, PhD.
Ročník, typ štúdia: 1., Inžinierske štúdium

Obsah

0 Úvod

0.1 Účel dokumentu

Tento dokument vznikol ako dokumentácia k Tímovému projektu 1. Obsahuje správu k téme: Podpora vzdelávania v predmete Špecifické a opisné jazyky. Dokument je určený pre vedúcu tímového projektu a pre konkurenčný tím.

Účelom tohto dokumentu je poskytnúť čitateľovi prehľad v problémovej oblasti, priblížiť jednotlivé e-learningové metódy, Petriho sieť, funkcie Tablet PC a na záver aj predpokladaný návrh riešenia.

0.2 Prehľad

Prvá kapitola obsahuje zadanie projektu, jeho ohraničenia a vymedzenia.

Analýza projektu je obsiahnutá v druhej kapitole. Obsahuje analýzy viacerých oblastí, ako E-learning a jeho štandardy, formy a existujúce riešenia, ďalej analýzu záverečných projektov, základných vlastností Petriho sietí a rôzne nástroje na prácu. V nasledujúca podkapitola obsahuje analýzu Tablet PC a zhodnotenie analýzy.

Treťou kapitolou je riešenie, kde definujeme špecifikáciu požiadaviek na aplikáciu a taktiež hrubý návrh riešenia. Poslednou kapitolou je použitá literatúra.

0.3 Slovník pojmov

0.3.1 E-learning

E-Learning je vzdelávací proces, ktorý využíva informačné a komunikačné technológie na tvorbu rôznych obsahov k štúdiu. V texte budeme používať aj označenie elearning.

0.3.2 Tablet PC

Týmto názvom budeme v dokumentácií označovať počítač s dotykovou obrazovkou, na ktorom budeme projekt realizovať.

0.3.3 Open Source

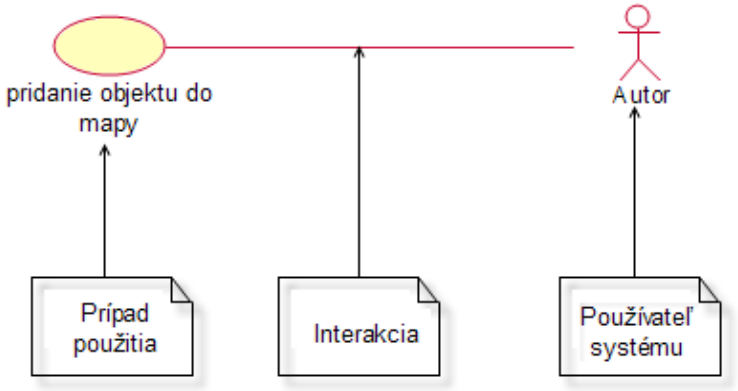
Anglický výraz pre voľne šíriace sa aplikácie, t.j. aplikácie, ktoré sú dostupné zdarma.

0.4 Skratky

ŠpOJ	Špecifikačné a opisné jazyky
FIIT STU	Fakulta informatiky a informačných technológií, Slovenská technická univerzita

0.5 Notácia

Notácia pri diagrame prípadov použitia:



1 Zadanie

Analyzujte existujúce materiály, aplikácie a systémy, vytvorené pre podporu predmetu Špecifikačné a opisné jazyky. Analyzujte tiež dostupné vzdelávacie systémy s podobným zameraním. Pri analýze sa zamerajte najmä na podporu získavania a overovania praktických zručností študentov.

Na základe analýzy navrhnite a implementujte e-learningové moduly (prípadne externé aplikácie) pre výučbu predmetu Špecifikačné a opisné jazyky, ktoré budú podporovať správu a vyhodnocovanie zadaní a zabezpečovať overovanie získaných vedomostí a praktických zručností študentov v rámci predmetu.

1.1 Ohraničenia a vymedzenia

Priestor pre realizáciu vidíme najmä v téme Petriho siete, nakoľko táto problematika nebola ešte ekvivalentným spôsobom riešená a v predmete Špecifikačné a opisné jazyky chýba. Po konzultáciách s Ing. Jelemenskou, kde sme si spresnili požiadavky a ciele nášho návrhu, sme vytvorili hrubý návrh projektu.

Plánovaný produkt možno charakterizovať ako komplexný elektronický systém na precvičovanie a overovanie vedomostí o Petriho sieťach. Vzhľadom k súčasným trendom a dostupným prostriedkom navrhujeme produkt implementovať ako modul, web aplikáciu, prostredníctvom existujúceho systému Moodle, prípadne obdobného systému zameraného na podporu výučby. Produkt je špecializovaný na Petriho siete. Preto je schopný poskytnúť novú funkcionalitu, ktorú zvyčajné e-learningové systémy neponúkajú.

Systém poskytne intuitívne rozhranie pre vyučujúceho, ako aj pre študenta. Samozrejmosťou sú prostriedky pre manažment systému. Kontá pre študentov, vytváranie tried alebo iná vhodná forma organizácie, aby sa dal systém prakticky využiť pri vyučovaní predmetu ŠpOJ.

Vyučujúci má možnosť jednoduchou formou zadávať otázky, ktoré slúžia študentom na precvičovanie nadobudnutých vedomostí. Tieto môžu byť použité aj na zostavenie bodovaných (percentuálne hodnotených) testov.

Otázky môžu mať formu:

- otázka s výberom odpovede
- otázka s doplnením odpovede
- úloha vyžadujúca zostavenie alebo upravenie Petriho siete (grafické znázornenie Petriho siete)

Všetky otázky sú systémom automaticky opravené a vyhodnotené. Toto sa týka aj otázok vyžadujúcich tvorivý prístup študenta, ktoré majú zvyčajne viac alebo nekonečne veľa správnych riešení. Správnosť týchto otázok je vyhodnotená prostredníctvom niektorej z metód overenia správnosti, ktoré sme už využili v predchádzajúcich projektoch:

- P-invariant
- Stremersch
- Strom dosiahnuteľnosti

Hlavnou výhodou strojového opravovania testov je úspora času. Ďalšou výhodou je, že študenti okamžite po ukončení testu vedia nakoľko boli úspešní pri riešení. Systém má byť postavený tak, aby zapadal medzi existujúce riešenia, prípadne bol s nimi integrovateľný alebo inak prepojiteľný či rozšíriteľný. Nemá zbytočne duplikovať existujúcu funkcionalitu, ale prinášať novú kvalitu. Chceme, aby systém nebol iba funkčný, ale bol aj spoľahlivý a stabilný. Chceme si byť istí, že deklarovaný prínos nášho produktu nie je iba zdanlivý, ale aj preukázateľný v praxi.

2 Analýza

2.1 E-Learning

E-Learning je vzdelávací proces, ktorý využíva informačné a komunikačné technológie na tvorbu kurzov, k distribúcií študijného obsahu, komunikáciu medzi študentmi a pedagógmi a k riadeniu štúdia.

2.1.1 Štandardy

Všetky elearningové systémy majú štruktúru odvodenú podľa určitých štandardov vytvorených autoritami uvedenými nižšie.

2.1.1.1 AICC

Aviation Industry Computer-Based Training Committee (AICC, komisia pre počítačovo založené tréningové kurzy pre letecký priemysel) vytvára všeobecné špecifikácie a štandardy, aby výrobcovia e-learningových technológií mohli rozširovať svoje pôsobenie na viacerých trhoch, čo znižuje cenu týchto produktov. Jeden z ich štandardov je využívaný v LMS (Learning Management Systems). Mnohými je považovaný za bezpečnejší a robustnejší štandard ako SCORM [5].

2.1.1.2 Ariadne

Európska nadácia vytvorená na tvorbu nástrojov a metodík pre manažovanie a znovu použitie pedagogických a tréningových materiálov. Základným prvkom štruktúry vytvorenej touto nadáciou je tzv. KPS (Knowledge Pool System, teda systém vedomostnej studne). KPS je digitálna knižnica obsahujúce rôzne dokumenty v elektronickej forme. Táto databáza však nie je priamo prístupná. Na komunikáciu koncového užívateľa s ňou je potrebný príslušný klient, ktorý potom dovolí vkladanie a pozeranie dokumentov [2].

2.1.1.3 IEEE

IEEE (Institute of Electrical and Electronics Engineers**)** je medzinárodná nezisková organizácia. Jej hlavné smery pôsobenia sú energia, zdravie, informačné technológie, telekomunikácie, transport a nanotechnológia. Zastrešuje obrovské množstvo materiálov a má viac ako 360 000 členov. Stojí za vytvorením viac ako 900 medzinárodných štandardov [4].

2.1.1.4 IMS GLOBAL

Bol vytvorený skupinou organizácií. Jeho hlavnou výhodou je, že bol vytváraný ako neprofitový. Snaží sa vyzdvihnúť adaptáciu učebných techník. Týmto spôsobom chce použiť už vytvorené, rokmi overené spôsoby výučby, ktoré sa snaží pretransformovať do elektronickej formy. Ďalej zahŕňa špecifikáciu výmeny údajov medzi účastníkom kurzu a systémom [3].

2.1.1.5 SCORM

SCORM (Sharable Content Object Reference Model, model zdieľaného obsahu objektových referencií) je kolekcia štandardov a špecifikácií pre e-learning. Prešiel viacerými vývojovými stupňami. Terajšia verzia 1.3 označovaná aj ako SCORM2004. Nová verzia priniesla so sebou možnosť vytvoriť na seba nadväzujúce úlohy a možnosť zdieľať informácie o úspešne vyriešených úlohách. Ďalej umožňuje individuálne spravovanie každého materiálu a zvyšuje aj kompatibilitu s väčším počtom materiálov. Samotný systém používa XML manifesty na vytvorenie a udržiavanie obsahu [1].

2.1.2 Technológie

Všetky existujúce systémy boli vytvorené pomocou určitej technológie. Vo väčšine prípadov dokonca s použitím viacerých z dostupných riešení. Keďže sú tieto elearningové systémy vo veľkej miere globálne dostupné a určené pre množstvo ľudí, tak sú upravované na prístup cez Internet. Avšak, ak chcú spracovávať veľké množstvo

informácií a odlišných zdrojov, prípadne vytvárať rôzne testy a iné formy skúšania, tak sa nemôže použiť HTML, lebo nepodporuje prácu s databázami a vyhľadávanie v nich. HTML je vhodné na vytvorenie jednoduchého e-learningového systému, ktorý obsahuje už vyselektované informácie.

Na vytvorenie komplexnejšieho riešenia sa používa jazyk PHP. Všetky materiály a kurzy je potom možné ukladať do databázy a taktiež s ňou aj pracovať. Avšak samotné PHP tiež nie je dobrým riešením, lebo tak ako HTML, neobsahuje spôsoby, ako spríjemniť a zefektívniť účastníkovi kurzu jeho štúdium. Preto sa používajú rôzne animácie vytvorené flash technológiou, prípadne technológiou silverlight. Technológia silverlight nie je však zatiaľ tak globálne rozšírená a nemá takú obľubu ako technológia flash.

Ďalším možným riešením je použitie jazyka Java. S pomocou tohto programovacieho jazyku je možné vytvoriť aplikácie, ktoré dokážu pomôcť s vyriešením určitého problému. Na vytvorenie elearningového systému by bolo potrebné vytvoriť obrovský program, ktorý by nebolo možné potom jednoducho rozšíriť. Preto je použitie Javy vhodné len na ukázanie špeciálnych prípadov, prípadne na vytvorenie pomôcok, s ktorými bude študent pracovať.

Iné programovacie jazyky ako C/C++ nie je možné spúšťať priamo z Internetu. Využívané sú na naprogramovanie externých pomocných programov zameraných na jednu špeciálnu oblasť a funkciu.

E-learning však neznamená len to, že sa musí jednať striktne len o určitý systém. Partia sem aj materiály, ktoré sú v elektronickej forme ako napríklad PDF, DOC, atď., z ktorých sa študent môže niečo naučiť.

Keď sa zoberú do úvahy jednotlivé technológie, ich plusy a mínusy, tak na vytvorenie elearningového systému je potrebná vhodná kombinácia viacerých technológií. Najčastejšia kombinácia je použitie PHP, flash, PDF a Java technológií.

2.1.3 Formy

Každý momentálny elearningový systém je špecializovaný na určitý typ výučby. Preto sa vytvorili skupiny (formy), do ktorých sa tieto programy môžu zaradiť. Nižšie sú uvedené najznámejšie a najpoužívanejšie formy.

2.1.3.1 LMS (Learning Management Systems)

Je to softvér na doručenie a riadenie tréningu. Tento softvér zahŕňa jednoduché systémy pre riadenie tréningu až po distribuovanie kurzov cez Internet, pričom jeho hlavné úlohy sú automatizovaná registrácia členov triedy a „online“ kurzov, taktiež riadenie tréningu manažmentu, vykonávanie rôznych testov na preskúšanie a samozrejme materiály pre samoštúdium. LMS je založené na viacerých platformách (napr. Java) a využíva robustnú databázu. Väčšina LMS systémov sú však komerčné [7].

2.1.3.2 CMS (Content Management System)

Počítačová aplikácia používaná na vytvorenie, editovanie, riadenie a vydávanie obsahu a to tak, aby to bolo čo najlepšie organizované. Je využívaný najmä na spracovanie novinových článkov, technických manuálov, marketingových brožúr a predajných príručiek. Taktiež môže obsahovať obrázky, audio súbory, video súbory a iné počítačové súbory. Výhodou tohto systému je, že povoľuje zachytenie obsahu, napríklad skenovaním alebo automatizované nastavovanie farby a fontu podľa obsahu [6].

2.1.3.3 VLE (Virtual Learning Environment)

Taktiež softvérový systém, avšak vytvorený na podporu vyučovania pomocou tzv. virtuálnych tried a na rozdiel od MLE, ktorý je zameraný na manažment. VLE je vytvorený na používanie cez Internet a poskytuje kolekciu nástrojov, ako napríklad nástroje na testovanie, komunikáciu alebo administráciu skupín. Do tohto systému boli implementované aj ďalšie novinky. Medzi najatraktívnejšie patria 3D virtuálne učebne a RSS. Pôvodne bolo VLE vytvorené ako pomôcka pre dištančné štúdium, ale momentálne sa najviac používa ako doplnok k normálnej výučbe [8].

2.1.4 Normálna forma výučby vs. e-Learning

Súčasné prostredie plné zmien v dôsledku kontinuálneho rozvoja informačných technológií dáva do popredia otázku, či by nebolo vhodné upriamiť sa na novšie a prepracovanejšie spôsoby výučby. Spomedzi troch na Slovensku využívaných metód – tradičný spôsob výučby, učenie doma tzv. „homeschooling“ a elektronické učenie tzv. „e-learning“ - by sme chceli objektívne porovnať dve najviac uplatňované, a to tradičný a elektronický spôsob výučby.

Tradičné vyučovanie nám je veľmi dobre známe, sprevádza nás od prvej triedy na základnej až po štúdium na vysokej škole, preto na základe našich dlhoročných skúseností vieme zhodnotiť jeho klady a zápory. E-learning je na Slovensku pomerne čerstvá novinka, ktorá sa postupne snaží zaujať významné miesto na mnohých vysokých školách (on-line výučba predmetov alebo podpora k výučbe predmetov, kurzy pre študentov, a pod.), ale aj vo firmách (on-line kurzy pre zamestnancov). V USA je však veľmi rozšírený, preto sa dajú zhodnotiť klady a zápory aj tejto metódy.

Existuje veľa dôvodov, prečo sa na základných a stredných školách uprednostňuje tradičná metóda vyučovania, a tým hlavným je určite socializácia, teda proces začleňovania jednotlivca do skupiny/spoločnosti a formovanie jeho osobnosti. Skupina (v tomto prípade trieda) dokáže nezanedbateľne ovplyvniť správanie jednotlivca, napr. motiváciou byť najlepší z triedy alebo umožňuje žiakom učiť a inšpirovať sa jeden od druhého – z cudzích chýb a úspechov, pozorovaním. Len v skupine sa naučíme, ako spolupracovať, správne komunikovať, riešiť konflikty, vyjednávať, stáť si za svojim názorom a brániť si ho, ale aj pristúpiť na kompromis alebo si pripustiť omyl. S týmito a podobnými situáciami sa však v e-learningu nestretneme. Ak by aj áno, stále tam však chýba nenahraditeľný osobný kontakt.

Ďalším kladom je vzájomná interaktivita učiteľa a študenta, okamžitá spätná väzba (overenie si nadobudnutých vedomostí, informácií), na hodine je prípustná živá diskusia, otázky pri nepochopení nejakých súvislostí a dodatočné vysvetlenie. Humor a určitý stupeň kreativity môže byť vítaný (napr. nejaké obdobie z histórie môže byť žiakmi zinscenované ako divadelné predstavenie, a tak sa toto učivo ľahšie vryje žiakom do pamäti), hodiny potom nie sú monotónne a stereotypné.

Úroveň predmetu je však meraná hlavne kvalifikáciou a skúsenosťami učiteľa (veľakrát býva slabým článkom), je dôležité, aby vedel zaujať, dobre podať kvantum informácií a aby obsah svojich prednášok priebežne aktualizoval. Očakáva sa tiež od

neho, že bude vedieť „ukorigovať“ veľký počet žiakov v triede. Musí vedieť odhadnúť povahu študentov, včas reagovať na sociálne nedostatky, aby nevznikali škodlivé konflikty medzi študentmi navzájom. Je to z dôvodu, že mimo domova, hlavne v skoršom veku študentov, práve on nahrádza rodičov. Ak túto úlohu nezvládne, môže byť študent nepriaznivo ovplyvnený, ak sa nachádza v zlom prostredí.

E-learning má jednu obrovskú výhodu, umožňuje človeku učiť sa týmto spôsobom prakticky kdekoľvek a kedykoľvek, potrebuje však primerané technické prostriedky (osobný počítač, pripojenie do Internetu). Ďalšia výhoda je možnosť simulácie komplexných procesov, do ktorých môže účastník elektronického vzdelávania zasahovať a ovplyvňovať tak ich vývoj [9]. Avšak samotná tvorba týchto aplikácií a programov je finančne, časovo a obsahovo dosť náročná, z dlhodobejšieho hľadiska sa počiatočné investície vrátia tým skôr, čím viac účastníkov sa do kurzu zapojí. Na druhej strane, prevádzkové náklady sú nižšie v porovnaní s tradičným vzdelávaním (netreba tu financovať učiteľa, ani priestory a prostriedky, školské materiály a pomôcky, a pod.) [9].

Slabým miestom v tomto prípade môže byť študent, ktorý je síce pánom svojho času, sám si naplánuje, kedy sa do on-line vzdelávania pustí, akú prednášku si pozrie, aký test si spraví (prínosom je objektívna spätná väzba, testovacie nástroje e-learningu umožňujú objektívne nastaviť požadované ciele kurzu a vytvárať tak štatistiky o jednotlivých študentoch: koľko dosiahli bodov v zadaných testoch, ako odpovedali na otázky a nad ktorými časťami učiva strávili najviac času), a pod., zvolí si aj vlastné tempo [10]. No musí byť motivovaný, mať na zreteli svoj cieľ a chcieť sa učiť, pretože ak nemá chuť a záujem sa učiť, je to pre neho len strata času.

Je veľmi zložité vyzdvihnúť na piedestál práve jednu metódu, ktorú určíme za najlepšiu. Obe majú svoje výhody aj nevýhody, avšak veríme tomu, že pokiaľ sa do e-learningu podarí začleniť aj socializácia, prostredníctvom video konferencií a pod., určite sa táto metóda raz dostane do popredia.

2.1.5 Existujúce riešenia

Z existujúcich riešení sme vybrali tri z komerčného prostredia a tri tzv. „open source“, ktoré sme rozanalyzovali a uviedli ich základné vlastnosti a funkcie.

2.1.5.1 Claroline

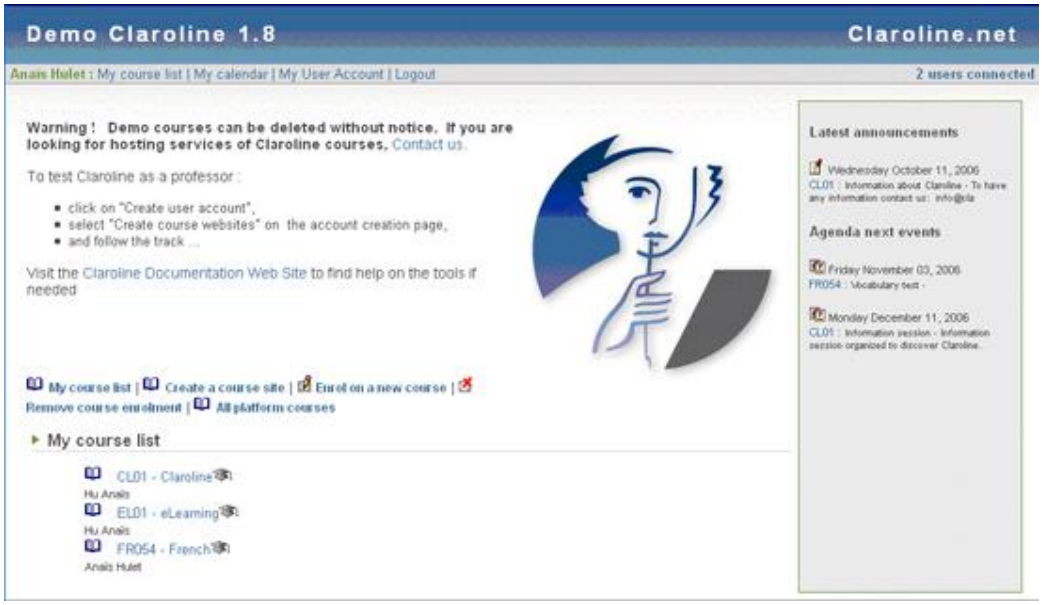
Systém Claroline je vydaný pod licenciou GPL. Čiže sa jedná o systém typu open-source. Používajú ho stovky organizácií vo svete v 93 krajinách. Na Slovensku sú to napríklad Fakulta hospodárskej informatiky Ekonomickej univerzity v Bratislave a Prírodovedecká fakulta UK. Je dostupný v 35 jazykových verziách. Dovoľuje vytvárať kurzy, zabezpečuje ich administráciu.

Funkcie, ktoré poskytuje sú napríklad:

- Vytvoriť kurz a napísať k nemu popis
- Publikovať dokumenty v ľubovoľnom formáte
- Administrovať fóra ktoré prislúchajú ku kurzu
- Vytvárať skupiny študentov
- Vytvárať online testy
- Manažovať program kurzu, zadávať termíny splnenia úloh
- Vytvárať oznamy (tieto systém môže rozoslať aj e-mailom)
- Zadeľovať úlohu jednotlivým študentom
- Prezerať si štatistiky aktivít používateľov
- Používať vstavaný chat
- Vytvárať a editovať wiki dokumenty

Claroline nepoužívajú len školy a univerzity, ale tiež mnoho spoločností. Platforma je prispôsobiteľná, ponúka flexibilné a prehľadné pracovné prostredie.

Je postavený na technológiách ako PHP, MySQL a používa štandardy SCORM a IMS/QTI. Je kompatibilný so systémami Mac OS, Windows a GNU/Linux. Poskytuje intuitívne a jednoduché prostredie pre administráciu. Takže nevyžaduje nejaké rozsiahle skúsenosti s administráciou podobných systémov. Jeho inštalácia je jednoduchá a rýchla a funguje v akomkoľvek prehliadači [11]. Na **Obr. 1** vidíme domovskú stránku systému Claroline.



Obr. 1 – Úvodná stránka systému Claroline

2.1.5.2 Moodle

Moodle je voľne dostupná e-learningová softvérová platforma (tiež známa ako Learning Management Systems – LMS). Má širokú používateľskú základňu – 38896 registrovaných stránok s 16927590 používateľmi v 1713438 kurzoch. (Údaje sú z januára roku 2008.)

Slovo Moodle bolo pôvodne akronymom pre Modular Object-Oriented Dynamic Learning Environment (modulárne objektovo orientované dynamické výukové prostredie); táto informácia môže byť zaujímavá predovšetkým pre programátorov a teoretických pedagógov. Možno ho tiež považovať za sloveso, ktoré popisuje proces voľného presunu pri získavaní informácií, robenia vecí podľa seba, hravosť, ktorá často vedie k pochopeniu problému a podporuje tvorivosť. V tomto zmysle sa vzťahuje na zrod Moodla, ale aj na prístup študenta či učiteľa k výuke v online kurzoch. Ktokoľvek, kto používa Moodle je tzv. Moodler (nadšenec Moodle).

Moodle je vytvorený s cieľom pomôcť pedagógom vytvárať online kurzy s možnosťami bohatej interakcie. Jeho open-source licencia a modulárny dizajn dovoľujú vytvárať moduly, ktoré mu dodávajú ďalšiu funkcionalitu. Na vytvorenie nových modulov môže byť použitý jazyk PHP.

Moodle funguje na platformách Unix, GNU/Linux, FreeBSD, Windows, Mac OS X, NetWare a mnohých ďalších bez nutnosti zmeny v zdrojovom kóde, vďaka

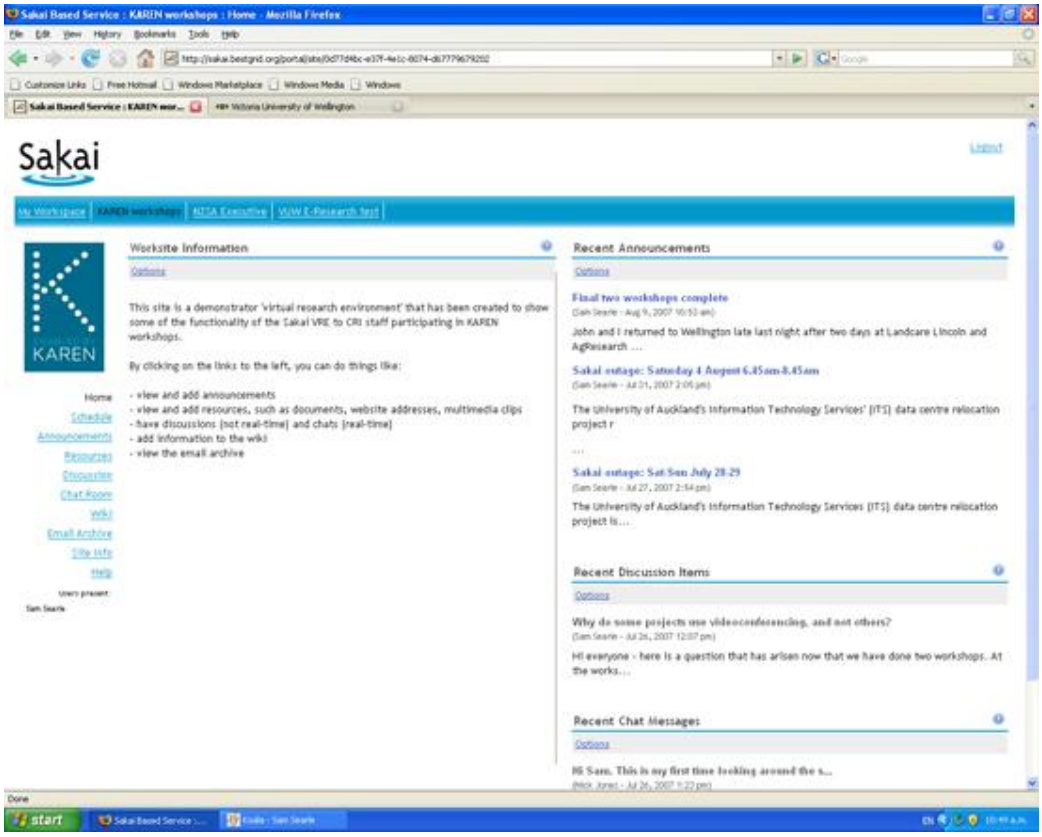
použitiu jazyka PHP. Údaje sú uložené v jednej databáze. V súčasnosti je Moodle vo verzii 1.9. [12], ktorý môžeme vidieť aj na **Obr. 2**.

Obr. 2 – Hlavná stránka systému Moodle

2.1.5.3 Sakai project

Sakai je komunita akademických inštitúcií, komerčných organizácií a jednotlivcov, ktorí pracujú spoločne na vytvorení spoločného Collaboration and Learning Environment (CLE). Jedná sa o voľne dostupnú softvérovú platformu distribuovanú pod licenciou typu Educational Community License. (Jedná sa o licenciu typu open-source.) Používa sa pre zjednodušenie výučby, výskumu a spolupráce.

Sakai (**Obr. 3**) je aplikácia implementovaná v jazyku Java, vytvorená tak, aby bola spoľahlivá a rozšíriteľná. Verzia 1.0 bola vydaná v roku 2005. V súčasnosti je k dispozícii verzia 2.5.3. Systém poskytuje mnoho služieb bežných v CMS systémoch, ako napríklad zdieľanie dokumentov, známkovanie, diskusiu, chat, online testovanie, wiki, mailing list, RSS čítačku, archivovanie správ a pod [13].



Obr. 3 – Sakai project

2.1.5.4 Angel Learning

ANGEL Learning, Inc. je súkromná spoločnosť zaoberajúca sa e-learningom. Jej hlavné produkty sú ANGEL Learning Management Suite (LMS) a ANGEL ePortfolio. ANGEL LMS je efektívny a výkonný systém, na manažment kurzov, vytváranie obsahu kurzov a pod. Používa sa na vytvorenie virtual learning environment-ov (VLE) pre online učenie.

ANGEL poskytuje inštruktorom (učiteľom) veľa nástrojov a flexibilitu na personalizáciu výučbového prostredia ich štýlu výučby, a ich cieľom. Jednoduchý, efektívny manažment kurzov spolu s mechanizmami na automatizáciu úloh zvyšujú efektivitu a eliminujú nutnosť vykonávať opakujúce sa úlohy. Príťažlivé a používateľsky prívetivé rozhranie aplikácie, inovatívne nástroje zvyšujú komfort používania [13][14].

Funkcie

- Organizácia obsahu
- Kontrola prístupu k obsahu
- Vytváranie Sylabov
- Hodnotenie študentov
- Manažment registrácii
- Efektívny manažment kurzov
- Vytváranie obsahu jednotlivých lekcií
- Interaktívne pracovné prostredie s podporou spolupráce
- Personalizácia komunikácie
- Podpora „podcastov“
- Wiki
- Blogy
- Email
- Online denník
- Chat a „instant messaging“
- Zdieľaná „kresliaca plocha“

2.1.5.5 Desire2Learn

Desire2Learn je vývojár e-learningovej rady produktov, ktorá obsahuje:

- § Learning Environment – webová kolekcia nástrojov na podporu výučby a učenia, ktorá poskytuje vývoj kurzov, ich manažment a pod.
- § Learning Repository – integrované „skladisko“, ktoré poskytuje ukladanie, tagovanie a umožňuje opakované použitie objektov určených pre výučbu.
- § LiveRoom – súbor komunikačných nástrojov slúžiaci na umožnenie komunikácie medzi študentmi a inštruktormi v reálnom čase.
- § ePortfolio – systém na zber, organizáciu, prezentovanie produktov, ktoré vznikli počas výučby. (Môže ísť o zadania, projekty a pod.) Môže byť použité aj na prezentáciu materiálov k úlohám, pri podpore práce na tímových projektoch. Pomáha pri vytváraní portfólia pri hľadaní práce.
- § Essentials – menší súbor webových nástrojov, ktoré poskytujú manažment, vytváranie kurzov a pod [15].

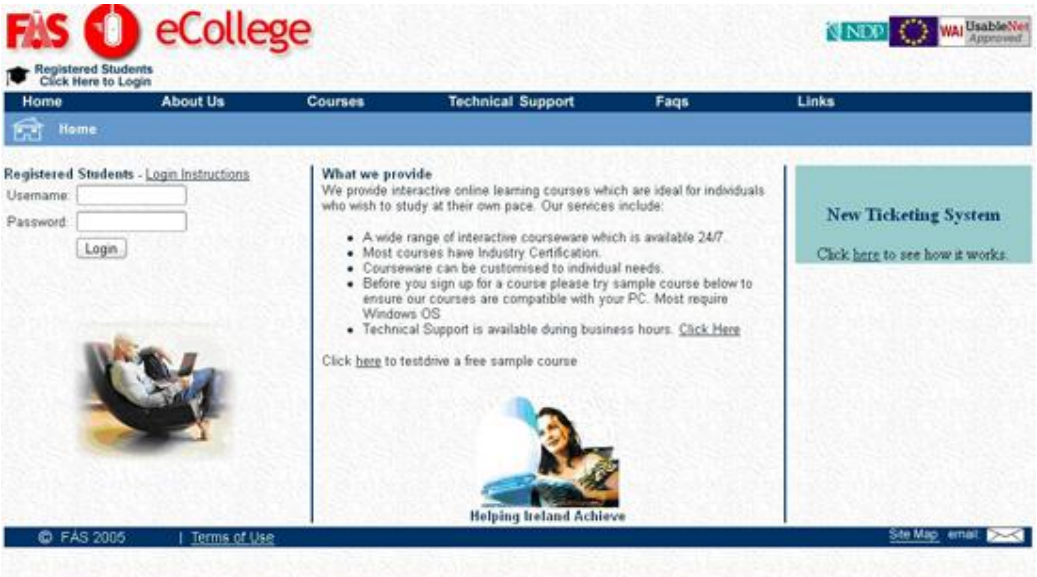
Obr. 4 – Rada produktov Desire2Learn

2.1.5.6 eCollege

eCollege je riešenie zaoberajúce sa e-learningom pozostávajúce z viacerých aplikácii a služieb, ktoré podporujú rôzne aspekty e-learningu. Hlavnou časťou, ako môžeme vidieť na obrázku **Obr. 5**, je Course Managment System (CMS). eCollege je však viac ako iba CMS, poskytuje mnoho ďalších funkcií prostredníctvom ďalších aplikácií balíka. Jedná sa o komerčné riešenie. Je spoľahlivé, flexibilné a prispôsobiteľné [16].

Ako sme spomínali, tak eCollege sa skladá z viacerých komponentov. Tu je ich zoznam:

- Course Management System (Systém manažmentu kurzu)
- Program Administration System (Systém na administráciu programu kurzu)
- Content Manager (Manažment obsahu)
- Enterprise Reporting (Slúži na vytváranie reportov)
- Learning Outcome Manager (Manažment výsledkov)
- Course Evaluation System (Systém na hodnotenie)
- ePortfolios (Vytváranie a prezeranie portfólií študentov)
- Synchronous eLearning (Video a audio konferencia, chat, zdieľaná pracovná plocha)



Obr. 5 - Systém eCollege

2.2 Iné riešenia

V tejto kapitole sme analyzovali iné riešenia s podobnou tematikou zadania. Dve práce sú zo záverečných prác (bakalársky a diplomový projekt) a dve práce sú z predmetu Tímový projekt.

2.2.1 Záverečné práce

2.2.1.1 Bakalársky projekt Ing. Jozefa Kytku

Ing. Jozef Kytka vytvoril vo svojej Bakalárskej práci Multimediálny výučbový modul pre Petriho siete. Rozhodli sme sa ho analyzovať, keďže v nami vytváranej aplikácii sa plánujeme venovať problematike Petriho sietí.

Výučbový modul je implementovaný ako modul do systému Moodle vo verzii 1.4.4. Použité sú jazyky PHP a Flash. Súčasťou výučbového modulu je voľne dostupný simulátor Petriho sietí PIPE, ktorý napomáha pochopeniu problematiky a v ktorom si môžu študenti prakticky overiť vedomosti.

Vo výučbovom module sa nachádza veľké množstvo textov zaoberajúcich sa Petriho sieťami, statické obrázky, neinteraktívne a interaktívne animácie vo Flash-i.

Kategórie používateľov

Používatelia systému sú rozdelení do dvoch základných skupín s rozdielnymi prístupovými právami. Prvým typom používateľov sú učitelia, ktorí sa starajú o údržbu systému a riadia celý proces výučby. Študenti majú možnosť využívať iba prostriedky, ktoré priamo súvisia s výučbou a možnosťami vzdelávania sa.

Funkcie

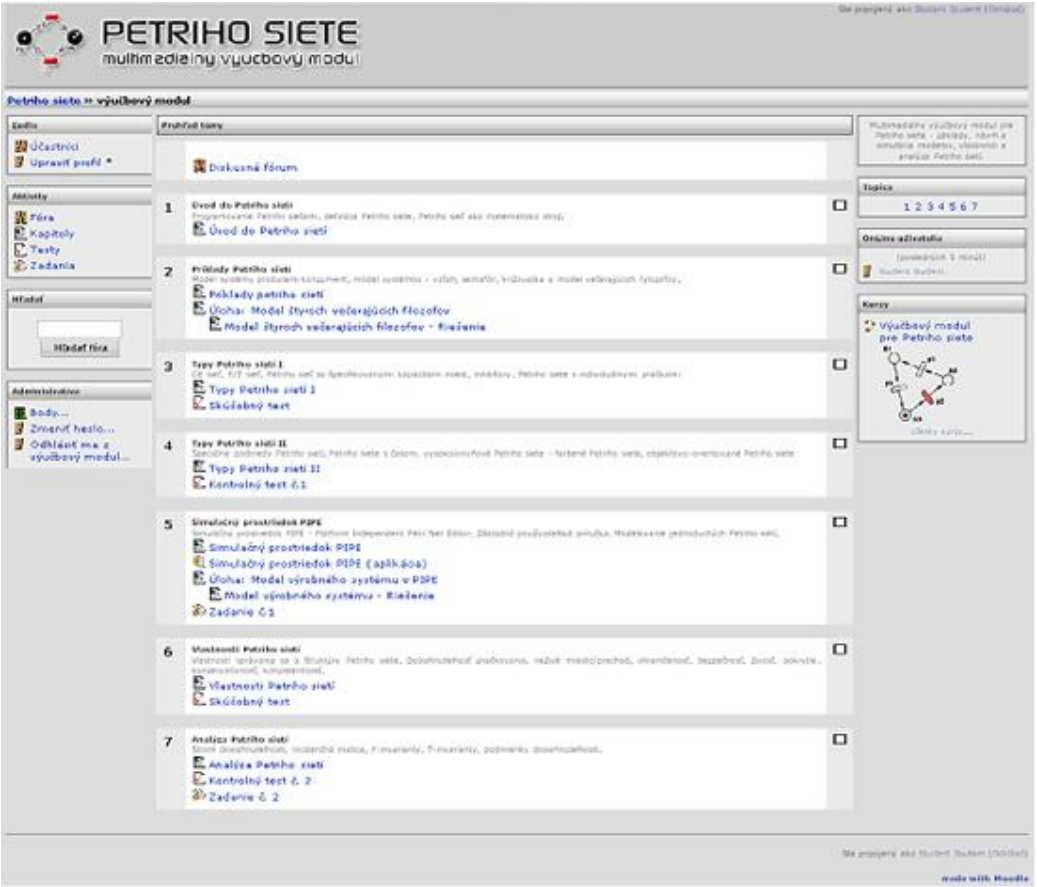
Medzi základné možnosti, ktoré ponúka výučbový modul patria: možnosť registrácie a prihlásenia sa do výučby, možnosť prehliadania jednotlivých tematických celkov kurzu Petriho siete, vypracovanie a hodnotenie skúšobných a hodnotených testov a možnosť odovzdávania súborov s vypracovanými zadaniami. Medzi rozšírené možnosti, ku ktorým majú prístup iba používatelia s prístupovými právami administrátora (patria medzi nich učitelia) sú: možnosť zadeľovania študentov do skupín, preberanie a hodnotenie súborov s odovzdávanými zadaniami od svojich študentov, možnosť upravovať vzhľad a obsah skúšobných, resp. hodnotených testov, možnosť úpravy tematických celkov a priebehu výučby a pod.

Dokumentácia

V dokumentácii je popísaný e-learning a jeho vzťah ku klasickému spôsobu výučby, zanalyzované existujúce riešenia simulátorov Petriho sietí, podrobne rozpísaný cieľ projektu a postup riešenia. Súčasťou dokumentácie je aj používateľská príručka, kde je popísaný spôsob inštalácie a návod na používanie. Rozsah dokumentácie je postačujúci vzhľadom na to, že sa jedná o bakalársky projekt.

Používateľské prostredie

Používateľské prostredie je jednoduché a prehľadné. Kurz je rozdelený na kapitoly, ktoré na seba nadväzujú. Nachádza sa v nich 9 krátkych animácií vo formáte Flash. 26 kontrolných otázok je rozdelených do štyroch testov, ktoré študenti vypracovávajú počas výučby. Na obrázku **Obr. 6** je zobrazená stránka obsahu kurzu „Petriho siete“ [18].



Obr. 6 - Multimediálny výučbový modul na Petriho siete

Zhodnotenie

Vytvorený výučbový modul je komplexný a poskytuje nástroje na výučbu problematiky Petriho sietí od úplných základov až po analýzu a vlastnosti Petriho sietí. V siedmich kapitolách študenti získajú vedomosti potrebné pre zvládnutie problematiky Petriho sietí v predmete ŠpOJ. Po každej kapitole musí študent vyvinúť aktivitu, či už pomocou nehodnoteného testu, alebo úlohy a pod. O kvalite tohto riešenia svedčí aj jeho reálne nasadenie na serveri http://spoj.fiit.stuba.sk/.

2.2.1.2 Diplomová práca Ing. Jozefa Kytku

Ing. Jozef Kytka vytvoril vo svojej diplomovej práci súbor štyroch modulov a to konkrétne modul VHDL Writer, modul VHDL Composer, Modul VHDL Analyser a modul Results. Spolu umožňujú študentom vypracovávať zadania z jazyku VHDL a tiež vyhodnocovať ich správnosť. To sa deje porovnaním reálnych výstupov s testovacími. Tiež umožňuje spravovať hodnotenie študentov.

Modul VHDL Writer

Toto je modul, v ktorom používatelia môžu vyvárať VHDL kód. Modul podporuje zvýrazňovanie syntaxe VHDL a taktiež sa v ňom nachádza VHDL parser, čiže automaticky dokáže vyhodnocovať správnosť písaného kódu a zároveň vizuálne označuje chyby. Modul podporuje ukladanie a otváranie rozpracovaných návrhov, to však pri takomto projekte považujeme za samozrejmosť.

Modul VHDL Composer

Modul VHDL Composer je veľmi podobný modulu VHDL Writer, avšak s tým rozdielom, že umožňuje komunikáciu s modulom VHDL Analyser. Čiže umožňuje odovzdať vypracovaný zdrojový kód modulu VHDL Analyser, ktorý ho následne spracuje. Po spracovaní modulom VHDL Analyser, čiže automatickom vyhodnotení, sa údaje zapíšu do databázy. Týmto je zadanie odovzdané a ohodnotené.

Modul VHDL Analyser

Ako bolo povedané pri module VHDL Composer, tento modul má za úlohu analyzovať a hodnotiť zdrojové kódy, ktoré sú odovzdané pomocou modulu VHDL Composer. Analyzuje sa syntax zdrojového kódu, z nej sa potom vytvorí štruktúra systému zadaného vo VHDL kóde. Potom sa vykonáva simulácia štruktúry tým spôsobom, že do štruktúry sú vpustené testovacie vstupy a výstupy sú porovnané s testovacími výstupmi. Výstupom z tohto modulu je hodnotenie zadania odovzdaného používateľom. Zadanie, teda zdrojový kód spolu s hodnotením sa následne zapíše do databázy.

Modul Results

Tento modul slúži používateľom na sprístupnenie informácií o hodnoteniach a stave odovzdaní jednotlivých zadaní. Tieto sa nachádzajú v kurze systému Moodle [19].

Zhodnotenie

Ing. Kytka vytvoril zaujímavú prácu, ktorej úroveň však degradujú niektoré nedostatky, a preto je jej nasadenie v reálnom prostredí diskutabilné. Oceňujeme zvýrazňovanie syntaxe v module VHDL Writer, ale na druhej strane tento modul nerozoznáva všetky rezervované slová. Modul VHDL Analyser umožňuje prácu s kombinačnými obvodmi, používa hradlá AND, OR, NOR, XOR, NAND. Nepoužíva

hradlo NOT. Neumožňuje prácu so sekvenčnými obvodmi, čo tiež obmedzuje použiteľnosť celého projektu. Systém vyhodnocovania, teda vytvorenia štruktúry až po základné logické členy napriek nedostatkom pôsobí zaujímavo. Najväčšie problémy vidíme v tom, že systém nepozná všetky príkazy jazyka VHDL a tiež nemožnosť práce s externými knižnicami. Keďže mnohí používatelia sú zvyknutí používať externé knižnice pri vytváraní svojich programov, systém na nich pôsobí obmedzujúco, lebo vyžaduje dodržiavať svoj štandard. Ďalší nedostatok vidíme v nemožnosti odsimulovať zadanie pred odovzdaním, čiže používatelia sú nútení použiť externý program, ak chcú vidieť, či je vytvorené zadanie správne. Myslíme si, že po odstránení spomínaných nedostatkov by mal tento projekt veľký potenciál. Keďže sme sa rozhodli vytvoriť výučbový modul zameraný na Petriho siete, nepredpokladáme využitie tohto projektu.

2.2.1.3 Diplomová práca Ing. Petra Šinkoviča

V rámci svojej diplomovej práce vytvoril Ing. Šinkovič štyri aplikácie, respektíve dve aplikácie a dve animácie. Jedná sa o: animáciu Simulácia vytvorenia Multiplexora v jazyku VHDL , animáciu Simulácia cyklu návrh, aplikáciu Simulátor šírenia hodnôt po signáloch v simulačnom cykle a aplikáciu Simulátor zotrvačného a prenosového oneskorenia priradenia signálu. Aplikácie majú interaktívny charakter a umožňujú používateľovi testovať rôzne varianty riešenia daného problému. Animácie sú väčšmi informatívne, bez možnosti zmeny údajov, obsahujú texty a majú za úlohu priblížiť danú problematiku používateľovi. Ďalšou časťou projektu je aplikácia pre overenie vedomostí z programovania.

Simulácia vytvorenia Multiplexora v jazyku VHDL

Ide o animáciu, ktorá zobrazuje postup vytvorenia entity multiplexora v jazyku VHDL. Entita pozostáva z niekoľkých komponentov. Aplikácia zobrazuje vytvorenie tejto entity po krokoch, pričom umožňuje používateľovi kedykoľvek sa vrátiť o krok späť resp. na začiatok. Animácia obsahuje texty, ktoré uľahčujú pochopenie procesu návrhu tejto entity v jazyku VHDL.

Simulácia cyklu návrhu

Táto animácia, ako už názov napovedá, zobrazuje cyklus návrhu. Používateľovi umožňuje si postupne prejsť po krokoch cyklus návrhu, ale umožňuje aj používateľovi zobraziť celý cyklus naraz a pozerať si jeho jednotlivé časti detailne. Nachádza sa tu tiež veľa textu, ktoré používateľovi popisujú cyklus návrhu obvodu.

Simulátor šírenia hodnôt po signáloch v simulačnom cykle

Táto aplikácia simuluje šírenie signálu cez obvod. Používateľ môže usporiadať obvod 243 spôsobmi a sledovať resp. simulovať šírenie signálu cez tieto obvody. Používateľ zadáva hodnoty dvoch vstupných signálov a ich oneskorenia. Aplikácia skontroluje správnosť vstupov a vytvorí výstup v podobe tabuľky, kde sú zaznamenané hodnoty signálov v jednotlivých krokoch simulačného cyklu. Aplikácia ponúka tiež funkciu pomocníka (help).

Simulátor zotrvačného a prenosového oneskorenia priradenia signálu

Táto aplikácia simuluje priradenie zotrvačných a prenosových oneskorení signálom v jazyku VHDL. Po zadaní vstupných hodnôt používateľov, vyhodnotí ich správnosť a následne ich spracuje. Ako výstup sa zobrazuje obsah ovládača signálu. Aplikácia tiež ponúka funkciu pomocníka (help).

Aplikácia pre praktické overenie vedomostí z programovania

Táto aplikácia slúži na automatické ohodnocovanie zdrojových kódov programov napísaných používateľom. Jej výhodou je to, že je aplikovateľná na ľubovoľný programovací jazyk, nielen na jazyk VHDL. Hodnotenie zdrojového kódu prebieha tak, že aplikácia hľadá kľúčové slová v zdrojovom kóde. Každé slovo má pridelené určité atribúty, ktoré ovplyvňujú pridelenie záporných alebo kladných bodov za použité slovo. Učiteľ má mnoho možností, ako upravovať hodnotenie a to napríklad zadefinovať počet bodov, ktoré budú strhnuté, ak je slovo mimo poradia, dané slovo zakázať, povoliť a pod. Aplikácia umožňuje, aby v nej bolo viacero testov. Používateľ má možnosť svoj zdrojový kód priebežne uložiť, načítať uložený zdrojový kód. Tiež poskytuje možnosť priebežného ohodnotenia zdrojového kódu. Predpokladané nasadenie aplikácie je navrhnuté na použitie najmä pri zápočtových písomkách. Pozostáva z dvoch častí prepojených pomocou databázy. Jednu časť potom využíva študent pri písaní zdrojového kódu a druhú učiteľ pri príprave testov a kontrole výsledkov študentov [20].

Zhodnotenie

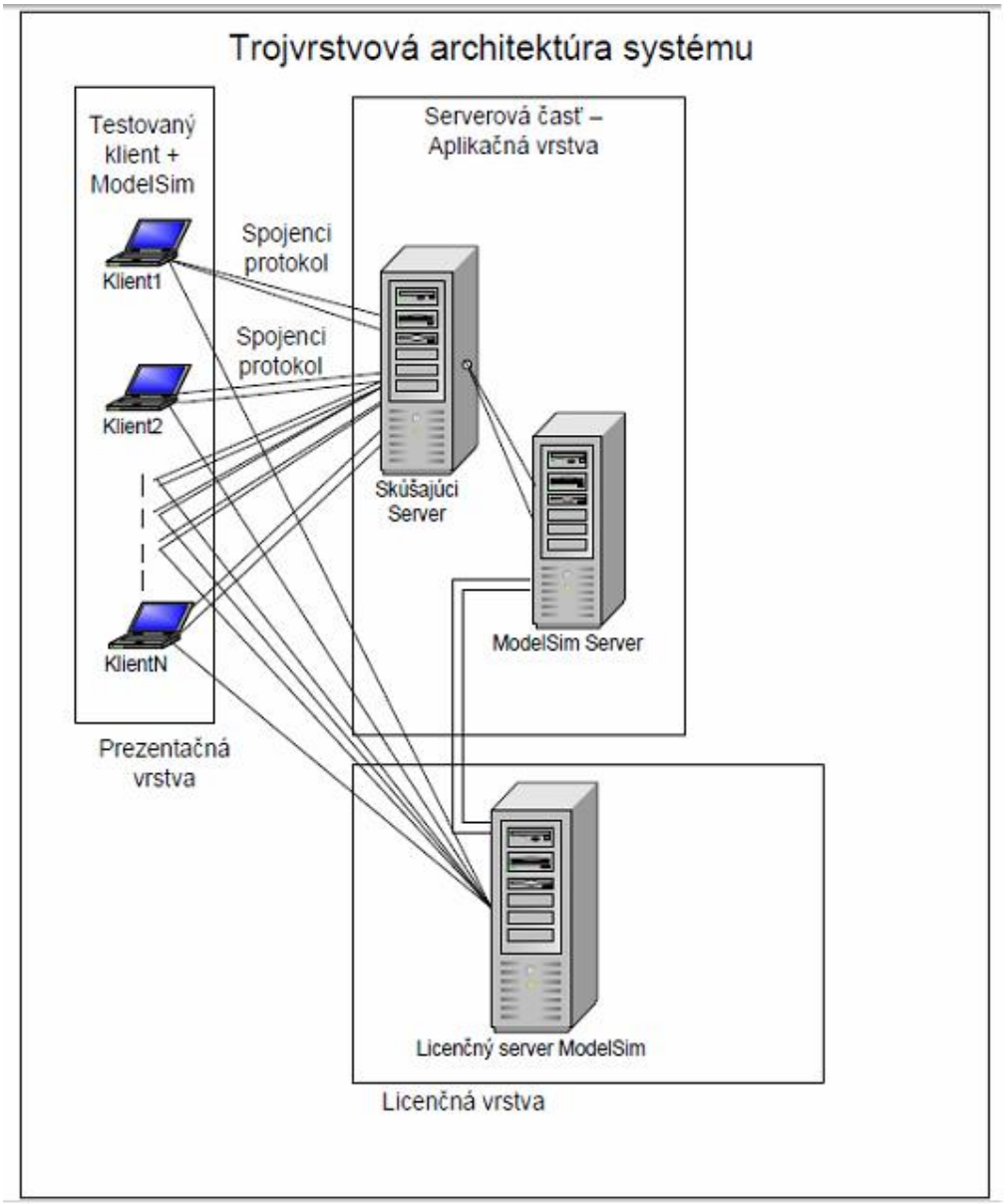
Ing. Šinkovič vytvoril zaujímavý súbor aplikácií, ktoré sú použiteľné na podporu výučby v predmete SPOJ. Jeho aplikácie poskytujú návod ako vytvoriť jednoduchú entitu v jazyku VHDL, tiež vysvetľuje postup cyklu návrhu. Umožňuje študentom pochopiť podstatu šírenia hodnôt signálu, ako aj simulovať oneskorenie signálu. Vysoko hodnotíme aplikáciu pre praktické overenie vedomostí z programovania, najmä pre jej univerzálnosť. Keďže náš projekt bude však riešiť problematiku Petriho sietí, nepredpokladáme ďalšie využitie tohto projektu.

2.2.2 Tímové projekty

2.2.2.1 Tím č.3 (2008) SPOJENCI

Podporou vzdelávania v predmete Špecifikačné a opisné jazyky sa zaoberal v rámci tímového projektu tím č. 3 SPOJENCI, konkrétne Bc. Štefan Beleš, Bc. Matej Jurikovič, Bc. Peter Pištek, Bc. Tomáš Polák, Bc. Jozef Zeman, Bc. Marián Žucha pod vedením Ing. Kataríny Jelemenskej, PhD. Ich tímový projekt sa skladal z dvoch častí a to aplikácie na testovanie zadaní v jazyku VHDL a modulu do systému Moodle. Keďže systém Moodle je predmetom inej časti analýzy, budeme sa tu venovať iba analyzovaniu aplikácie na testovanie zadaní.

Ide o aplikáciu je typu klient-server. Bola implementovaná v jazyku C++. Architektúra implementovaného systému je znázornená na nasledujúcom obrázku **Obr. 7**.



Obr. 7 - Architektúra aplikácie vytvorenej tímom SPOJENCI

Študent - klientská časť

Keď študent príde na test, musí sa prihlásiť do aplikácie prideleným menom a heslom. Po spustení testovania mu príde od servera vygenerované zadanie a začne sa odpočítavať pridelený čas. V pravidelných intervaloch je zdrojový kód prejdený programom a bude zvýraznená syntax, čo slúži na lepšiu orientáciu v zdrojovom kóde. Študent má možnosť kontroly zdrojového kódu. Na klientskej časti tiež prebieha spracovanie kompilácie, simulácia aj komparácia. Klientská časť toto realizuje pomocou skriptov, pomocou ktorých ovláda program ModelSim. Zadanie odovzdáva študent alebo v prípade naplnenia časového limitu, je napísaný zdrojový kód automaticky odovzdaný

klientskym programom. Po spracovaní serverom bude študentovi zobrazené priebežné hodnotenie odovzdaného zadania.

Funkcie klienta:

- Prihlásenie študenta
- Príjem zadania
- Lokálna kompilácia a simulácia
- Automatické odovzdanie po vypršaní časového limitu
- Zobrazenie priebežného hodnotenia
- Učiteľ - serverovská časť

Serverová časť aplikácie je rozdelená na 3 časti:

Tvorba a manažment zadaní: Pri vkladaní zadania musia byť vytvorené 4 komponenty: Text zadania, Vzorová entita, Testbench, Referenčné riešenie – formát wlf, Obrázok so schémou. Správa zadaní obsahuje ich úpravu a prípadné mazanie.

Spúšťanie testov a monitoring klientov: Pred začatím testovania sa nastaví časový limit a overí sa prihlásenie klientov. Všetkým klientom sa naraz pošlú zadania a majú jednotný čas riešenia zadania. V prípade výskytu akýchkoľvek problémov môže učiteľ tento čas jednoducho predĺžiť. Počas testovania môže systém v prípade podozrenia z nekalých aktivít zobraziť klientskú obrazovku, prípadne poslať varovanie, či úplne zrušiť test inkriminovanému študentovi.

Hodnotenie zadaní: Server udržuje zoznam odovzdaných zadaní, ktorý poskytuje prezeranie odovzdaného zdrojového kódu, výstupnej vzorky priebehov a porovnanie vzorky študenta s referenčnou vzorkou. Na základe týchto informácií môže učiteľ poopraviť alebo potvrdiť priebežné hodnotenie systému.

Funkcie servera:

- Import zoznamu študentov zo súboru
- Nastavenie zadania testu – skladá sa zo 4 súborov: Text zadania, Vzorová entita, Testbench, Referenčné riešenie – formát wlf, Obrázok so schémou
- Nastavenie adresára, kde sa majú ukladať výsledky testu
- Nastavenie času trvania testu

- Test sa spustí len pre študentov, ktorí sú prihlásení na serveri. Je možné individuálne spustiť test, napr. v prípade technických problémov alebo neskorého príchodu.
- Nastavenie bonusového času
- Kicknutie študenta – okamžité zrušenie testu
- Ukončenie testu – umožní študentovi poslať aktuálnu verziu testu alebo ponechať ako odovzdané riešenie súbor naposledy odovzdaný na server.
- Učiteľ si môže prezerať: zdrojový kód v editore WordPad a ModelSime a vizuálne porovnanie odovzdaného a referenčného riešenia v ModelSime
- Úprava hodnotenia vygenerovaného systémom [21]

Zhodnotenie

Tím č.3 SPOJENCI vytvoril plne funkčnú aplikáciu na testovanie študentov v predmete Špecifikačné a opisné jazyky. Aplikácia je zameraná na testovanie znalostí z jazyka VHDL. Aplikácia je do istej miery univerzálna, keďže by bez zmeny v kóde aplikácie mohla byť použitá pre testovanie vedomostí študentov z jazyka SystemC. V aplikácii je použitý tzv. Hrubý klient, t.j. kompilácia zdrojového kódu sa realizuje lokálne a tým sa značne znižujú nároky na stranu servera. Nevýhoda tohto riešenia je v tom, že klienti musia mať prístup k licenčnému serveru ModelSim. Na strane klienta sú kvôli bezpečnosti zachytávané a oznamované klávesy a klávesové skratky, ktoré by mohli zmariť vykonávanie testu. Komunikácia medzi klientom a serverom prebieha cez komunikačný protokol vytvorený tímom. Toto zabezpečuje istú mieru bezpečnosti, keďže použitý protokol podporuje zotavovanie sa z chýb. Používateľské rozhranie je jednoduché a prehľadné.

2.2.2.2 Tím č. 4 (2008) @Tím

@Tím vytváral v akademickom roku pod vedením Ing. Tomalovej Tímový projekt s názvom Podpora vzdelávania v predmete Špecifikačné a opisné jazyky. Zloženie tímu bolo: Bc. Marián Chovanec, Bc. Jaroslav Brutenič, Bc. Tomáš Pulai, Bc. Vladimír Gregor, Bc. Michal Bilík.

Tím sa v svojej práci zaoberá návrhom e-learningového modulu pre testovanie študentov v predmete Špecifikačné a opisné jazyky. Ako súčasť praktickej výučby, navrhovaný a implementovaný systém by mal študentom pomôcť v lepšom pochopení problematiky a zjednodušiť proces testovania pre vyučujúceho.

Implementácia

Tím implementoval projekt v jazyku PHP na strane servera. Okrem toho boli použité aj technológie AJAX a JavaScript. Technológia AJAX umožňuje vytvárať interaktívne stránky bez nutnosti ich znovu-načítania, takže požiadavky smerom od klienta na server sú posielané, aj keď je už načítaná celá stránka. Toto umožnilo tímu ukladať priebežné stavy riešenia počas práce. Na implementáciu používateľského rozhrania bola použitá trieda z technológie vyvíjanej spoločnosťou YAHOO – UI framework. Ide o framework napísaný v jazyku JavaScript. Táto technológia umožňovala riešiť previerku metódou drag & drop, od toho je aj odvodený názov riešenia „Chyťaposunovka“.

Používateľské rozhranie, funkcie

Systém slúži na testovanie praktických znalostí študentov z jazyka VHDL. Je to realizované tak, že učiteľ vytvorí zadanie a zadá test. Študenti sa potom prihlásia do systému Moodle a začnú test. Študenti majú v hlavnom okne zdrojový kód v jazyku VHDL, aj so zvýraznením syntaxe. V tomto zdrojovom kóde sú vyznačené miesta, kde chýbajú riadky kódu. Na pravej strane majú ponuku možností (riadky kódu vo VHDL) na doplnenie do zdrojového textu v hlavnom okne. Systémom drag & drop umiestňujú možnosti z ponuky do zdrojového kódu. Systém po dokončení previerky automaticky overí správnosť riešenia a vyhodnotí študenta. Používateľské prostredie je veľmi prehľadné a intuitívne, aj vďaka použitiu metódy drag & drop.

Dokumentácia

Dokumentácia projektu je na vysokej úrovni. Veľmi kladne hodnotíme kapitolu *Podrobný návrh systému*, kde je detailný opis implementovaných tried, funkcií a modulov. Naprogramované funkcie sú veľmi dôkladne v tejto časti zdokumentované, čo by nám veľmi uľahčilo prípadné rozširovanie projektu v budúcnosti [22].

Zhodnotenie

@Tím vytvoril zaujímavý a pôsobivý modul do systému Moodle, ktorý umožňuje testovanie vedomostí z jazyka VHDL. Vysoko hodnotíme použitie jednoduchého používateľského rozhrania a metódy drag & drop. Dokumentácia k projektu je postačujúca, podrobne je rozpísaný podrobný návrh systému. V používateľskej príručke je aj podrobný návod na inštaláciu systému. Ako najväčšie nedostatky vidíme zvýrazňovanie syntaxe iba pre jazyk VHDL a tiež situáciu, keď je študent neprítomný na teste – dostane automaticky 0b.

2.3 Petriho siete a E-Learning

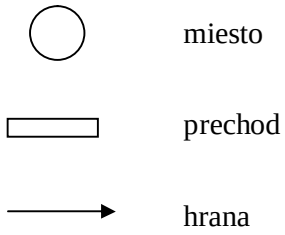
V tejto kapitole si uvedieme definíciu Petriho sietí, vlastnosti a základné funkcie a rozanalyzujeme si aplikácie, ktoré pracujú s Petriho sieťami.

2.3.1 Petriho siete

Systémy diskrétnych udalostí sa dajú popísať pomocou stavových automatov. Tie sú dané množinou stavov, prechodovou funkciou a jedným aktuálnym stavom. V prípade distribuovaných systémov, kde je potreba popisovať aktivity jednotlivých podsystémov a ich vzájomné vzťahy, môže byť model stavovými automatmi ťažkopádny. Je to preto, že každá možná kombinácia stavov jednotlivých podsystémov potrebuje vlastný stav vo výslednom stavovom automate. Na jeho úrovni sa potom odohráva analýza jeho vlastností. Zmienený nedostatok čiastočne odstraňuje iný nástroj pre modelovanie a analýzu systémov diskrétnych udalostí, nazvaný *Petriho sieť*. Analýzu vlastností Petriho siete je možné v niektorých prípadoch previesť bez vyčíslení grafu dosiahnuteľných značkovaní, ktorá odpovedá výslednému stavovému automatu [22].

Petriho siete vznikli rozšírením modelovacích možností konečných automatov. Koncepčným jadrom popisu diskrétneho systému Petriho sieťou je znázornenie distribúcie stavu systému na parciálne stavy, vyjadrené podmienkami a spojenie týchto podmienok s udalosťami systému. V grafickej reprezentácií Petriho sietí sa parciálne stavy systému zobrazujú pomocou kružníc a nazývajú sa *miesta Petriho siete*. Udalosti, graficky zobrazované ako obdĺžniky alebo úsečky sa nazývajú *prechody Petriho siete*. Spojenie

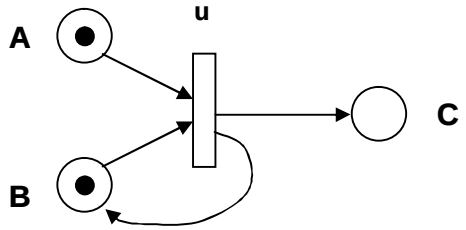
prechodu a miesta sa nazýva *hrana Petriho siete*. Konkrétne grafické znázornenie je uvedené na **Obr. 8**.



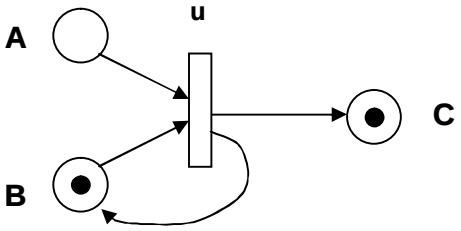
Obr. 8 - Grafická reprezentácia Petriho siete

Petriho sieť možno chápať ako automat, ktorý definuje správanie systému postupnosťami udalostí a odpovedajúcimi stavovými zmenami systému. V prípade Petriho sietí je okamžitý stav modelovaného systému určený určitými parciálnymi stavmi spojenými s príslušnými miestami siete. Dosiahnutie určitého parciálneho stavu v systéme je znázornené označkovaním miest Petriho siete. Označkovanie alebo značkovanie miesta je nezáporná celočíselná informácia, ktorá je graficky vykreslená určitým počtom čiernych bodiek nazývanými *značky*.

Aby mohla prebehnúť udalosť modelovaná určitým prechodom, je nutná podmienka existencie značky vo všetkých vstupných miestach prechodu. Ak daná udalosť prebehne, odoberie značky zo vstupných miest. Tým prestanú platiť vstupné podmienky a vzniknú značky vo výstupných miestach (začnú platiť podmienky spôsobené touto udalosťou). Príklad, ktorý ukazuje spustenie prechodu so vstupným značkovaním je znázornený na obrázkoch **Obr. 9**, resp. **Obr. 10**. Udalosť *u* nastane, ak sú splnené podmienky *A, B, C*.

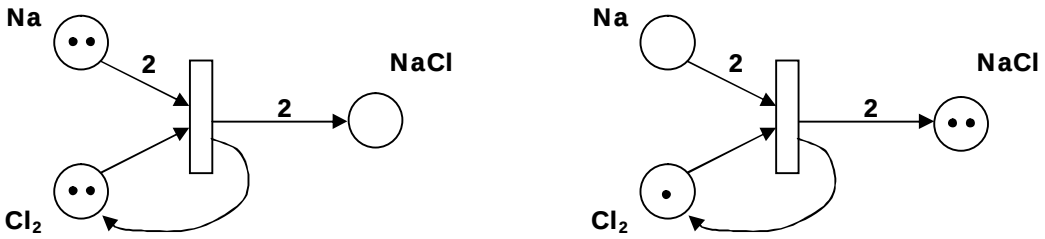


Obr. 9 - Značenie miest, kedy môže udalosť nastať



Obr. 10 - Značenie miest, po prebehnutí udalosti

Značkovanie miesta môže byť neobmedzené alebo môže byť obmedzené *kapacitou miesta* – celým kladným číslom udávajúcim maximálny možný počet značiek v danom mieste. Počet značiek potrebných pre spustenie udalosti (prechodu) je znázornený ohodnotením hrany. Ohodnotenie hrany určuje počet značiek, ktoré sa odoberajú alebo pridávajú do príslušných miest [24].



Obr. 11 - Vznik NaCl pomocou chlóru a sodíka reprezentované Petriho sieťou

2.3.2 Vlastnosti Petriho sietí

2.3.2.1 Dosiahnuteľnosť (Reachability) značkovania

Značkovanie M_r je dosiahnuteľné vtedy, ak existuje spúšťacia postupnosť prechodov σ začínajúca v M_0, pre ktorú platí:

$$s = t_{i_1} t_{i_2} \ldots t_{i_3}, \qquad t_{i_k} \in T$$

Potom označkovanie M_r dostávame vzťah:

$$M_r = M_o + \sum_{s=1}^{r} \Delta t_{i_s}$$

V tomto prípade je značkovanie M_r dosiahnuteľné zo značenia M_0, zapisujeme $M_0[s\rangle M_n$. Zápis zároveň vyjadruje, že postupnosť s možno zrealizovať zo značenia M_0. Množina všetkých dosiahnuteľných označkovaní Petriho siete je daná vzťahom:

$$R_{PN}(M_0) = \{M_k \mid M_k \quad je \quad dosiahnuteľné\}$$

Množina $[M\rangle$ sa nazýva *množina dosiahnuteľných značkovaní* (reachability set) *zo značenia M*; množina $[M_0\rangle$ dosiahnuteľných značkovaní z počiatočného značenia sa nazýva *množina dosiahnuteľných značkovaní siete N*.

2.3.2.2 Ohraničenosť a bezpečnosť (Boundedness & Safeness)

Nech $N = (P, T, F, W, K, M_0)$ je Petriho sieť. Miesto $p \hat{I} P$ sa nazýva k – ohraničené, ak platí $\exists k \in N : \forall M \in [M_0\rangle : M(p) \leq k$.

Miesto *p* sa nazýva ohraničené, ak je k – ohraničené pre určité $k \in N$. Ak je každé miesto Petriho siete *N* ohraničené, potom *N* sa nazýva *ohraničenou Petriho sieťou.*

2.3.2.3 Živosť (Liveness)

Nech $N = (P, T, F, W, K, M_0)$ je Petriho sieť. Dosiahnuteľné značkovanie $M_d \in R_{PN}(M_0)$ *je mŕtve*, ak neexistuje prechod spustiteľný z M_d. Ak je Petriho sieť korektným modelom reálneho systému a systém prejde do stavu reprezentovaného označkovaním M_d, v ktorom nemôže ďalej pokračovať vo svojej činnosti, vtedy hovoríme, že systém je v *mŕtvom stave*.

Nasledujúce postupnosti udalostí, ktoré si uvedieme, dostanú systém do mŕtveho stavu.

1) Prechod $t \in T$ je *živý na hladine 0*, ak nie je spustiteľný pri žiadnom značení z $[M_0\rangle$. Hovoríme, že prechod je neživý.
2) Prechod $t \in T$ je *živý na hladine 1*, ak existuje $M \in [M_0\rangle$ také, že prechod t je M - spustiteľný.
3) Prechod $t \in T$ je *živý na hladine 2*, ak existuje postupnosť spúšťania prechodov σ Petriho siete N, v ktorej sa prechod t vyskytuje aspoň n - krát, kde n je kladné prirodzené číslo.
4) Prechod $t \in T$ je *živý na hladine 3*, ak existuje postupnosť spúšťania prechodov σ Petriho siete N, v ktorej sa prechod t vyskytuje nekonečne veľakrát.
5) Prechod $t \in T$ je *živý na hladine 4*, ak pre každé $M \in [M_0\rangle$ existuje značenie M' také, že $M' \in [M\rangle$ a t je M'- spustiteľný.

Petriho sieť N je *živá na hladine h*, $h \in \{0,1,2,3,4\}$, ak každý prechod $t \in T$ je živý na hladine *h*.

Nech $N = (P, T, F, W, K, M_0)$ je Petriho sieť. Značenie $M \in [M_0\rangle$ je *živé,* ak pre $\forall t \in T$ existuje značenie $M' \in [M\rangle$ také, že prechod t je M'- spustiteľný. Ak sú všetky značenia $M \in [M_0\rangle$ živé, je aj Petriho sieť *N živá* (na hladine 4).

2.3.2.4 Pokryteľnosť (Coverability)

Nech $N = (P, T, F, W, K, M_0)$ je Petriho sieť. Značkovanie $M \in [M_0\rangle$ je pokryté práve vtedy, ak platí, že $M' \in [M_0\rangle$ také, že pre každé miesto $p \hat{I} P$ platí $M'(p) \geq M(p)$. Ak značkovanie nie je pokryté, Petriho sieť je v danom značkovaní L0 živá.

2.3.2.5 Konzervatívnosť (Conservative)

Konzervatívnosť je vlastnosťou štruktúry Petriho siete, ktorá skúma zachovávanie počtu značiek v Petriho sieti. Petriho sieť $N = (P, T, F, W, K, M_0)$ nazývame konzervatívnou vzhľadom k váhovému vektoru *V: P ® N*, ak platí:

$$\forall M \in [M_0\rangle : \sum_{p \in P} V(p).M(p) = \sum_{p \in P} V(p)M_0(p)$$

2.3.2.6 Konzistentnosť (Consistency)

Petriho sieť môžeme nazvať konzistentnou, ak sa po spustení všetkých prechodov (pričom nezáleží na poradí spúšťaní prechodov) sieť dostane opäť do počiatočného značkovania.

2.3.2.7 Reverzibilita (Reverzibility)

Ak pre každé dosiahnuteľné značkovanie $M \in [M_0\rangle$, platí, že počiatočné označkovanie M_0 patrí do množiny dosiahnuteľných označkovaní z *M,* teda $M_0 \in [M\rangle$,

môžeme povedať, že Petriho sieť je reverzibilná. Táto definícia sa dá zapísať aj matematicky:

$$M \in R_{PN}(M_0) \quad \wedge \quad M_0 \in R_{PN}(M)$$

Popísané tri vlastnosti Petriho siete (ohraničenosť, živosť a reverzibilita) sú navzájom nezávislé [22].

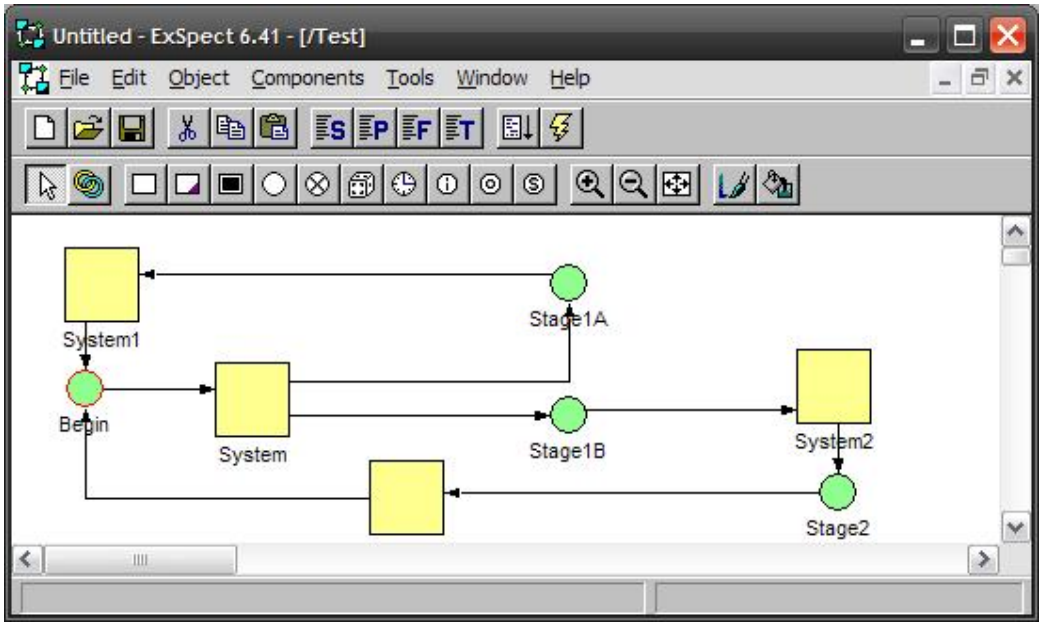
2.3.3 Nástroje pre Petriho siete

Predtým, ako začneme pracovať na vlastnom riešení sa oboznámime s niekoľkými existujúcimi riešeniami pre modelovanie a analýzu Petriho sietí a príbuzných systémov. Umožní nám to lepšie navrhnúť nový produkt, implementovať kľúčovú funkcionalitu a vyhnúť sa mnohým chybám.

2.3.3.1 ExSpectT

ExSpectT je softvérový nástroj na modelovanie diskrétnych systémov, predovšetkým na modelovanie procesov v obchode a výrobe, modelovanie prípadov použitia atď. Pôvodne vznikol ako výsledok akademického projektu na TU Eindhoven, v súčasnosti je ponúkaný ako komerčný produkt [25].

ExSpecT model opisuje procesy ako množinu úloh, ktoré navzájom komunikujú správami. Sieť úloh sa modeluje graficky. Jednotlivé úlohy môžeme ďalej dekomponovať a vytvoriť tak hierarchickú sieť. Podporované sú aj farebné Petriho siete. K dispozícii sú analytické nástroje: overenie štrukturálnej správnosti, simulácia s možnosťou krokovania. Okrem bežných súborových operácií dokáže importovať modely z príbuzných systémov.



Obr. 12 - Používateľské rozhranie je typizované, hrany sú vedené ako lomené čiary v pravom uhle

Aplikácia má typizované používateľské rozhranie s variabilnou veľkosťou pracovnej plochy. Pri vytváraní siete ani simulácii sme neodhalili žiadne závažné nedostatky. Pozitívne hodnotíme „pravouhlé“ vedenie hrán a možnosť upravovať veľkosť a tvar grafických objektov. Za nekomfortný považujeme spôsob vytvárania siete, pred každým ďalším objektom treba kliknúť na tlačidlo pre nové miesto, prechod alebo novú hranu. Zostavená sieť sa nedá okamžite simulovať, je potrebné nakonfigurovať správanie sa prechodov. Chýbajú nám typické analytické nástroje: dosiahnuteľnosť, živosť alebo bezpečnosť.

Na aplikácii oceňujeme uhladené používateľské rozhranie a dobre spracovanú grafickú reprezentáciu siete. ExSpecT je dobrý modelovací nástroj, naše predstavy o jednoducho použiteľnej aplikácii špecializovanej na Petriho siete však nespĺňa.

2.3.3.2 Maria

Maria alebo Modular Reachability Analyzer [26] je analyzátor dosiahnuteľnosti pre Petriho siete a príbuzné modely. Umožňuje predovšetkým:

- Vizualizovať a interaktívne simulovať model.
- Generovať graf dosiahnuteľnosti.

- Analyzovať živosť – identifikácia deadlock.
- Analyzovať bezpečnosť.

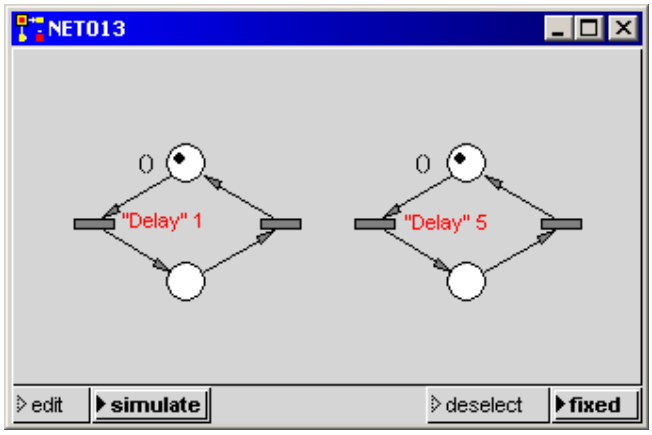
Používateľské rozhranie je textové (konzola) alebo grafické, prostredníctvom nástroja na vizualizáciu grafov Graphviz - Graph Visualization Software. Vstupom pre program je formálne opísaný model siete v textovej forme. Aplikácia nevyniká jednoduchosťou použitia, skôr naopak. Jej význam spočíva v tom, že ako jedna z mála ponúka potrebné analytické funkcie a je k nej dostupný kompletný zdrojový kód napísaný v jazyku C++.

2.3.3.3 PAGE

PACE [27] je integrované vývojové prostredie od firmy IBE, na modelovanie, simuláciu a analýzu komerčných, logistických a technických procesov. Jedná sa o komerčný produkt, ktorý sa nám nepodarilo otestovať. Umožňuje modelovanie a simuláciu udalostných systémov pomocou Petriho sietí a fuzzy modelov. Ponúka rozličné metódy optimalizácie, ktoré sa dajú navzájom kombinovať. Výsledkom je určenie optimálnych parametrov modelu a tým optimálne nastavenie reálnych systémov a biznis procesov. Výrobca uvádza tieto charakteristiky:

- Petriho siete s atribútmi umožňujú modelovanie paralelných procesov.
- Možnosť vytvárať hierarchické siete.
- Obmedzenia kapacity miest.
- Objektovo orientovaný jazyk Smalltalk pre efektívnu implementáciu komplexných operácií.
- Spolupráca s kancelárskymi programami.
- Volanie externých procedúr, graficky vstup a výstup
- Použitie vlastných ikon pre miesta a prechody, ktoré prispievajú k lepšej čitateľnosti modelu.
- Predvolené štatistické modely.
- Fuzzy logika je ideálna na opis reálnych systémov s paralelnými procesmi.
- Automatická a grafická optimalizácia procesov.
- Modelovanie času.
- Modelovanie náhodného správania.

Na základe príručky k softvéru vieme povedať, že používateľské rozhranie sa skladá z viacerých nezávislých okien, v ktorých môžeme na model nahliadať z viacerých uhlov pohľadu. Pri vytváraní Petriho siete v príslušnom okne používame ponuku na spodnom okraji alebo kontextové menu. Pre pridanie nového miesta alebo prechodu zvolíme príslušnú položku z menu a klikneme na miesto, kam chceme objekt umiestniť. Pre každý ďalší musíme zvoliť príslušnú položku menu, čo môže byť pri rozsiahlych sieťach zdĺhavé a unavujúce. Hranu vytvárame uchopením objektu, kde táto začína a ťahaním na objekt, kde hrana končí. Nepodarilo sa nám zistiť, aké konkrétne analytické funkcie sú podporované. Simulácia ponúka nadštandardné možnosti, spúšťa sa cez menu „Simulate“.



Obr. 13 - Okno pre modelovanie a simuláciu siete je len jedným z mnohých.

Už na prvý pohľad je jasné, že PACE je viac ako iba editor a simulátor Petriho sietí. Z tu ponúkanej funkcionality si však uplatnenie v našom riešení nájde iba malá časť.

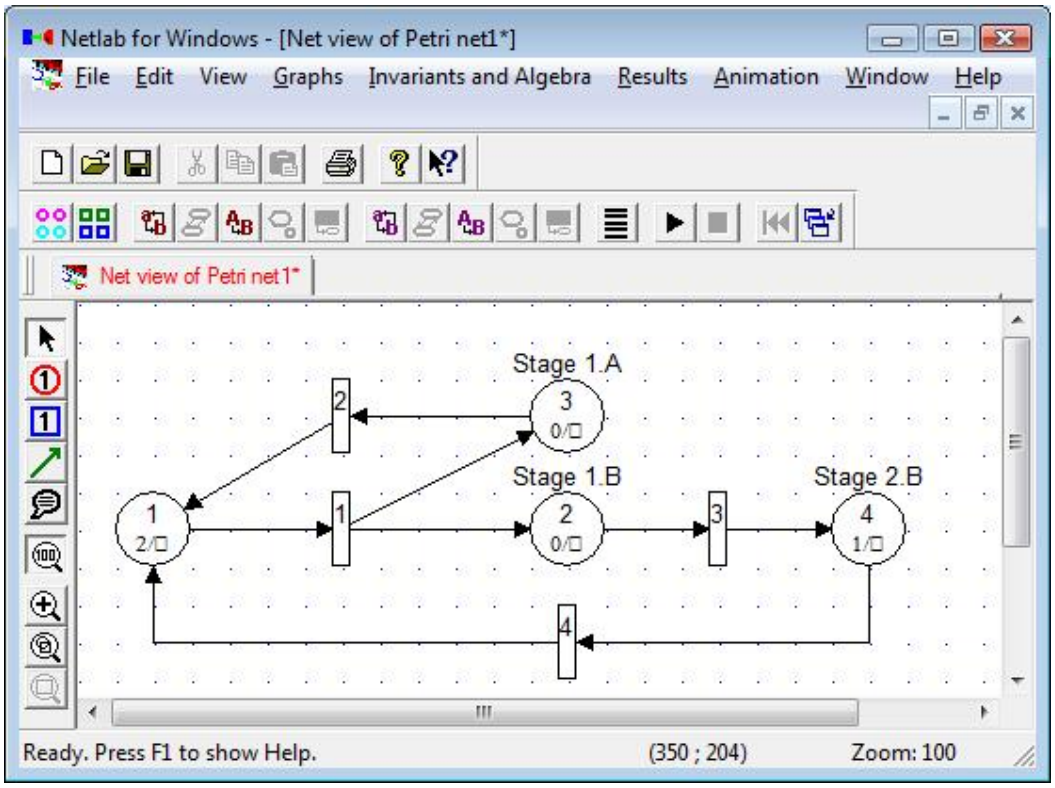
2.3.3.4 Netlab

Netlab [28] je softvérový nástroj pre modelovanie a analýzu P/T Petriho sietí. Spolu s nasledujúcim programov, najlepšie spĺňa naše predstavy o používateľsky príjemnej, jednoduchej a funkčne vyspelej aplikácii. Potvrdzuje to aj ponúkaná funkcionalita:

- Grafický editor Petriho sietí.
- Analytické nástroje s možnosťou súhrnného vyhodnotenia:
  - Dosiahnuteľnosť
  - Pokrytie

- § Živosť, čiastočný a úplný deadlock
- § Mŕtve prechody
- § Konzistentnosť
- § Ohraničenosť
- § Konflikty
- § Kontakty

- Kroková interaktívna simulácia – prechody sú manuálne spúšťané používateľom.
- Spolupráca s prostredím Matlab.



Obr. 14 - Typizované rozhranie, výborné modelovacie schopnosti a široká paleta funkcií sú spoločné znaky.

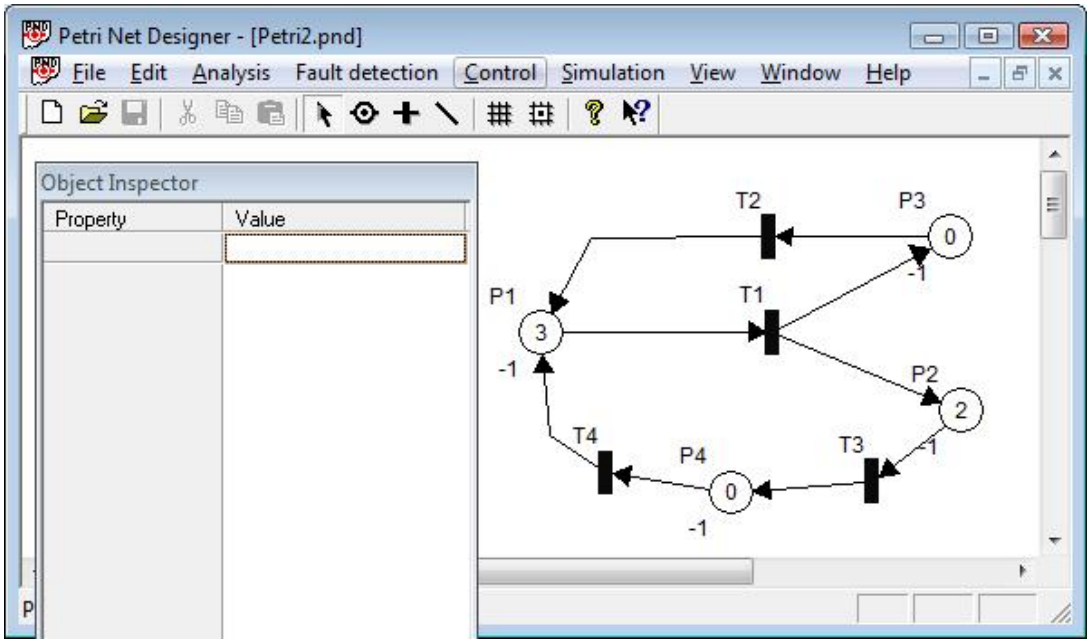
Grafické rozhranie je familiárne, používateľ sa stretáva s už známymi prvkami. Podporovaná je práca s viacerými oknami. Myš pracuje v jednom z štyroch režimov: presúvanie a úpravy objektov – štandardný režim, pridávanie miest, pridávanie prechodov, pridávanie hrán. Režimy sú výhodné najmä pri vytváraní väčších sietí, kedy urýchľujú prácu. Jednotlivé objekty sa dajú ďalej prispôsobovať ako po grafickej stránke (veľkosť, tvar, menovka), tak aj po funkčnej (nastavenie kapacity, aktuálneho

značkovania, a iné). Hrany sú riešené ako lomené čiary, čo prispieva k sprehľadneniu siete. Vytvárané sú kliknutím na zdrojový a následne cieľový objekt. Analytické nástroje sú dostupné takpovediac na jedno kliknutie, ich výsledky si môžeme uložiť priamo z aplikácie. Veľkosť modelov nie je obmedzená, autori však hovoria o spomalení behu programu pri viac ako 100 uzloch, respektíve 10000 uzloch pri analýze dosiahnuteľnosti.

Pozitívne hodnotíme použitie typizovaného používateľského rozhrania, možnosť prispôsobenia objektov po funkčnej aj grafickej stránke, režimy myši a širokú paletu analytických funkcií. Rezervy vidíme v simulácii, kde nám chýba plne automatická simulácia.

2.3.3.5 PetriNet Designer

Aplikácia vznikla ako výsledok študentského projektu na FIIT STU v Bratislave. Ako sme už uviedli, patrí medzi dve riešenia, ktoré nám najlepšie vyhovujú.



Obr. 15 - Aplikácia má mnoho spoločného s programom Netlab ako po vzhľadovej, tak aj po funkčnej stránke.

Vo všeobecnosti sa aplikácie takmer zhodujú, či už po vzhľadovej stránke, ovládaním, ako aj ponúkanou funkcionalitou. Rovnako používajú štyri režimy myši na vytváranie siete. Drobný rozdiel je v postupe vytvárania hrany. Petri Net Designer

neumožňuje meniť vzhľad prechodov a miest, úpravy vedenia hrany sú však zvládnuté lepšie. Hrana obsahuje uzly, ktoré sa dajú presúvať, a tak sa mení jej vedenie. Zaujímavé je použitie karty „Object Inspector“ na zmenu parametrov objektu namiesto dialógového okna. Toto riešenie je jednoduchšie na použitie a podstatne rýchlejšie. Dostupné sú tieto analytické nástroje:

- Matica incidencie a počiatočné značkovanie
- P- invariat, T-invariant
- Dosiahnuteľnosť
- Ohraničenosť
- Pokrytie
- Bezpečnosť
- Živosť a deadlock
- Konzistentnosť

Všetky sú funkčné, nie sú však tak spoľahlivé ako v Netlab. Niekoľkokrát nám celá aplikácia prestala reagovať, keď sme sa pokúsili vytvoriť graf dosiahnuteľnosti. Výsledky analýzy nie sú dostupné v jednotnom formáte, nie vždy sa dajú jednoducho uložiť ako v Netlab. Ponuka simulátora nie je dostupná, domnievame sa, že ešte nebol implementovaný.

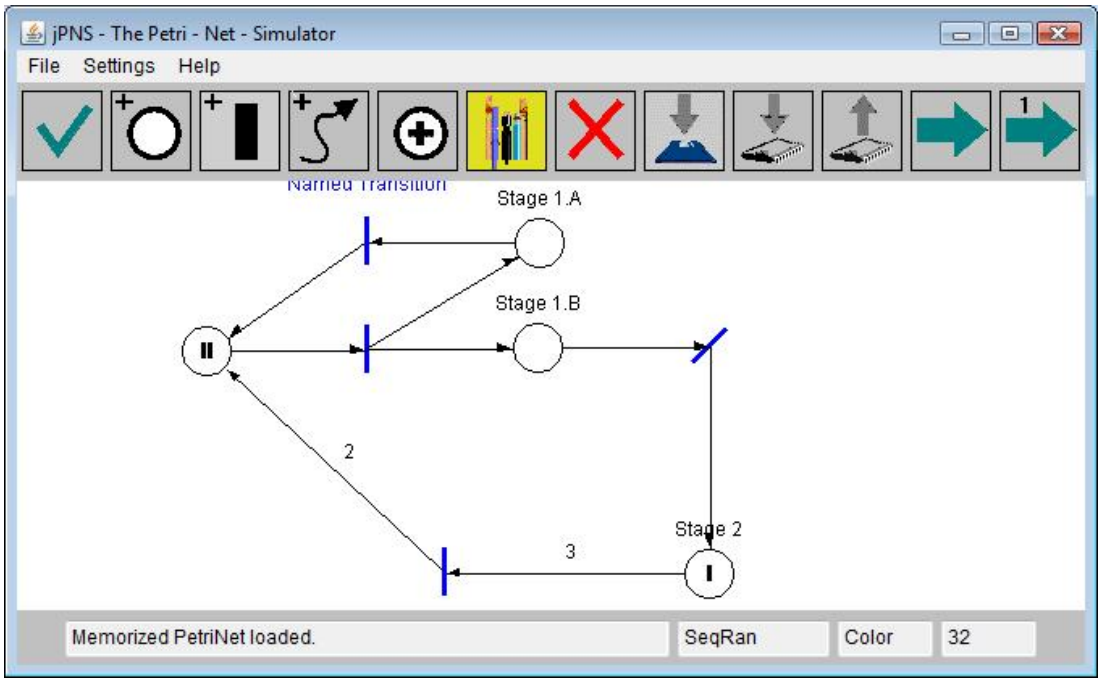
Napriek uvedeným nedostatkom je Petri Net Designer dobrá aplikácia, vhodná aj na reálne používanie. Vďaka dostupnosti zdrojových kódov v jazyku C++ a funkčnej zhode s našimi požiadavkami má predpoklad stať sa východiskom pri vývoji budúcej aplikácie.

2.3.4 Web aplikácie pre Petriho siete

Uvedené produkty doplníme o ďalšie tri, ktoré sú špecifické implementáciou ako web aplikácie. Vykonávanie programu on-line má svoje výhody, niekedy môže byť obmedzením alebo zdrojom potenciálneho rizika.

2.3.4.1 Thomas Bräunl's S/T Petri-Net Simulation System

S/T Petri-Net Simulation System [29] je web aplikácia napísaná v jazyku Java, implementovaná ako súčasť softvérového projektu s cieľom vytvoriť jednoduchý systém na simuláciu P/T Petriho sietí.



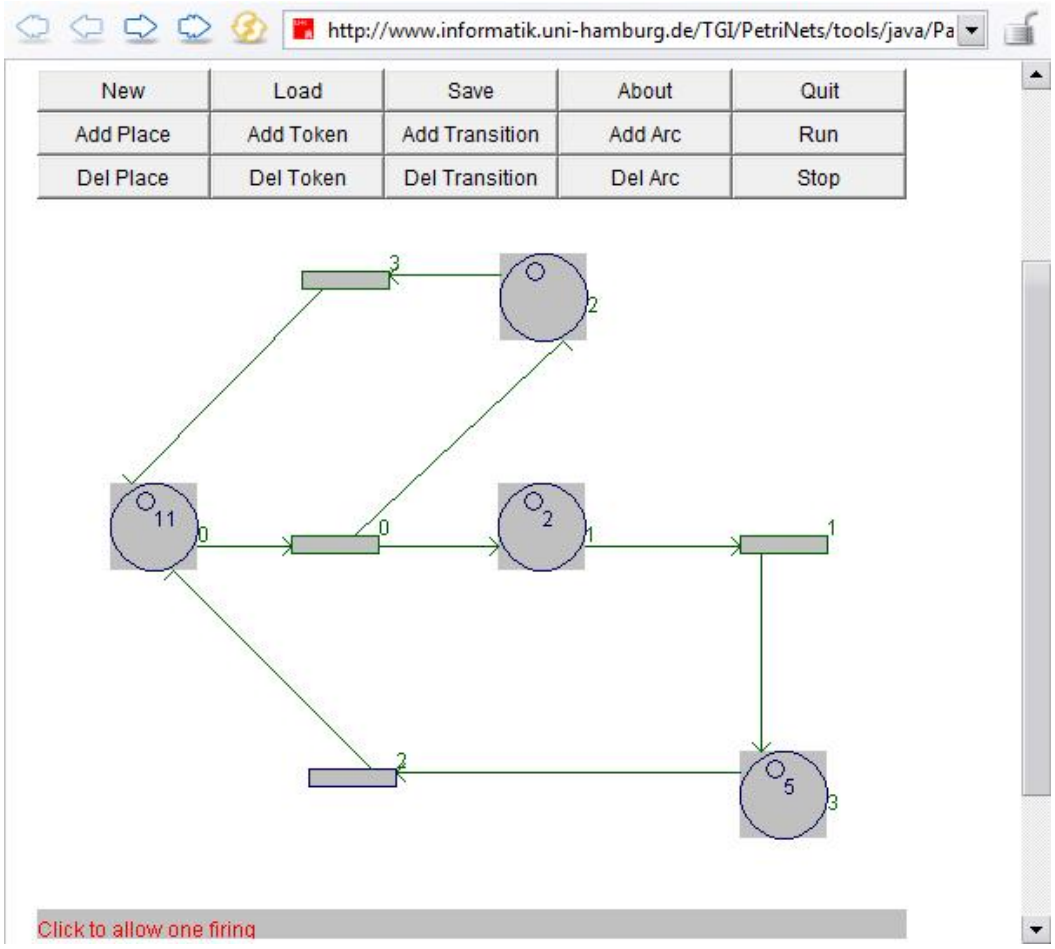
Obr. 16 - Používateľské rozhranie je strohé, avšak prehľadné a plne funkčné.

Spúšťa sa v samostatnom okne, ktorého príslušnosť k internetovému prehliadaču nie je zreteľná. Toto môže zvádzať ukončiť prehliadač, čo však bez akejkoľvek výstrahy ukončí aj aplikáciu. Používateľské rozhranie je síce strohé, no obsahuje všetky potrebné nástroje. Disponuje typickými prvkami: pracovná plocha, menu, nástrojová lišta a stavová lišta. Výstižné ikony robia rozhranie intuitívne, celá aplikácia sa ovláda veľmi jednoducho. Tlačidlami sa vyberá režim myši: presúvanie objektov, pridávanie miest, pridávanie prechodov, pridávanie hrán, pridávanie značiek, vymazávanie objektov a režim úpravy parametrov. Upravujú sa váhy hrán, značkovanie miest, dajú sa pomenovať miesta a prechody, zmeniť orientácia prechodu (horizontálne, vertikálne, diagonálne). Petriho sieť vieme simulovať, dostupné je aj krokovanie. Žiadne analytické nástroje nie sú dostupné. Súborové operácie sú implementované, pokiaľ však aplikácia beží v rámci prehliadača tieto nefungujú.

Nedostatky vidíme v nemožnosti akýmkoľvek spôsobom posúvať celú sieť, dokážeme hýbať iba s jednotlivými objektmi. Hrany sú priamky, nedajú sa použiť krivky alebo lomené čiary. Obmedzenia pri práci so súbormi a chýbajúce analytické funkcie.

2.3.4.2 Patrice Torguet's Petri Network Simulator

Ďalšia [30] Java aplikácia sa spúšťa priamo v okne prehliadača. Veľkosť pracovnej plochy je fixná, zhora ohraničená tromi radmi tlačidiel, zdola stavovým riadkom. Disponuje iba najnutnejšími funkciami pre vytvorenie a simuláciu Petriho siete. Pridanie miesta, prechodu, značky alebo hrany sa vykonáva stlačením príslušného tlačidla a následným kliknutím na cieľové miesto pracovnej plochy. Odstránenie obdobným spôsobom, pričom pre odstránenie rôznych objektov sa používajú rôzne tlačidlá. Hrany nemôžu mať váhu ani byť viacnásobné. Sieť sa dá simulovať, ale iba po krokoch spúšťaných kliknutím. Žiadna ďalšia funkcionalita nie je dostupná.



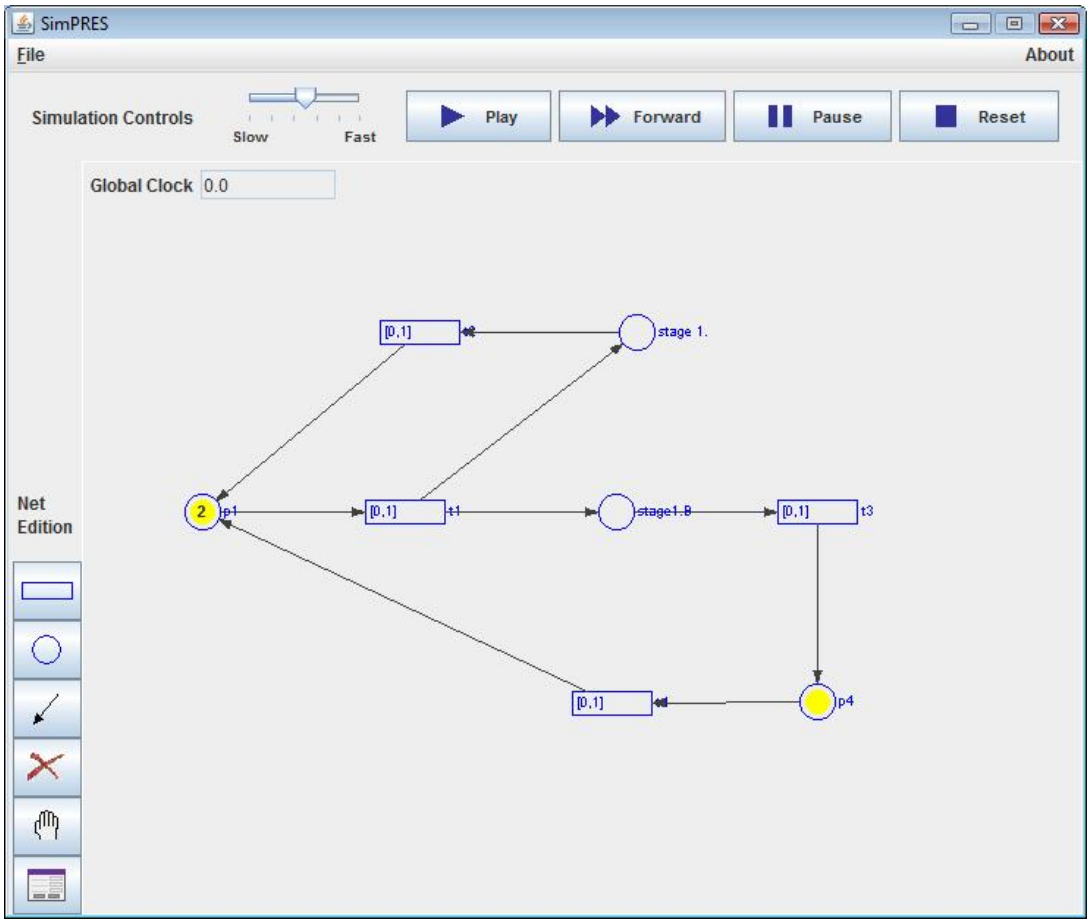
Obr. 17 - Malá pracovná plocha a veľké objekty obmedzujú Petriho sieť.

Program neprináša žiadne výnimočné vlastnosti. Nepríjemné je, že pred vytvorením akéhokoľvek objektu musíme zakaždým kliknúť na tlačidlo „Add“ (Pridať). Grafické objekty sú zbytočne veľké, v kombinácii s malou pracovnou plochou obmedzujú

reálnu použiteľnosť. Táto web aplikácia pochádza ešte z roku 1996, a tak jej celková primitívnosť neprekvapuje.

2.3.4.3 Luis Alejandro Cortés' SimPRES

SimPRES [31] je simulátor Petriho sietí s časovaním. Je navrhnutý ako samostatná Java aplikácia, no spustiť sa dá aj ako web aplikácia. Vtedy však nefungujú súborové operácie.



Obr. 18 - Aplikácia je funkčná a zároveň používateľsky prívetivá.

Program sa spúšťa v okne pevnej veľkosti, na okraji ktorého sú ľahko zrozumiteľné tlačidlá pre tvorbu a simuláciu Petriho siete. Veľkosť grafických objektov je primeraná. Tieto sa nedajú ďalej upravovať, orientácia prechodu sa nedá zmeniť, hrany sú vždy priamkou. Ovládanie aplikácie je prakticky rovnaké ako v S/T Petri-Net Simulation

System. Parametre miest a prechodov sa dajú meniť v dialógových oknách. Nastaviť sa dá menovka, počiatočné značkovanie, oneskorenie a pravidlá spúšťania prechodu. Vytvorenú sieť môžeme jednoducho uložiť a neskôr načítať zo súboru. Autor venoval pozornosť simulácii, ktorá sa dá detailne riadiť. Užitočné je tlačidlo „Reset“ pre návrat do počiatočného značkovania. Aplikáciu hodnotíme kladne, je funkčná a zároveň používateľsky prívetivá. Okrem vyššie spomenutých nedostatkov sme objavili problém s odozvou rozhrania, ktoré občas nezobrazuje skutočný stav siete.

2.4 Tablet PC

Slovom Tablet PC označujeme prenosný počítač obsahujúci dotykovú obrazovku. Existujú rôzne typy obrazoviek. Niektoré reagujú len na stylus (označenie pre pero dodávané s tabletom), niektoré reagujú aj na iné predmety než elektronické pero. Taktiež existujú rôzne typy pier. Rôznymi technológiami je stylus rozoznávaný a projektovaný na obrazovku, pričom moderné tablety dokážu určiť tlak, akým stylus pôsobí na dotykovú plochu, jeho polohu, vzdialenosť od dotykovej plochy a dokonca aj sklon tohto elektronického pera. Týmto pádom plne nahrádza klasickú počítačovú myš. V niektorých prípadoch, ako napríklad pri kreslení, je toto riešenie dokonca oveľa príjemnejšie a efektívnejšie ako klasické riešenie pomocou myšky. Klasický stylus obsahuje väčšinou dve tlačidlá. Pohybom hrotu pera, pokiaľ sa nedotýka dotykovej plochy (ale je vo vzdialenosti, pri ktorej ho dokáže tablet rozpoznať), je ekvivalentný pohybu kurzoru pomocou počítačovej myšky a v prípade, že sa hrot pera dotkne obrazovky, tak sa vykoná rovnaká akcia ako pri kliknutí ľavého tlačidla na myši. Prvé tlačidlo pera, ktoré sa nachádza priamo boku pera, plní funkciu pravého tlačidla myšky. Pri stlačení tohto tlačidla spolu s dotykom hrotu s dotykovou plochou sa vykoná príslušná akcia nastavená na pravom tlačidle myšky. Druhé tlačidlo sa nachádza na opačnej strane ako hrot pera. Jeho funkcia je podobná gume na ceruzke. Ak sa teda toho tlačidlo dotkne obrazovky, tak sa vykonáva mazanie nami nakreslenom obrázku prípadne textu. Vyššie spomínaný tlak pera na obrazovku je tiež možné využiť, napríklad na zobrazenie dodatočného menu. Sklon pera sa dá využiť napríklad na zmenu kurzoru, prípadne zmeny šírky štetca. Záleží to však od konfigurácie tohto vstupného zariadenia. Nevýhoda stylus-u je, že i keď sa s ním pomerne dobre píše, nie je praktický na písanie napríklad slohových prác alebo na programovanie. Na to je vhodnejšia klasická klávesnica. Dôvod je ten, že prekladač nami

napísaného textu nie je ešte tak vyvinutý a natoľko robustný, aby vedel rozpoznávať s vysokou presnosťou písané slová a prepisovať ich. Ďalšou nevýhodou je, že je potrebné robiť pauzy pri písaní, aby mal prekladač čas rozpoznať slová.

Existujú dva typy elektronických pier, a to pasívne a aktívne. Pasívne stylus-y nepotrebujú vlastné napájanie a ich polohu má na starosti daný Tablet PC. Aktívne stylus-y určujú vzhľadom na obrazovku svoju polohu samy, potrebné je ale vlastné napájanie a senzory. Veľkou výhodou je, že aktívne perá je možné použiť aj na iných zariadeniach ako na Tabletoch [33].

2.4.1 Delenie tabletov

2.4.1.1 Slate

Je to v podstate prenosný počítač, ktorý nemá klávesnicu a obsahujúci ako vstupné zariadenie dotykovú obrazovku, na ktorú kreslíme a píšeme perom. Preto musí byť rozhranie tomuto spôsobu práce aj plne prispôsobené. Pri potrebe písania textu je možné použiť virtuálnu klávesnicu alebo napojiť externú.

2.4.1.2 Thin-client slate

Je podobný typu Slate, avšak neobsahuje plnohodnotný počítač. Tento typ tvorí v podstate len obrazovka, ktorá sa dá napojiť na iný počítač, aby slúžila ako ďalšia forma vstupno-výstupného zariadenia.

2.4.1.3 Convertible

Tablet PC tohto typu je prenosný počítač, ktorý umožňuje po vyklopení obrazovky jej následné otočenie a zaklapnutie. Výhodou je používanie klávesnice.

2.4.1.4 Hybrid

Posledný typ Hybrid obsahuje odnímateľnú klávesnicu, takže sa podobá znova na typ Slate. Rozdiel je však v tom, že klávesnica sa pripája napevno k obrazovke. V praxi sa ale tento typ nepresadil.

2.4.2 Programovanie aplikácií pre Tablet PC

Na vytváranie aplikácií podporujúce využívanie pera ako náhradu za myš nie je potrebné inštalovať žiaden špeciálny vývojový softvér, lebo o tieto vlastnosti sa postará príslušný operačný systém obsahujúci príslušné knižnice. Ak však chceme používať špeciálne funkcie pera, ako napríklad rozoznávať gestá, tak je potrebné nainštalovať si tieto knižnice, z ktorých je potom možné volať špeciálne funkcie. Dôvod je ten, že pohyby myšou a perom vie operačný systém rozpoznať, avšak funkcia tlaku pera na dotykovú obrazovku na myške nenájdeme. Preto je potrebné niektoré vlastnosti naprogramovať do danej aplikácie.

Princíp, akým funguje rozpoznávanie nami pomocou pera napísaných slov je takýto. Používateľ začne písať. Tzv. Pen API (aplikačné programové rozhranie pre pero) začne zachytávať pohyb. Súradnice sa prenesú do Ink API (rozhrania pre prácu s „atramentom“). Ink API tieto pohyby vykreslí a uloží si ich. Ďalej si zhrnie nakreslené čiary a predá ich prekladaču (spoznávaču), ktorý tieto skupiny čiar interpretuje ako určité znaky.

Ďalšia možnosť je využívanie gest. Gestá sú určité pohyby stylus-u, ktoré majú špeciálny význam, ako napríklad vymazanie celého obsahu stránky, respektíve návrat späť a pod.. Gestá sa rozdeľujú na systémové, aplikačné a tzv. flicks (prebliknutia).

Systémové gestá sú tie, ktoré sú mapujú tradičné príkazy myšky na pero. Tieto sa využívajú automaticky. Ak chceme používať aplikačné gestá, tak je ich najprv potrebné povoliť. Aplikačné gestá môžu byť fixné alebo voľné. Obe skupiny môžeme upravovať, ale v prípade, že je gesto fixné, tak je tento pohyb pera definovaný ako odporúčanie na vykonanie určitej akcie, ako napríklad pri nakreslení zvislej paličky zdola nahor a znovu nadol sa vykoná operácia „späť“. Túto akciu je možné zmeniť, avšak v iných programoch je toto gesto zaužívané, tak sa to neodporúča. Voľné gestá sú také, pre ktoré nie je definované žiadna odporúčaná akcia a môžeme si ju naprogramovať podľa vlastnej potreby. Posledný typ, teda flick gesto je nový a nachádza sa zatiaľ len v operačnom systéme Microsoft Vista. Je ich zatiaľ osem a sú reprezentované pohybom v smere

svetových strán. Ak je toto gesto rozpoznané, tak sa vykoná príslušná akcia, napríklad kopírovať alebo vložiť. Gestá sú rozpoznávané tzv. Gesture recognizer-om (prekladač gest).

Aby sme však vedeli určiť, či sa vlastne jedná o Tablet PC alebo len o obyčajné PC, tak bola v operačnom systéme vytvorené funkcia GetSystemMetrics(), ktorá berie ako parameter metriku alebo konfiguračné nastavenia operačného systému (OS), o ktorých chceme vedieť. Pre zistenie, či je operačný systém nainštalovaný na Tablet PC zadáme definovanú konštantu SM_TABLETPC, respektíve číslo 86. Funkcia nám vráti nenulovú hodnotu, ak je práve bežiaci OS typu Windows XP Tablet PC edition alebo je OS typu Windows Vista a služba zaoberajúca sa vstupom pre Tablet PC je zapnutá. V opačnom prípade je návratová hodnota rovná nule. Takto sa dá ľahko určiť či je daný počítač Tablet PC, alebo nie [32].

2.5 Zhodnotenie analýzy

Po rozanalyzovaní jednotlivých prác predchádzajúcich tímov, bakalárskych a diplomových prác, či rôznych externých aplikácií sme postupne zmenili myšlienku vytvoriť modul pre Moodle a transformovali sme ju na vlastnú externú aplikáciu. Bolo to z viacerých dôvodov či nedostatkov, na ktoré sme pri analyzovaní postupne prichádzali. Chýbala nám v nich interakcia medzi učiteľom a študentom, väčšia kreativita pri vytváraní úloh, či technické pozadie aplikácií. Veľmi dobre po technickej stránke je spracovaná aplikácia PNDesigner, z ktorej budeme čerpať inšpiráciu minimálne pri prototype. Takisto server fiit.stuba.sk poskytuje vhodné námety na typy úloh a celkového pohľadu na stánku s vyučovacím zámerom. Z tímového projektu SPOJENCI sa nám páčila univerzálnosť kódu, ktorý sa môže ľahko zmeniť pre iný typ úloh iného jazyka. V tíme @Tím nás zaujala metóda drag & drop, ale na druhej strane projekt mal svoje nedostatky, na základe ktorých sme sa rozhodli pre inú formu riešenia.

Hlavné body, na ktorých sa chceme oprieť sú flexibilita, nenáročnosť, kreatívnosť či nezávislosť na internete. Externá aplikácia bude typu klient-server, kde sa chceme zamerať aj na bezpečnosť prenosu a spoľahlivosť pri skúšaniach. Bližšiu špecifikáciu si uvedieme v nasledujúcej kapitole.

3 Riešenie

3.1 Špecifikácia požiadaviek

Podľa zadania máme navrhnúť a implementovať aplikáciu na vyhodnocovanie zadaní, získavanie vedomostí a ich praktické overovanie. Po konzultáciách a na prvých stretnutiach sme dospeli k názoru, že vytvoríme externú aplikáciu typu klient – server. I keď v ponuke sme prezentovali, že vytvoríme modul pre Moodle, rozhodli sme sa pre externú aplikáciu z dôvodu prenosnosti našej aplikácie (kapitola 2.1.5.2 - Moodle).

Naša aplikácia bude podporovať správu a vyhodnocovanie rôznych úloh a testov, ktoré sa budú dať tiež v tejto aplikácii vytvárať. Týmto spôsobom zabezpečíme možnosť overovania získaných vedomostí na predmete Špecifikačné a opisné jazyky. Keďže naša aplikácia je zameraná hlavne na problematiku Petriho sietí, bude v nej možné vytvárať a testovať aj úlohy, ktoré si vyžadujú tvorivú činnosť študenta, ako napríklad nakresliť Petriho sieť so zadanými vlastnosťami. Takto dokážeme pomocou našej aplikácie overiť aj praktické zručnosti študenta v rámci problematiky Petriho sietí.

Jednotlivé požiadavky na aplikáciu, ktoré sme vyšpecifikovali na stretnutiach sme rozdelili do nasledujúcich kapitol:

3.1.1 Klient server

Potrebujeme vytvoriť aplikáciu, ktorú budú môcť študenti používať aj doma, ale zároveň chceme, aby bola využiteľná aj na skúškach. Preto sme sa rozhodli, že hlavná funkcionalita programu bude zabezpečená v klientskej časti, a to okrem kreslenia a testovania aj overovanie správnosti zadaní. Serverová časť bude slúžiť výlučne len pri skúškach, aby sme pomocou nej zabezpečili určitú úroveň bezpečnosti (Bezpečnosť 3.1.5.).

3.1.2 Nezávislosť na Internete

Keďže sme sa rozhodli hlavné jadro aplikácie implementovať na klientskej strane, používateľ nepotrebuje internetové pripojenie na overovanie testov. To znamená,

že zadania úloh, ale aj ich správne výsledky budú obsiahnuté v externom súbore, ktorý si stačí nahrať na počítač a otvoriť v našej aplikácii. Internet, prípadne lokálna sieť bude potrebná iba pri skúškach kvôli autentifikácii študentov.

3.1.3 Flexibilita

Aby bolo možné čo najefektívnejšie testovanie študenta, sú potrebné rozmanité typy otázok. Do našej aplikácie chceme zahrnúť klasické testové typy otázok s možnosťou výberu jednej alebo viac správnych odpovedí. Pokiaľ bude viac správnych odpovedí, ich počet bude pre študenta dopredu známy. Ďalší typ otázok vyžaduje spájanie rôznych, spolu súvisiacich výrazov. Taktiež chceme zaradiť kreslenie Petriho sietí pomocou implementovaných nástrojov. Práve tento typ otázok predstavuje nosný pilier našej aplikácie a zároveň jej najväčší prínos.

3.1.4 Nenáročnosť

Z vlastných skúseností vieme, že na niektorých skúškach boli problémy s vyhodnocovaním testov. Veľakrát sa stalo, že systém spadol, a z toho dôvodu nebolo možné testy odovzdať, niektoré sa nenávratne stratili alebo celý systém bol extrémne spomalený. Nato, aby sme sa tomuto vyhli by bol potrebný veľmi rýchly hardvér počítača, alebo odľahčenie servera od vyhodnocovania testov. My sme sa rozhodli, že sa nebudeme spoliehať na použitý hardvér servera, ale uskutočníme vyhodnocovanie na klientskej strane programu.

3.1.5 Bezpečnosť

Táto požiadavka vyplýva z rozhodnutia, že overovanie testov bude prebiehať na klientskej strane aplikácie. To znamená, že súbor so zdaním obsahuje okrem samotných otázok aj ich odpovede. Rozhodli sme sa pre dva typy súborov. Pokiaľ to budú len zadania na precvičenie, tieto nie je potrebné nijak chrániť. To znamená, že študent bude mať možnosť nahliadnuť na výsledok, pokiaľ rozlúšti formát, akým sa ho my rozhodneme zapísať. Avšak súbory, ktoré sa nepoužívajú na skúškach, budú zašifrované,

a iba aplikácia s nimi vie pracovať, a teda nebude možné, aby tento súbor vedel študent prečítať bez toho, aby poznal šifrovací kľúč a šifrovací algoritmus. Pre tieto súbory sa vytvorí tiež kontrolná suma, aby sa dala overiť ich integrita. Ďalej je potrebné, aby v nich bol zapísaný čas na vypracovanie celého testu, ktorý keď uplynie, testovanie skončí, test sa uloží a pošle na server.

3.1.6 Prenosnosť

Chceli by sme, aby bola naša aplikácia moderná, a preto sme sa rozhodli, že ju implementujeme ako multiplatformovú. Okrem obyčajného domáceho počítača, ju bude možné používať aj na Tablet PC. Tu sa chceme zamerať a využiť rôzne výhody, ktoré táto platforma ponúka, ako napríklad tlak pera, ktorým sa kreslí.

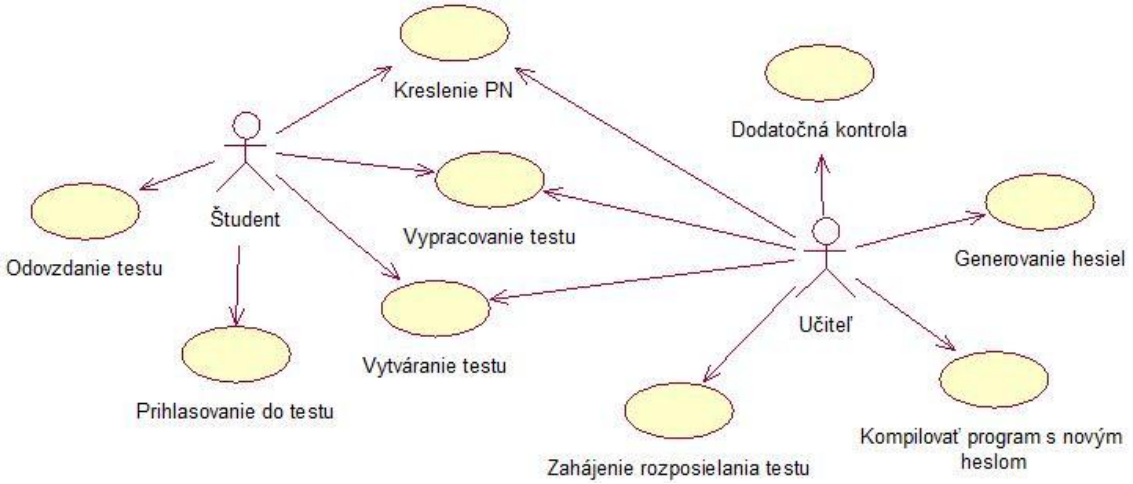
3.1.7 Intuitívnosť

Aby bol náš program čo najviac využívaný, budeme sa snažiť implementovať jeho ovládanie pre používateľa čo najjednoduchšie, teda anglickým výrazom „user-friendly“. Z dôvodu prenosnosti našej aplikácie, musíme tiež navrhnúť ovládanie, ktoré by vyhovovalo všetkým platformám.

Aby bolo ovládanie na Tablet PC prijateľné je dôležité, aby naša aplikácia vedela rozpoznávať rôzne pokusy o nakreslenie jednotlivých objektov s použitím pera. Spôsob, akým sme sa rozhodli identifikovať tieto pokusy, je bližšie popísaný v kapitole 3.2 Hrubý návrh.

Okrem rozpoznávania je tiež dôležité zabezpečiť funkcionalitu aj v prípade poruchy pera. Z tohto dôvodu je nutné implementovať aj záložné možnosti kreslenia. Rozhodli sme sa poskytnúť panel s nástrojmi, v ktorom si užívateľ jednoducho klikne na objekt, ktorý chce nakresliť a umiestni ho na pracovnú plochu programu.

Ďalej sme sa rozhodli implementovať určitú formu pomocníka. To znamená, že pri kreslení Petriho sietí doma, nás bude pomocník informovať, ak robíme neregulárne, zakázané operácie, ako napríklad vytvorenie hrany medzi dvoma miestami. Tento pomocník však bude automaticky vypnutý na skúškach, čiže aj nezmyselné operácie budú povolené.



Obr. 19 - Diagram prípadov použitia pre študenta a učiteľa

3.2 Hrubý návrh

V tejto kapitole by sme radi priblížili doterajší návrh aplikácie a detailnejšie popísali niektoré riešenia niektorých požiadaviek z predošlej kapitoly.

Keďže sme sa rozhodli vytvoriť samostatnú aplikáciu a nie modul napríklad pre Moodle, budeme ju implementovať v jazyku C++. Vyplýva to tiež z našich osobných názorov, sympatií a praxe v tomto jazyku.

3.2.1 Klient – server

Ako sme už písali, aplikácia bude typu klient – server. Na serveri bude databáza používateľov, ku ktorej bude mať prístup iba administrátor, teda pedagóg. Okrem toho bude možné na ňom automaticky generovať náhodné heslá pre zadaných používateľov, ktoré vyexportujú do textového súboru. Tento potom môže pedagóg vytlačiť a rozdať študentom pred skúškou. Každé konto okrem prihlasovacieho mena a hesla bude tiež obsahovať IP adresu, z ktorej sa používateľ prihlásil, a procesy ktoré spustil po prihlásení, aby sme mohli kontrolovať prípadné zneužitie nedovolených aplikácií.

Server bude tiež slúžiť ako distribútor zadaní. To znamená, že bude obsahovať ďalšiu databázu, tentokrát však jednotlivých úloh. Z týchto sa náhodne vyberie určitý

počet, pre každého používateľa v náhodnom poradí a uložia sa do súboru spolu s odpoveďami. Kvôli bezpečnosti budú v súbore na začiatku najprv všetky otázky a k nim prislúchajúce odpovede sa budú nachádzať až na konci. Tieto súbory sa potom zašifrujú a začnú sa postupne posielať na IP adresy priradené pri jednotlivých kontách.

Aby sme zabezpečili vysokú úroveň bezpečnosti, rozhodli sme sa zadania na skúškach pred poslaním zašifrovať. Ako šifrovací algoritmus použijeme nejaký blokový algoritmus, avšak konkrétny sme ešte nevybrali. Kľúč, ktorý bude vstupovať do algoritmu bude vytvorený pomocou hesla zadaného pri kompilácii aplikácie a prihlasovacích údajov študenta, z toho vyplýva, že pre každého študenta bude jedinečný. Vytvoríme nástroj, pomocou ktorého sa bude dať aplikácia externe skompilovať s použitím rôznych hesiel. Tento nástroj vytvorí spúšťači súbor klientskej aj serverovej časti programu. Tým pádom študent nevie dopredu zistiť heslo, ktoré bude aplikácia používať na dešifrovanie zadaní. Ak by chcel náš program nabúrať za pomoci dekompilátora alebo iných nástrojov, bude jeho pokus zaznamenaný pri spustení akejkoľvek inej aplikácie. Samozrejme sme si vedomí, že ani takýto systém nie je nepremožiteľný, ale myslíme si, že pre dané prostredie a požiadavky je postačujúci.

Ako teda bude prebiehať samotná komunikácia? Študenti sa prihlásia za pomoci prihlasovacieho mena a vygenerovaného hesla, ktoré im budú rozdané na skúške. Server tieto údaje overí so svojou databázou, uloží si IP adresu a označí študenta ako autentifikovaného, pokiaľ boli správne. Pedagóg po autentifikácii všetkých študentov spustí distribúciu zadaní, server začne postupne generovať a posielať zadania autentifikovaným študentom. Pre každého študenta si jeho zadanie ponechá, aby sme mali dôkaz, aké otázky študent dostal. Klientsky program po prijatí overí kontrolnú sumu súboru a ak je správna, dešifruje iba časť s otázkami, začne odpočítať čas a spustí samotné testovanie. Po uplynutí času alebo ak študent ukončí vypracovávanie testu, uložia sa jeho odpovede do súboru, tento sa zašifruje a vypočíta sa jeho kontrolná suma. Súbor sa potom odošle na server, kde sa tiež uloží, aby sa náhodou záhadne nestratil. Server overí jeho kontrolnú sumu, ak je správna, odošle klientskej aplikácii potvrdenie o úspešnom prijatí. Až potom klient dešifruje časť súboru s odpoveďami a začne opravovať. Nakoniec zobrazí výsledky a týmto je test ukončený.

Identifikátor	I 1.		
Názov	Odovzdanie zadania		
Opis	Študent odovzdá zadanie pomocou aplikácie, ktorá ho následne skontroluje a vyhodnotí		
Priorita	1 = vysoká	Frekvencia	desiatky krát počas testu
Vst. podm.	Súčasný stav vypracovaného zadania ešte nie je odovzdané		
Výst. podm.	Odovzdaná súčasná verzia vypracovaného zadania		
Používatelia	Študent		
Základná postupnosť	Krok	Činnosť	
	1	Uplynie čas alebo študent stlačí tlačidlo pre odovzdanie zadania	
	2	Zobrazí sa okno pre odovzdanie zadania	
	3	Odpovede sa uložia do súboru	
	4	Súbor sa zašifruje a vypočíta sa jeho kontrolná suma	
	5	Súbor sa odošle na server	
	6	Server overí kontrolnú sumu súboru a uloží si ho	
	7	Odošle sa potvrdenie o správnom prijatí súboru	
	8	Klientská aplikácia dešifruje dešifruje časť súboru s odpoveďami	
	9	Porovnajú a vyhodnotia sa odpovede	
	10	Aplikácia informuje študenta o jeho úspešnosti	
	11	Ukončenie odovzdávania zadania a celkového testu	

Tab. 1 - Odovzdanie zadania

3.2.2 Vytváranie testov

Pre tento účel budú poskytnuté rôzne nástroje. To znamená, že používateľ si najprv vyberie typ otázky, ktorý chce vytvoriť. Otázky z viacerými možnosťami výberu odpovede budú obsahovať políčko, kde autor zadá otázku, zvolí počet možných odpovedí, tieto popíše a označí správne.

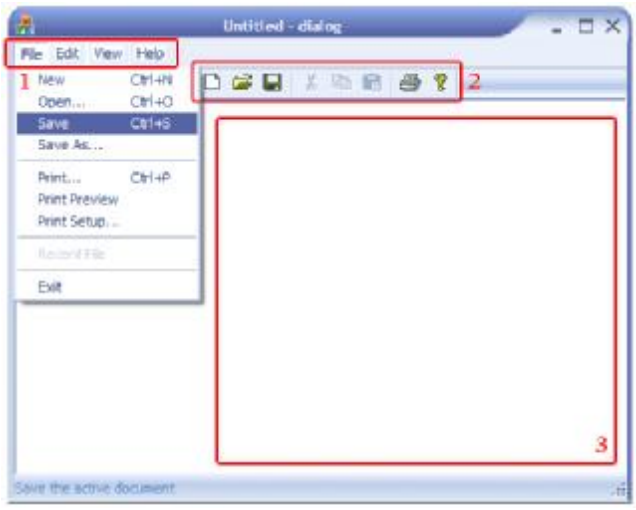
Pri otázkach na spájanie spolu súvisiacich otázok si autor vyberie počet položiek na ľavej a pravej strane. Každú položku popíše a na ľavej strane napíše čísla položiek, s ktorými položkami sú spojené na pravej strane.

Čo sa týka otázok, ktoré vyžadujú kreslenie Petriho sietí, rozhodli sme sa implementovať dve možnosti. Prvá bude obsahovať časť Petriho siete, ktorú bude treba doplniť, aby spĺňala zadanie. Pri jej vytváraní autor napíše zadanie, nakreslí počiatočnú a finálnu sieť. Tento typ otázok si vyžaduje jednoznačnú a jedinú možnosť odpovede, pretože odpoveď sa bude porovnávať iba s finálnou sieťou, ktorú autor nakreslil. Pokiaľ by existovala aj iná, správna odpoveď, bola by táto označená za nesprávnu. Druhý typ bude poskytovať výber z niekoľkých vlastností Petriho siete, ktoré sa dajú overiť nejakým algoritmom. Autor si teda len vyberie jednu z vlastností a napíše konkrétne zadanie.

Pri vytváraní celého testu sa jednotlivé otázky budú môcť postupne spájať do jedného súboru. Tu je potrebné, aby boli očíslované zadania otázok a k nim prislúchajúce odpovede, aby sme vždy zadanie otázky pridali na správne miesto a odpovede až na koniec súboru, za zadania otázok. Pri pripájaní nových otázok sa najprv zistí číslo poslednej otázky, táto hodnota sa inkrementuje a použije pri novej otázke. Takto zabezpečíme, aby bolo jednoznačné spojenie otázky a odpovede.

3.2.3 Pracovná plocha

Pracovná plocha aplikácie rozdelená do mriežky, každý objekt ma svoje vlastnosti. Pre najjednoduchšiu interpretáciu pracovnej plochy aplikácie sme sa rozhodli vytvoriť približný nákres, ktorý je zobrazený na obrázku **Obr. 20**.



Obr. 20 - Náčrt pracovnej plochy

Označenie 1 zobrazuje ponuku, kde si používateľ môže zvoliť rôzne akcie, ako napríklad otvoriť test alebo vytvoriť test. Označenie 2 zobrazuje panel nástrojov, kde budú zobrazené ikony využívané hlavne pri kreslení Petriho sietí, ako napríklad miesto či prechod. Nakoniec plocha označená číslom 3 bude obsahovať samotné otázky testu alebo priestor pre kreslenie Petriho sietí.

Pracovnú plochu pre kreslenie Petriho sietí chceme rozdeliť do mriežky. V každom bode mriežky sa môže nachádzať práve jeden objekt, v našom prípade buď miesto, alebo prechod. Tieto objekty budú mať v aplikácii definované svoje vlastnosti, ako napríklad značkovanie miesta a iné. Každý objekt bude mať jedinečný identifikátor, aby sme mohli zapísať do objektu prípadné spojenie hranou s iným objektom.

3.2.4 Ovládanie

Aplikácia sa bude ovládať pomocou ponuky akcií a panelu nástrojov. Keďže na ich používanie stačí jedno tlačidlo myši, nebude ovládanie týchto funkcií pomocou pera na Tablet PC nijak odlišné.

Avšak pri kreslení Petriho sietí, bude ovládanie troška odlišné. Keďže nám bolo oznámené, že druhé tlačidlo na pere k Tablet PC, ktoré sa používajú na škole niekedy nefunguje najspoľahlivejšie, musíme vytvoriť alternatívu ako jeho náhradu. Druhé tlačidlo sme chceli využiť pri nastavovaní vlastností objektov, ako napríklad značkovanie miesta. Pre tento účel teda vytvoríme v paneli nástrojov ikonku, na ktorú používateľ klikne, potom klikne na objekt, a podľa toho na aký objekt klikne, otvorí sa okno, kde bude môcť nastavovať vlastnosti objektu.

Čo sa týka kreslenia pomocou klasickej myši, bude tiež možné. Myslíme si však, že v tomto prípade je vhodnejšie používať panel nástrojov. Toto rozhodnutie je ale ponechané samozrejme na používateľovi.

Ďalej je potrebné, aby sa nakreslená sieť dala upraviť bez nutnosti mazať objekty. To znamená, že je potrebné, aby sme vedeli objekty premiestňovať po mriežke bez toho, aby sme narušili už nakreslenú sieť. Na túto možnosť vytvoríme ďalšiu ikonku v paneli nástrojov, na ktorú keď klikneme nebudeme v pracovnej ploche kresliť, ale iba hýbať objektmi.

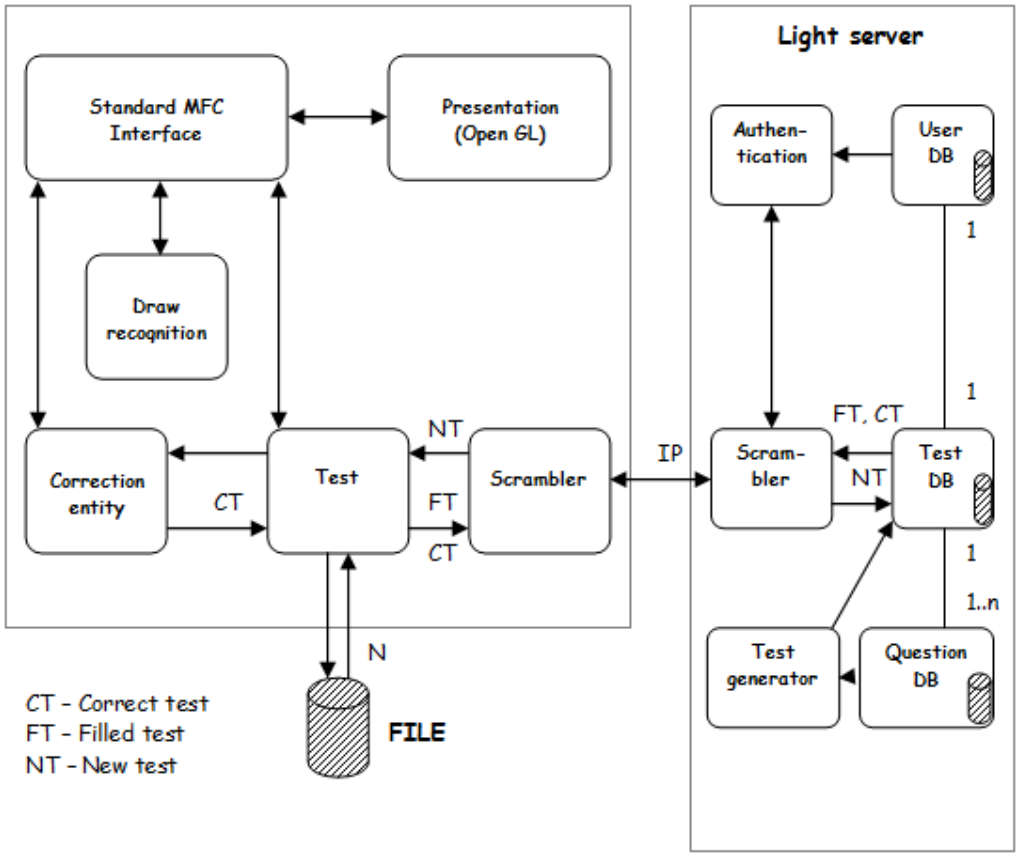
3.2.5 Rozpoznávanie objektov pri kreslení

Pri kreslení Petriho sietí potrebujeme vedieť rozpoznať dva objekty – miesto a prechod, a jednu akciu – hranu. Pokiaľ používateľ nakreslí akýkoľvek súvislý obrázok, ktorý sa pretne iba v jednom bode, tento sa rozpozná ako miesto a automaticky sa nahradí nami zdaným obrázkom pre miesto. Pokiaľ sa obrázok nepretne v žiadnom bode, tento sa rozpozná ako prechod a tiež sa automaticky nahradí. Tretia možnosť vyžaduje spájanie dvoch objektov a vyžaduje si na ne kliknúť. To znamená, že ak používateľ chce vytvoriť hranu, klikne na prvý objekt, potom na druhý a vytvorí sa medzi nimi hrana.

Alternatívna možnosť je používanie už implementovaných gest v operačnom systéme pre Tablet PC.

4 Návrh a implementácia

4.1 Architektúra systému



Obr. 21 – Architektúra systému

Ako je vidno z obrázku 21, naša aplikácia sa skladá zo serverovej časti a z klientskej časti ktoré spolu komunikujú po sieti. Každá časť sa skladá z niekoľkých základných entít, ktoré sa starajú o vykonávanie potrebných akcií.

4.1.1 Server:

Scrambler – táto entita má na starosti šifrovanie a dešifrovanie údajov posielaných po sieti.

Authentification – stará sa o autentifikáciu používateľov klientskej časti, priamo spolupracuje s entitou User DB

User DB – databáza používateľov aplikácie

Test DB – databáza uložených testov na serveri

Question DB – databáza otázok uložených na serveri

Test generator – entita slúžiaca na vytvorenie testov, pričom náhodne vyberá otázky z QuestionDB

4.1.2 Klient:

Scrambler – tak, ako aj v prípade servera, táto entita má na starosti šifrovanie a dešifrovanie údajov posielaných po sieti

Test – entita na internú reprezentáciu testu a jeho obslužné funkcie, ktorými sa majú jednotlivé otázky, celá petriho sieť a pod. ďalej čítať a modifikovať

Correction entity – táto entita sa stará o opravovanie a ohodnotenie vypracovaných testov

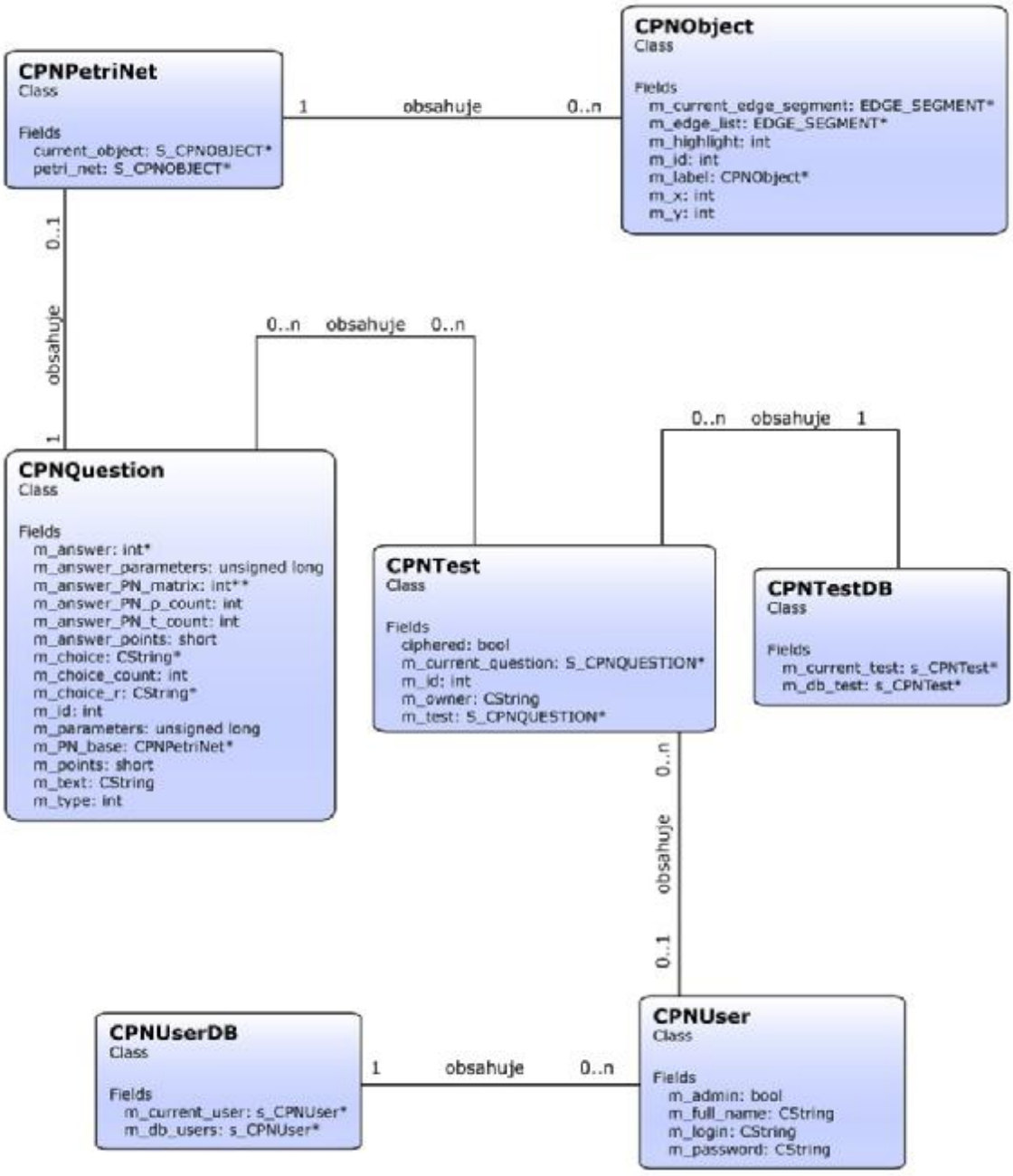
Standard MFC interface - Microsoft Foundation Classes je knižnica, ktorá zabaľuje časti Windows API do ucelených C++ tried, ktoré zabezpečujú plnú kompatibilitu s väčšinou platforiem OS Windows.

Draw recognition – entita zabezpečujúca rozoznávanie nakreslených objektov

Presentation (Open GL) – knižnica Open GL je použitá na vykresľovanie grafiky Petriho sietí.

4.2 Štruktúra údajov

4.2.1 Fyzický model údajov



Obr. 22 – Fyzický model údajov systému

4.2.2 Opis dátových entít a ich atribútov

CPNObject

Abstraktný objekt zastrešujúci prechody, miesta, uzly a menovky .Všetky tieto objekty okrem iného obsahujú:
***int m_id* – Jednoznačný číselný identifikátor objektu.**
***int m_x* - Horizontálna poloha.**
***int m_y* - Vertikálna poloha.**
***int m_highlight* – Príznak, či objekt nie je vybratý /je vybratý/ je vybratý na presúvanie.**
***CPNObject* m_label* - Ukazovateľ na objekt menovky (Ukazovateľ na menovku).**
***EDGE_SEGMENT* m_edge_list* - Zoznam objektov, do ktorých vedú hrany (segment hrany) od tohto objektu.**
***int m_edgeCount* - Počet segmentov vychádzajúcich z tohto objektu (len pre ukladanie a načítavanie zo súboru)**

***typedef struct edge_segment* - Štruktúra na modelovanie hrán po častiach - segmentoch, z ktorých je hrana zložená.**
***CPNObject* m_target_object* - Segment hrany - odkaz na ďalší objekt na hrany. Hrana vedie od tohto objektu k objektu na ktorý ukazuje smerník. Od tohto pokračuje ako nový segment tej istej hrany.**
***int m_weight* - Váha hrany – segmentu.**
***edge_segment* m_next* - Odkaz na ďalší segment vychádzajúci z tohto objektu.**
} EDGE_SEGMENT;

***EDGE_SEGMENT* m_current_edge_segmen* - Segment hrany s ktorým sa práve pracuje.**

CPNPlace

Miesta obsahujú tieto premenné
int m_radius - polomer krúžku = miesto
int m_capacity - kapacita miesta, 0 – 32767 = nekonečno
int m_tokens - aktuálny počet značiek v mieste

CPNTransition

Prechody obsahujú tieto premenné
int m_width - šírka obdĺžnika pre prechod
Int m_height - výška obdĺžnika pre prechod

CPNNode

Nemá žiadne premenné navyše oproti objektu.

CPNLabel

Menovky sú tiež len objekty, ale sú vždy pripojené k nejakým Miestam, Prechodom pripadne Uzlom. Samostatne nemajú zmysel.

CString m_name - Meno objektu (ako bude zobrazene používateľovi).
bool m_visible – Príznak, či sa ma menovka zobraziť alebo nie.

CPNQuestion

Dátový typ pre reprezentáciu otázky.
***int m_id* - Jednoznačný identifikátor otázky (dve v celej databáze otázok nemajú rovnaké ID).**
***CString m_label* - Menovka otázky, ako sa zobrazí v ponuke otázok.**
***Int m_type* - Typ otázky.**
***CString m_text* - Text otázky.**
***Short m_text_breaklines* - Počet riadkov textu otázky (pre účely zobrazenia v programe).**
***int m_choice_count* - Počet možností pri otázkach s výberom odpovede.**
***CString* m_choice* - Možnosti, z ktorých si používateľ vyberie / Súvisiace frázy na ľavej strane / Správna odpoveď typu WRITE_ANSWER - použitie je podľa typu otázky**
***CString* m_choice_r* - Súvisiace frázy na pravej strane.**
***int* m_choice_selection* - Odpovede študenta pri výberových otázkach.**
***CPNPetriNet* m_PN_base* – Predkreslená Petriho sieť, ktorú ma používateľ dokončiť (iba pre FINISH_PN, pre DRAW_PN je prázdna).**
***unsigned long m_parameters* - Parametre PN, ktoré sa majú vyhodnocovať (či sa má alebo nemá kontrolovať „deadlock“ alebo iný parameter).**
***unsigned long m_selected_parameters* - Parametre nakreslenej PN tak ako ich určí algoritmus na vyhodnocovanie. Výsledky vyhodnotenia ulož sem.**
***short m_points* - Pridelený počet bodov po opravení.**

***int* m_answer* - Správne odpovede pre výberové odpovede.**
***CPNPetriNet* m_answer_PN* - Výsledná Petriho sieť.**
***unsigned long m_answer_parameters* - Správne hodnoty parametrov (ohraničenosť, deadlock a pod).**
***short m_answer_points* - Celkový počet bodov, ktorý môže byt pridelený za túto otázku. O rozdelení bodov sa rozhoduje podľa typu otázky a jej konkrétneho zadania a vypracovania.**
***int m_question_position* - Pozícia otázky v liste otázok (pre účely zobrazenia v programe)**

CPNPetriNet

Petriho sieť ako súbor objektov.
***S_CPNOBJECT* petri_net* - Petriho sieť - ukazovateľ na prvý objekt spájaného zoznamu objektov tejto Petriho siete.**
***S_CPNOBJECT* current_object* - Ukazovateľ na aktuálny objekt, s ktorým sa pravé pracuje v tejto Petriho sieti**
***int m_objectCount* - Počet objektov v Petriho sieti (len pre ukladanie a načítavanie zo súboru).**

CPNTest

<u>Test ako súbor otázok.</u>
***int m_id* - Jedinečný identifikátor. Používa sa na rozlíšenie jednotlivých testov.**
***Bool m_ciphered* – Príznak, či sa má tento test ukladať šifrovaný alebo nie.**
***CString m_owner* - Vlastník testu. Používateľ, ktorý má test vypracovať.**
***boolm_active* - Či je test aktívny k vypracovaniu alebo je už vypracovaný a uložený pre archiváciu.**

time_t m_remaining_time **- Čas testu, zostávajúci čas testu.**
int m_questionCount **- Počet otázok testu (len pre ukladania a načítavanie do súboru)**
S_CPNQUESTION* m_test **- Test - smerník na prvú otázku v spájanom zozname.**
S_CPNQUESTION* m_current_question **- Otázka s ktorou sa práve pracuje.**

CPNUser

Objekty používateľ sú zabezpečené už pri programovaní. Prístup je možný len z databázy používateľov cez bezpečné funkcie. Všetky zapisujúce funkcie vyžadujú overenie identity používateľa.

CString m_login **- Prihlasovacie meno používateľa. Očakáva sa jednoznačné, inak program môže fungovať nekorektne.**
CString m_full_name **- Celé meno používateľa. Doplňujúci udaj pre jeho lepšiu identifikáciu. Nemusí byť uvedené.**
CString m_password **- Heslo používateľa.**
Bool m_admin **– Príznak, či používateľ má alebo nemá administrátorské práva.**

CPNUserDB

Databáza používateľov je implementovaná len na serveri.
int m_userCount **- Počet používateľov (len pre súborové operácie)**

s_CPNUser* m_db_users **– Databáza používateľov - smerník na prvého používateľa v spájanom zozname.**
s_CPNUser* m_current_user **- Aktuálny používateľ, s ktorým sa pracuje.**

CPNTestDB

Databáza testov.
s_CPNTest* m_db_tests - Databáza testov – smerník na prvý test v spájanom zozname testov.
s_CPNTest* m_current_test - Aktuálny test, s ktorým sa pracuje.

4.3 Návrh algoritmov spracovania

Pri spracovaní jednotlivých úloh, sme museli navrhnúť niekoľko algoritmov na ich spracovanie. Jedným z hlavných bol algoritmus na rozpoznávanie nakreslených objektov, aby sme vedeli identifikovať, aký objekt používateľ nakreslil. Po dokončení všetkých úloh sa jednotlivé Petriho siete spracovávajú pomocou algoritmu, ktorý vyhodnocuje vlastností týchto sieti. Ešte pred tým sme museli navrhnúť algoritmus, ktorý porovná, či daná Petriho sieť sa objektovo zhoduje so sieťou, ktorú mal používateľ nakresliť. Až po vykonaní tohto algoritmu sa spustí vyhodnocovanie vlastností Petriho siete.

4.3.1 Algoritmy na rozpoznávanie nakreslených objektov

Keďže jednou z hlavných funkcií našej aplikácie je podpora Tablet PC a teda kreslenie Petriho sietí voľnou rukou, museli sme implementovať algoritmy na rozpoznávanie niekoľkých typov objektov. Petriho sieť sa skladá z miest, prechodov a hrán medzi nimi.

Je teda treba rozoznávať nakreslené miesto, čiže kružnicu, a prechod čiže úsečku. Hranu nie je potrebné rozoznávať, lebo hrana sa začne vykresľovať automaticky keď začne používateľ kresliť úsečku z objektu. Jej grafická podoba je reprezentovaná pomocou triedy CPNNode. Ďalším objektom ktorý pribudol na rozoznávanie je krížik, ktorý sa používa na mazanie nakreslených objektov.

Miesto (kružnica) – kružnicu je možné matematicky rozdeliť na štyri časti podľa toho ako sa správajú súradnice x a y bodov na nej.
Rozdelíme teda kružnicu podľa správania sa súradníc na štyri časti:

1. x-ová súradnica rastie, y-ová súradnica rastie
2. x-ová súradnica rastie, y-ová súradnica klesá
3. x-ová súradnica klesá, y-ová súradnica rastie
4. x-ová súradnica klesá, y-ová súradnica klesá

Potom pri rozoznávaní objektu sledujeme, či nájdeme aspoň 3 z týchto častí na čiare, ktorú nakreslil používateľ a keď sa tam nachádzajú, tak je táto krivka rozoznaná ako kružnica a teda miesto. Následne sa na toto miesto vloží obrázok miesta.

Prechod – prechod je rozoznaný na základe toho, že používateľ nakreslil čiaru. Po začatí kreslenia sa tak ako pri každom rozoznávaní ukladajú súradnice bodov ktorým i prešiel kurzor. Po dokončení kreslenia čiary resp. krivky sa potom analyzujú jej body. Prechod sa rozozná vtedy, keď používateľ nakreslil úsečku v ľubovoľnej orientácii. Úsečku matematicky reprezentujeme tak, že zoberieme dva body, každý vzdialený 10 bodov od začiatku resp. konca čiary, ktorú nakreslil používateľ. Z týchto dvoch bodov vytvoríme priamku a následne porovnáme, či všetky body na priamke boli v určitej vzdialenosti od tejto priamky. Ak je toto splnené, je rozoznaný objekt typu prechod a čiara nakreslená používateľom je nahradená obrázkom prechodu.

Krížik (mazanie) – Mazanie je v našej aplikácii realizované nakreslením krížika, čo pripomína prešKrtnutie na papieri. Aby bolo možné realizovať nakreslenie krížika, je nutné vložiť za nakreslenie krivky a jej rozoznanie oneskorenie, pretože krížik sa skladá v podstate z dvoch úsečiek a nebolo by ho možné nakresliť ak by okamžite po dokreslení čiary bola táto rozoznávaná. Toto však ovplyvňuje aj rozoznávanie objektov typu miesto a prechod, takže aj tu je po nakreslení a rozoznaní resp. prekreslení našim obrázkom určité zdržanie. Krížik je reprezentovaný ako dve úsečky, ktoré sa pretínajú. To znamená, že je nutné rozoznať najprv dve úsečky spôsobom ktorý bol popísaný pri rozoznávaní prechodu a následne treba porovnať ich rovnice priamok a overiť že sa naozaj pretínajú. Ak sú splnené tieto dve podmienky, tj. že ide o 2 úsečky, ktoré sa navyše pretínajú, je tento objekt rozoznaný ako krížik a môže byť vykonaná akcia mazania objektov.

4.3.2 Algoritmy na vyhodnotenie vlastností Petriho sietí

Z hľadiska rozsiahlosti projektu viacerým vlastným algoritmom, sme sa rozhodli prebrať[1] algoritmy na vyhodnotenie vlastností Petriho sieti.

[1] Bc. Ľuboš Heriban – Diagnostika porúch založená na PS modeloch, Diplomová práca

4.3.3 Algoritmus na porovnávanie Petriho sietí

Jeden z najväčších problémov pri vyhodnocovaní otázok, v ktorých sa kreslila Petriho sieť, bolo porovnanie, či sú dve Petriho siete rovnaké. Správny algoritmus sme nenašli, preto sme sa pokúšali zostrojiť vlastný pre naše potrebné výpočty.

Prvým krokom je transformovanie správnej aj študentovej Petriho siete na incidenčnú maticu. Rôznym prehadzovaním riadkom a stĺpcov zisťujeme, či sú tieto siete ekvivalentné. Najprv sa zistí, majú rovnaký počet miest a prechodov (rovnaké počty riadkov a stĺpcov) a potom sa urobia ich súčty, podľa ktorých sa pokúsi nájsť prislúchajúce riadky k riadkom a stĺpce k stĺpcom.

Príslušnosť miesta k miestu (v oboch maticiach) si stále pamätá, ale zároveň si označí tie, ktoré nie je schopný spárovať. Po dokončení zoberie označené miesta (pokiaľ také sú) a riadky, ktoré tieto miesta obsahujú začne prehadzovať tak, že sa pokúša hľadať rovnaké stĺpce medzi maticami. Ak sa mu to nepodarí, tak skúša ďalej, ale ak áno, tak to znamená, že sú rovnaké.

Nakoľko je zložitosť algoritmu veľmi vysoká vzhľadom na počet vykonávaných operácií, rozhodli sme sa nastaviť určité obmedzenie, do ktorého keď to nestihne vyhodnotiť, tak program prehlási, že je potrebné Petriho sieť opraviť ručne.

4.4 Ohraničenia a zmeny špecifikácie

Aplikácia je vyvíjaná pre Tablet PC, kde využívame najmä dotykovú obrazovku tohto typu PC na kreslenie Petriho sieti. Nakoľko škola nemôže zabezpečiť dostatočný počet týchto PC, takisto študenti si ich nemôžu všetci dovoliť, prispôsobili sme toto kreslenie aj na počítačovú myš. Z tohto hľadiska teda aplikácia funguje aj pre Tablet PC, ale pre obyčajné PC. Ďalšie obmedzenie mohlo vzniknúť používaním rôznych operačných systémov. V škole, aj na Tablet-och sú nainštalované operačné systémy Windows XP a preto sme v tomto prostredí aplikáciu aj implementovali. Skúšali sme ju však aj na Windows Vista a fungovala bez problémov. Pre iné operačné systémy naša aplikácia testovaná nebola, pretože nebol dôvod.

Kvôli jednoduchšej implementácii sme funkcie pre správu používateľov a generovanie testov presunuli z klientskej časti na serverovú.

V niektorých prípadoch sme zmenili aj automatické opravovanie otázok na ručné z viacerých dôvodov. Jedným bolo príliš všeobecné zadanie a teda nevedeli sme správne určiť, či bola daná Petriho sieť nakreslená dobre alebo nie. Druhým dôvodom bol fakt, že algoritmus na porovnávanie a vyhodnocovanie jednotlivých vlastností Petriho sieti trval výpočtovo veľmi dlho a preto so súhlasom používateľa sa daná otázka môže vyhodnotiť dodatočne.

Drobná zmena nastala aj v implementácii objektov. Pre Menovku (Label) sme spravili samostatný objekt zdedený od objektu CPNObject. Objekty, ktoré majú parameter Label obsahujú smerník k tomuto typu objektu. Pred tým sme Menovku mali ako súčasť každého objektu. Výhodou tohto riešenia je možnosť nezávislé meniť pozíciu Menovky od objektu.

4.5 Výber implementačného jazyka a prostredia

Rozhodli sme sa pre implementáciu v jazyku C++ vo vývojovom prostredí Microsoft Visual studio 2008, ktorý sme mali prístupný cez MSDN Academic Alliance na FIIT. Využili sme tiež Feature pack pre Visual studio 2008. Pre toto riešenie sme sa rozhodli na základe našich predošlých skúseností s týmto jazykom a prostredím, sympatií a praxe v tomto jazyku.

Toto riešenie nám umožňuje zefektívnenie práce a tiež využívanie externých tried a knižníc, ktoré taktiež uľahčujú prácu ako napríklad triedy CArchive na ukladanie súborov alebo knižnice CAPICOM pomocou ktorej je realizované šifrovanie.

4.6 Optimalizácia programu

V priebehu celého obdobia vývoja softvéru sme sa snažili optimalizovať všetky časti zdrojových kódov. Našim cieľom bolo dosiahnuť čo najefektívnejšie vykonávanie jednotlivých funkcií a eliminovanie redundantného kódu.

Väčšina algoritmov na rozpoznávanie objektov pri kreslení bola zachovaná z prototypu. Zmena algoritmu bola prevedená len úpravou parametrov, kvôli komfortnej práci s rozhraním.

Počas testovania sme odstraňovali chyby, ktoré narušovali stabilitu programu.

5 Overenie riešenia

Táto kapitola hovorí o testovaní produktu a metodike testovania. U takto rozsiahleho softvérového systému je nutné venovať dostatočne dlhý čas testovania a pokryť testovaním všetky možné situácie, ktoré by behom používania aplikácie mohli nastať. Navyše, keďže naša aplikácia má slúžiť aj na hodnotenie študentov, bolo nutné venovať testovaniu zvláštnu pozornosť a zistiť, či nie je možné aby študenti podvádzali. Bolo potrebné otestovať všetky časti aplikácie, ako napríklad prenášanie súborov, ukladanie, nahrávanie, rozpoznávanie kreslenia a pod.

5.1 Metodika overovania

Testovanie prebiehalo jednak počas vývoja aplikácie po jednotlivých častiach, ale aj po dokončení celého riešenia. Testovanie prebiehalo na počítačoch vybavených rôznymi verziami operačného systému Microsoft Windows a na počítačoch s rôznymi hardvérovými konfiguráciami. Testovalo sa jednak na zapožičanom Tablet PC ale aj na klasických PC. Produkt bol testovaný na systémoch MS Windows XP Home, Professional, Tablet PC edition, MS Windows 2000, MS Windows Vista Business v 32 bit aj 64 bit verzii. Testovanie rozpoznávanie kreslenia prebiehalo hlavne na Tablet PC, keďže na tejto platforme bude rozpoznávanie najviac používané(aj keď je možné používať rozpoznávanie aj na klasickom PC). Ďalšie testy spočívali spustení servera na jednom PC a pripojení a vypracovávaní testov na ďalších 3 PC. Počas testovania sme prišli na nedostatky ktoré boli následne opravené. Ďalším testovaním sme overili stabilitu a bezpečnosť našej aplikácie a jej pripravenosť na reálne použitie.

6 Záver

Všeobecný pokec + čo sme nestihli a čo sme sa naučili. Asi to necháme na Šimona. Ma básnické črevo.

7 Použitá literatúra

[1] Redbird Software Corp. (posledný prístup 12.11.2008) http://www.scormsoft.com/

[2] ARIADNE Foundation for the Knowledge Pool. (posledný prístup 12.11.2008) http://www.ariadne-eu.org/

[3] IMS Global Learning Consortium, Inc. (posledný prístup 22.08.2008) http://www.imsglobal.org/

[4] IEEE Corporate. (posledný prístup 12.11.2008) http://standards.ieee.org/

[5] The Aviation Industry Committee. (posledný prístup 12.11.2008) http://www.aicc.org/

[6] Faculty Use Of Course Management Systems, Glenda Morgan, University of Wisconsin System, Volume 2. 2003. http://net.educause.edu/ir/library/pdf/ers0302/rs/ers0302w.pdf

[7] Ing. Ľudovít Mikuš, doc. Ing. Matilda Drozdová, PhD., Ing. Petr Ivaniga.: Komunikácia znalostí prostredníctvom LMS a LCMS. 2007. http://www.cvtisr.sk/itlib/itlib073/mikus.htm.

[8] Weller, M., Virtual Learning Environments: Using, choosing and developing your VLE - London. Routledge. 2007.

[9] VADKERTIOVÁ, E.: E-learning vo vzdelávacom procese: Katedra inžinierskej pedagogiky a psychológie. Materiálotechnologická fakulta STU. http://www.mtf.stuba.sk/docs//internetovy_casopis/2005/2/vadkertiova.pdf

[10] VALÁŠEK, M. E-learning je pohodlný, ale drahý, TREND. (posledný prístup 22.10.2002). http://www.etrend.sk/podnikanie/riadenie-a-kariera/e-learning-je-pohodlny-ale-drahy/20510.html

[11] CLAROLINE.NET. 2008. (posledný prístup 31.10.2008) http://www.claroline.net/

[12] Moodle. 2008. (posledný prístup 1.11.2008) www.moodle.org

[13] Sakai. 2008. (posledný prístup 1.11.2008)
http://www.sakaiproject.org/portal

[14] Angel Learning. 2008. (posledný prístup 1.11.2008)
http://elearning-india.com/content/view/102/38/

[15] ANGEL Learning - Learning Management Suite for K-12 and Higher Education. 2008. (posledný prístup 1.11.2008)
http://www.angellearning.com/

[16] Desire2Learn Innovative Learning Technology. (posledný prístup 1.11.2008)
http://www.desire2learn.com/

[17] eCollege: eLearning Solutions. 2008. (posledný prístup 1.11.2008)
http://www.ecollege.com/

[18] KYTKA, J.: Multimediálny výučbový modul pre Petriho siete: FIIT STU Bratislava. Bakalársky projekt.

[19] KYTKA, J.: Podpora dištančného vzdelávania v predmete Špecifikačné a opisné jazyky: FIIT STU Bratislava. 2006. Diplomová práca.

[20] ŠINKOVIČ, P.: Podpora dištančného vzdelávania v predmete Špecifikačné a opisné jazyky: FIIT STU Bratislava. 2008. Diplomová práca.

[21] Tím č.3 PSS: Podpora vzdelávania v predmete Špecifikačné a opisné jazyky: FIIT STU Bratislava. 2008. Tímový projekt.

[22] Tím č.4 PSS: Podpora vzdelávania v predmete Špecifikačné a opisné jazyky: FIIT STU Bratislava. 2008. Tímový projekt.

[23] ČEŠKA, M.: Petriho sítě, Akademické nakladatelství CERM, Brno, 1994. 94 s. ISBN: 8-085-86735-4

[24] CHOLEVA, M.: Programovanie algoritmov pre supervízorové riadenie v jazyku Java: FIIT STU, 2008. Bakalársky projekt.

[25] ExSpecT - Executable Specification Tool. (posledný prístup 22. apríl 2006)
http://www.exspect.com/

[26] Maria: The Modular Reachability Analyzer. (posledný prístup 29. júl 2005)
http://www.tcs.hut.fi/Software/maria/

[27] IBE PACE: business process management. (posledný prístup 13. február 2006)
http://www.ibepace.com/index_en.html

[28] Netlab in general. (posledný prístup 5. február 2007)
http://www.irt.rwth-aachen.de/fileadmin/IRT/Download/NetLab/HilfeHTML/

[29] Bräunl's Petri Net Java Applet. (posledný prístup 12. apríl 2004) http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/java/Braunl/

[30] Patrice's Petri Net Java Applet (posledný prístup 12. apríl 2004) http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/java/Patrice/

[31] SimPRES - A Simulator for time Petri nets. (posledný prístup 21. február 2002) http://www.ida.liu.se/~luico/SimPRES/index.html

[32] Microsoft Corporation 2008, http://msdn.microsoft.com/en-us/library/ms704849(VS.85).aspx

[33] HRUBŠA, J.: Interaktívny prezentačný systém, máj 2005. Diplomový projekt I.