Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

Podpora vzdelávania v predmete Špecifikačné a opisné jazyky

Akademický rok: 2008/2009
Študijný program: Počítačové systémy a siete
Vedúci projektu: Ing. E. Tomalová

Bc. Richard Varga
Bc. Andrej Zelman
Bc. Peter Huska
Bc. Tomáš Kelemen
Bc. Viktor Mészáros

Obsah

Zadanie

Podpora vzdelávania v predmete Špecifikačné a opisné jazyky

Počet tímov: 2
Vedúci tímov: Ing. K. Jelemenská, PhD., Ing. E. Tomalová

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.

Odporúčaná literatúra:

Jozef Kytka, Bc., Podpora dištančného vzdelávania v predmete Špecifikačné a opisné jazyky, Diplomová práca, FIIT STU Bratislava, december 2006

Peter Polačko, Bc., Podpora dištančného vzdelávania v predmete Špecifikačné a opisné jazyky, Diplomová práca, FIIT STU Bratislava, máj 2007

Izsák Peter, Automatické vyhodnocovanie programov vo VHDL, Záverečný projekt, FEI STU Bratislava, máj 2000

Tím 3PSS, Podpora vzdelávania v predmete Špecifikačné a opisné jazyky, Tímový projekt, FIIT STU Bratislava, máj 2008

Tím 4PSS, Podpora vzdelávania v predmete Špecifikačné a opisné jazyky, Tímový projekt, FIIT STU Bratislava, máj 2008

Úvod

Účelom tohto dokumentu je popísať postup pri tvorení a výsledky tímového projektu, ktorý sa zaoberá problematikou Petriho sietí a podporou výučby tohto tematického okruhu v predmete "Špecifikačné a opisné jazyky" na Fakulte informatiky a informačných technológií.

Motivácia

Predmet Špecifikačné a opisné jazyky (ŠPOJ) tematicky nadväzuje na predchádzajúce predmety Architektúra počítačov a Logické obvody a niektoré predmety v ďalšom štúdiu študijného odboru PSS nadväzujú na tento predmet, preto považujeme vedomosti z tohto predmetu za veľmi dôležité. Tento predmet sa zaoberá základnými prostriedkami pre formálnu špecifikáciu a opis číslicových systémov a venuje sa viacerým tematickým okruhom ako opis systémov v jazyku VHDL alebo SystemC a taktiež metóde na opis správania sa Petriho sietí.

Každý člen nášho tímu absolvoval tento predmet, a preto si myslíme, že dokážeme najlepšie identifikovať oblasti výučby, ktoré je treba zlepšiť. A keďže nám v tomto projekte bola poskytnutá šanca venovať sa tejto problematike, budeme radi ak svojou prácou zlepšíme kvalitu výučby tohto kľúčového predmetu pre budúcich študentov.

Ciele práce

ŠPOJ, podobne ako väčšina predmetov na našej fakulte, je z veľkej časti založený na praktických skúsenostiach a aplikovaní teoretických vedomostí pri riešení rôznych úloh. Na praktických cvičeniach z tohto predmetu sa riešia úlohy s gradujúcou náročnosťou hlavne z oblasti opisu v jazyku VHDL a SystemC. Na druhej strane, pri testovaní sa zisťujú aj praktické znalosti z oblasti Petriho sietí, ktoré študenti nemajú počas cvičení čas dostatočne rozvinúť. Preto považujeme túto oblasť za najviac kritickú a zároveň tú, ktorú by sme v našom projekte radi podporili. Podporu si predstavujeme vo vytvorení aplikácie na modelovanie, simuláciu a analýzu Petriho sietí, ktorá bude schopná vyhodnotiť správnosť študentom namodelovanej siete a tým mu poskytnúť spätnú väzbu na jeho vedomosti. Aplikácia bude použiteľná aj pri testovaní študentov a tým odľahčenie náročného kontrolovania správnosti vyučujúcimi, čo odbúra aj nespravodlivé subjektívne ľudské rozhodovanie.

Štruktúra dokumentu

Tento dokument je logicky štrukturovaný podľa jednotlivých fáz, ktoré môžeme rozdeliť na analýzu, špecifikáciu a návrh.

Analýza

V kapitole sme sa sústredili na analýzu existujúcich riešení, tímové projekty a bakalárske práce z predchádzajúcich rokov. Na základe týchto výsledkov sme sa rozhodli pre ďalšie kroky na našom projekte.

Špecifikácia

Kapitola špecifikácia jednoznačne určuje všetky požiadavky, ktoré by mal náš projekt spĺňať. Venovali sme sa aj téme hardvérových a softvérových ohraničení.

Návrh

Posledná kapitola v tomto semestri bude popisovať hrubý návrh vyvíjaného systému. Návrh je rozdelený na architektúru systému a fyzický model údajov.

Použité skratky

ŠPOJ – Špecifikačné a opisné jazyky (predmet druhého ročníka na FIIT v odbore Počítačové systémy a siete).

PSS – študijný odbor Počítačové systémy a siete

VHDL – (VHSIC Hardware Description Language) programovací opisný jazyk slúžiaci pre opis hardwéru. Používa sa pre návrh a simuláciu digitálnych integrovaných obvodov

VHSIC – (Very-High-Speed Integrated Circuit) veľmi rýchle integrované obvody

PN - Petri Nets Petriho siete

TCP/IP - Balík internetových protokolov (Internet protocol suite) Niekedy sa nazýva sada TCP/IP protokolov podľa dvoch najdôležiteších protokolov, ktoré obsahuje: Transmission Control Protocol (TCP) a Internet Protocol (IP), ktoré boli zároveň aj prvé definované

JRE - The Java Runtime Environment (JRE) poskytuje knižnice, the Java Virtual Machine, a iné komponenty pre beh appletov a aplikácií napísaných v programovacom jazyku Java.

JVM - Java Virtual Machine (JVM) je seria softwerových programov a dátových štruktúr, ktoré používajú model virtuálneho stroja pre realizáciu aplikácií.

JRE – (Java Runtime Environment) – implementácia JVM na spúštanie Java programov

HTML - (HyperText Markup Language; HTML) je značkový jazyk určený na vytváranie webových stránok a iných informácií zobraziteľných vo webovom prehliadači. HTML kladie dôraz skôr na prezentáciu informácií (odseky, fonty, váha písma, tabuľky atď.)

MySQL - slobodný a otvorený viacvláknový, viacužívateľský SQL relačný databázový server. MySQL je podporovaný na viacerých platformách (ako Linux, Windows či Solaris a je implementovaný vo viacerých programovacích jazykoch ako PHP, C++ či Perl. Databázový systém je relačný typu DBMS (database management system). Každá databáza je v MySQL tvorená z jednej alebo z viacerých tabuliek.
PHP - skriptovací jazyk pre tvorbu dynamického webu. Sada týchto skriptov bola vydaná pod názvom "Personal Home Page Tools", skrátene PHP.

PIPE – jednoduchá aplikácia voľne šíriteľná, ktorá umožňuje navrhnúť Petriho sieť a taktiež ju odsimulovať

SQL – (Structured Query Language) – jazyk na manipuláciu s relačnou databázou

JDBC ovládač - JDBC - Java DataBase Connectivity - súbor tried umožňujúcich aplikácii posielať SQL príkazy na systém riadenia bázy dát (SRBD) a získavať výsledky. Tieto triedy a rozhrania sa nachádzajú v balíku java.sql

Analýza

Pri analýze existujúcich riešení sme sa sústredili hlavne na tímové projekty z minulých rokov – konkrétne na projekty tímov číslo 3 a 4 z akademického roku 2007/2008. Tieto zdroje boli uvedené aj v odporúčanej literatúre v zadaní projektu. Okrem týchto dvoch prác sme analyzovali aj bakalárske práce dvoch študentov nášho tímu, keďže majú podobný charakter resp. podobné niektoré črty, ako nami vytváraný projekt. Jedná sa o prácu Richarda Vargu s názvom Multimediálny výučbový modul pre Petriho siete a bakalársky projekt Petra Husku s názvom Programovanie algoritmu pre simuláciu správania sa udalostného systému. V tejto kapitole prinesieme bližší pohľad na túto štvoricu.

Podpora vzdelávania v predmete Špecifikačné a opisné jazyky

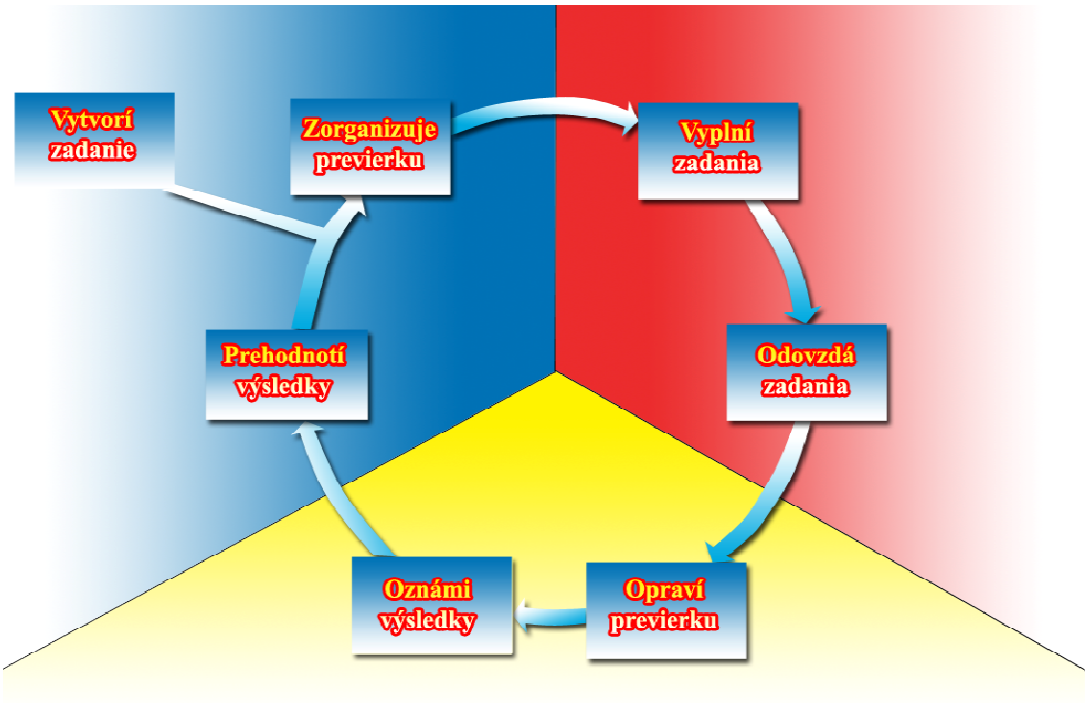
(Tím 4 PSS, Tímový projekt 2008)

Ciele projektu

Autori tohto projektu mali za úlohu vytvoriť výučbový modul pre predmet ŠPOJ, ktorý má slúžiť ako študijný materiál k predmetu, má zabezpečiť správu a vyhodnocovanie zadaní a taktiež overovanie praktických zručností študentov. Výučbový modul má pritom slúžiť predovšetkým na testovanie znalostí z oblasti jazyka VHDL.

Špecifikácia požiadaviek a opis riešeného problému

V dokumente je špecifikovaných viacero požiadaviek na funkciu výsledného systému. Akcie vykonateľné pomocou e-learningového modulu sú znázornené na nasledujúcom obrázku. Obrázok je prevzatý z daného dokumentu a znázorňuje okrem akcií aj role jednotlivých používateľov.



Obr. 1 – akcie v systéme

Na modrom pozadí sú akcie učiteľa, na červenom akcie študenta. Žltá farba značí akcie ktoré vykonáva samotná aplikácia. V reálnom svete by tieto akcie vykonával učiteľ, preto sa tento používateľ (systém) nepovažuje za samostatnú rolu. Autori teda počítajú s dvoma typmi používateľov:

- Učiteľ
- Študent

Na základe analýz mnohých existujúcich e-learningových systémov, sa autori rozhodli aplikáciu realizovať ako rozšírenie pre systém Moodle. Ide teda o webový systém založený na jazyku PHP a HTML. Ako databázový systém bol použitý MySQL.

Návrh, testovanie

V časti návrhu bol navrhnutý fyzický model údajov a jednotlivé metódy, pričom boli použité aj niektoré funkcie systému Moodle. Keďže cieľom projektu bol návrh modulu pre tento systém, bolo potrebné dbať aj na zachovanie súdržnosti medzi novým systémom a Moodlom. Počas overenia funkčnosti boli vykonané testy z pohľadu učiteľa a študenta. Okrem toho sa kontrolovala aj správnosť zobrazenia v jednotlivých prehliadačoch.

Záver

Analyzovaná práca má podobný charakter ako nami vytváraná. Obe majú za úlohu tvorbu výučbového systému, ktorý je zameraný na vyhodnocovanie zadaní a testovanie zručností študentov. Analyzovaný tímový projekt je však zameraný na jazyk VHDL, ktorý je textového charakteru, pričom cieľ našej práce – Petriho siete – sú skôr grafického charakteru. Práve tento rozdiel medzi týmito dvoma témami je dôvodom pre nás na použitie iného implementačného prostredia ako PHP a HTML. Pre naše potreby však môže byť vhodná architektúra systému, ktorú navrhli riešitelia. Potrebné akcie, typy používateľov sú dobre rozpracované a môžu byť vhodnou inšpiráciou pre nami vytváraný systém.

Podpora vzdelávania v predmete Špecifikačné a opisné jazyky

(Tím 3 PSS, Tímový projekt 2008)

Ciele projektu

Podobne ako pri predchádzajúcom projekte, aj tu bolo cieľom autorov navrhnúť a implementovať e-learningový systém na podporu vzdelávania v predmete Špecifikačné a opisné jazyky. Tím mal za úlohu analyzovať existujúce riešenia a dostupné e-learningové systémy a na základe týchto vedomostí vytvoriť moduly pre tieto systémy, prípadne externú aplikáciu. Tím mal riešiť okrem iného problém automatického vyhodnocovania kódu VHDL.

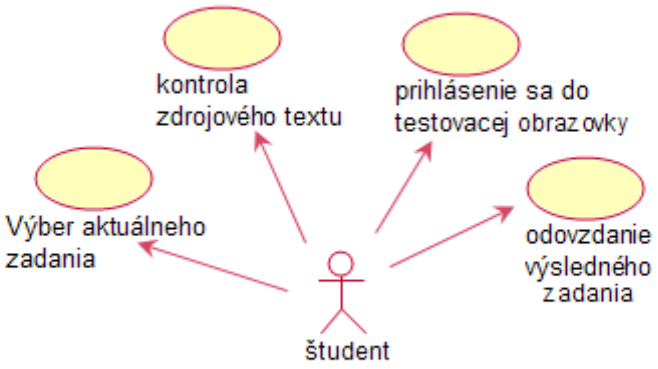
Špecifikácia požiadaviek a opis riešeného problému

Po vyčerpávajúcej analýze existujúcich e-learningových systémov a softvérov na podporu vzdelávania sa tím rozhodol rozdeliť úlohy v zadaní na dve samostatné časti. Keďže v zadaní projektu sa vyžadovalo aj vytvorenie modulov na zabezpečenie učebných materiálov pre študentov a v predchádzajúcich ročníkoch boli študentmi vytvorené moduly pre systém Moodle, autori sa rozhodli realizovať túto časť zadania podobným spôsobom.

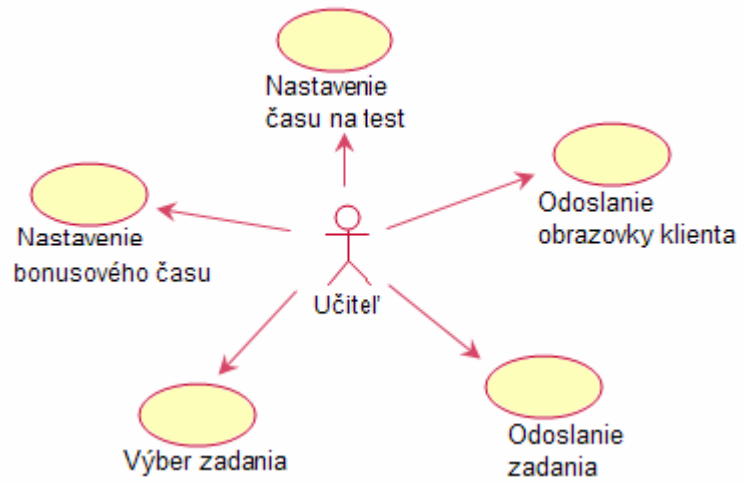
Výučba, správa študentov a testovanie teórie teda riešili pomocou modulu pre Moodle. Keďže systém Moodle je hotový produkt, a tento problém (poskytnutie obsahu, študijných materiálov) bol riešený už v predošlých rokoch, tejto časti sa autori venovali len v časti špecifikácie požiadaviek.

Druhú časť zadania (automatické vyhodnocovanie VHDL kódu) sa tím rozhodol riešiť pomocou aplikácie typu klient-server. Pri tejto aplikácii sa počíta s dvoma typmi používateľov: učitelia a študenti. Operácie vykonávané týmito používateľmi sú rozdelené

podľa klient-server charakteru aplikácie. Učiteľ teda vykonáva operácie na serverovej časti a študent na klientskej strane. Prípady použitia pre obe role sú zobrazené na nasledujúcich obrázkoch.



Obr. 2 - Diagram prípadov použitia pre študenta [zdroj]



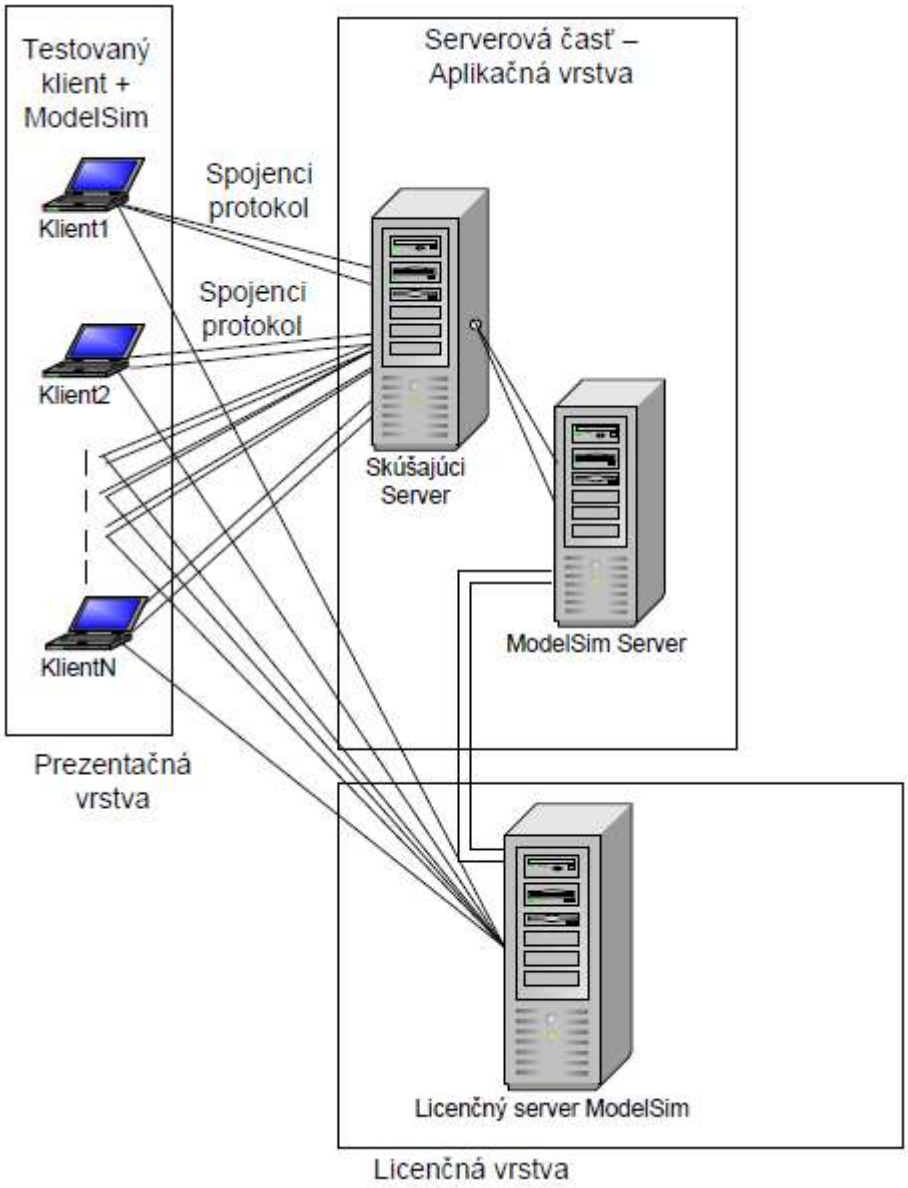
Obr. 3 - Diagram prípadov použitia pre učiteľa [zdroj]

Návrh, implementácia, testovanie

V kapitole „Hrubý návrh“ autori uviedli hlavné princípy testovania zadaní (kontrola syntaxe, simulácia), ako aj funkcie serverovskej a klientskej časti aplikácie. Simulácia kódu sa vykonáva na klientskej časti prostredníctvom nástroja Modelsim.

V časti návrhu je uvedená trojvrstvová architektúra systému. Táto pozostáva z klientskej časti (autori ju nazvali prezentačná vrstva), serverovej časti (aplikačná vrstva) a licenčnej vrstvy. Pre komunikáciu medzi prezentačnou a aplikačnou vrstvou bol použitý

vlastný sieťový protokol, ktorý pracuje na 2. vrstve modelu TCP/IP. Autori ho pomenovali podľa názvu tímu: „Spojenci protokol“. Architektúra systému je znázornená na nasledujúcom obrázku.



Obr. 4 - Architektúra systému [zdroj]

Pre uchovávanie údajov sa rozhodli nepoužiť externú databázu, preto sú všetky dáta reprezentované triedami a ukladané vo forme dynamických polí. Ako implementačný jazyk bol použitý C++, pre zabezpečenie sieťovej komunikácie bola použitá knižnica WinPcap vo verzii 4.02. Testovanie bolo vykonané na piatich počítačoch (4 klienti, 1 server) pomocou zadania „4-bitová sčítačka“. Pri testovaní objavili autori niekoľko chýb, ktoré následne opravili. Odhliadnuc od týchto chýb, systém fungoval podľa autorov bezproblémovo a je využiteľný v praxi.

Záver

Členovia tímu úspešne navrhli a implementovali systém na automatické vyhodnotenie kódu VHDL, ktorý môže slúžiť na testovanie vedomostí študentov. Podľa autorov je systém využiteľný aj na testovanie kódu v System C, táto možnosť však nebola otestovaná. Aplikácia vytvorená tímom má architektúru klient-server a pri tejto komunikácii je požitý vlastný sieťový protokol.

Tento tímový projekt je pre náš tím zaujímavý z mnohých dôvodov. Jedným je použitá klient-server architektúra, ktorá je aj podľa nášho názoru výhodná. Ďalším hodnotným bodom je riešenie samotného prístupu používateľov k jednotlivým funkciám. Naopak, projekt má aj niekoľko nedostatkov, z ktorých by sme sa pri vytváraní našej aplikácie radi poučili.

Programovanie algoritmu pre simuláciu správania sa udalostného systému

(bakalárska práca, Bc. Peter Huska)

Ciele projektu

Ciele projektu bakalárskej práce môžeme rozdeliť do viacerých kategórií. Ako prvé bolo potrebné analyzovať existujúcu prácu, rozobrať použité kódy a detaily použitých knižníc a zistiť použiteľnosť jednotlivých komponentov. Ako druhé bolo potrebné vymyslieť algoritmus, ktorý by vedel spraviť simuláciu jedného prechodu Petriho siete tak aby tento výsledok uložil do určitej matice, aby sme mohli s týmto výsledkom simulácie pokračovať v simulácií . Tento algoritmus sa následne aplikoval na maticu až pokiaľ sa Petriho sieť nedostala do Deadlocku alebo nebola simulácia zastavená používateľom. Algoritmus by mal byť schopný odsimulovať Petriho sieť krokovo, t.j. používateľ si spúšťa ľubovoľné spustiteľné prechody. Ďalej by mal algoritmus vedieť zbehnúť sám a to spôsobom náhodného výberu prechodu. Táto súčasť algoritmu nazvaná náhodný výber prechodu nebola implementovaná do bakalárskeho projektu. Treťou časťou projektu bol výpis priebehu simulácie, ktorý obsahoval číslo spusteného prechodu, názov každého miesta a k nemu priradené značkovanie. Do projektu som doplnil ďalší ciel, ktorého zámerom bolo zjednodušiť komplikované riadenie celej aplikácie od návrhu samotnej Petriho siete až po jeho simuláciu.

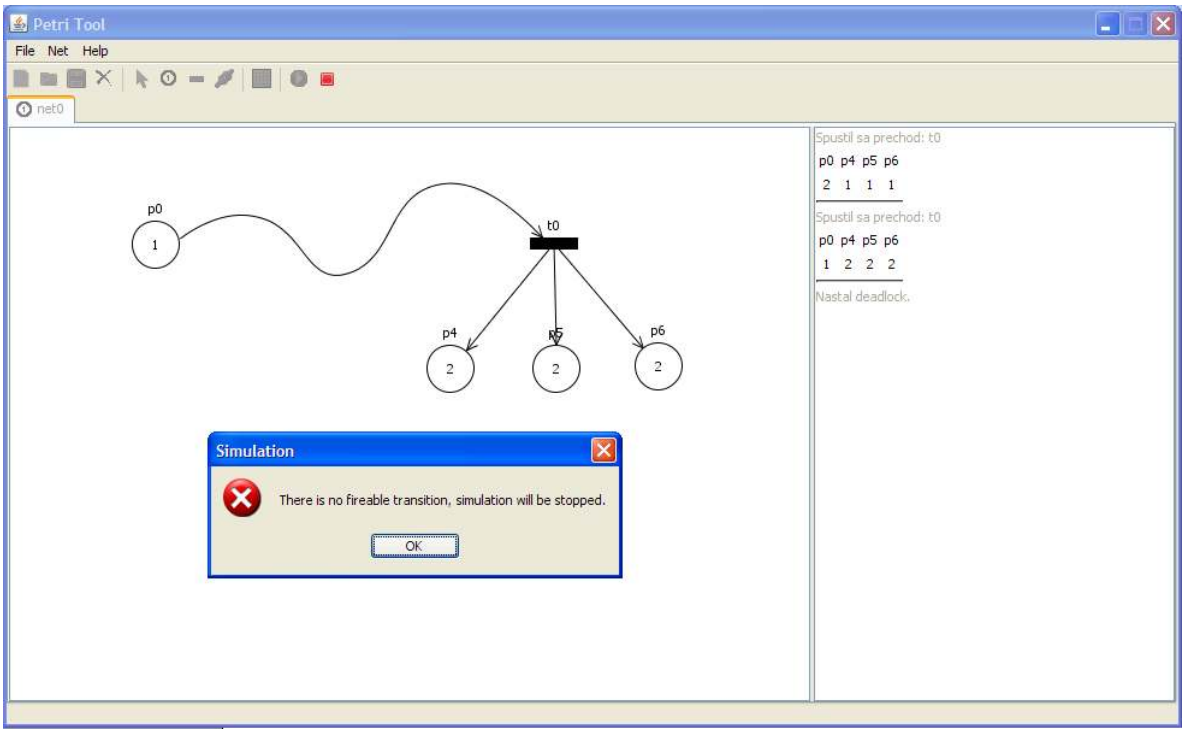
Obr. 4 – prvky ovládania

Analýza požiadaviek

Požiadavky na aplikáciu sú viaceré. V prvom rade ako každá aplikácia by mala byť prehľadná, jednoduchá a mala by mať intuitívne ovládanie. Toto som sa snažil dodržať preto bolo upravované neskôr ovládanie aplikácie. Čo sa týka hardwarových požiadaviek tie sú minimálne. U softwarových požiadaviek je potrebné mať nainštalovanú Javu JRE6.

Návrh a testovanie

Pre overenie správnej činnosti bolo použitých niekoľko príkladov, ktoré dôkladne overili činnosť a správnosť aplikácie. Správnosť simulácie sme overili podľa stromu dosiahnuteľnosti.



Obr. 5 – ukážka aplikácie (deadlock počas simulácie)

Implementácia a testovanie

Autor implementoval aplikácie v programovacom jazyku Java. Pre grafické rozhranie použil knižnice Swing a JGraph. Program bol otestovaný na viacerých Petriho sieťach a v plnom rozsahu spĺňa požiadavky v zadaní.

Záver

Výsledkom celej práce je funkčná aplikácia použiteľná na kreslenie Petriho sietí a simuláciu Petriho sietí. Jadro aplikácie sme sa rozhodli použiť ako základ Tímového projektu s názvom Výučbový modul pre predmet Špecifikačné a opisné jazyky. Pre tento projekt je použiteľná časť v ktorej sa navrhuje Petriho sieť a overuje sa Incidenčná matica.

Multimediálny výučbový modul pre Petriho siete

(bakalárska práca, Bc. Richard Varga)

Analýza

Bakalárska práca Richarda Vargu bola výsledkom práce po dvoch semestroch. Autor si vybral implementačné prostredie Moodle, do ktorého nasadil výučbový modul pre Petriho siete. Tento modul bol rozšírením modulu *spoj.fiit.stuba.sk*, ktorý slúžil ako podklad pre predmet Špecifikačné a opisné jazyky. Modul obsahoval 6 kapitol. Kapitoly boli prehľadné a tematicky rozdelené podľa učebného plánu Petriho sietí pre vyššie spomínaný predmet. Na úvod bola prebratá teória a príklady na jednoduché Petriho siete, neskôr autor rozobral typy a vlastnosti Petriho sietí a na konci sa venoval metódam analýzy. Kapitoly boli doplnené o nové obrázky a interaktívne Flash animácie, pomocou ktorých si študent mohol danú sieť odsimulovať. V module bol zahrnutý aj test, v ktorom si mohol študent otestovať svoje teoretické vedomosti. Súčasťou modulu bol aj program PIPE, voľne dostupný a šíriteľný simulátor Petriho sietí, pomocou ktorého študent mal možnosť vytvoriť vlastnú sieť a túto odsimulovať a analyzovať. Takto si študent rozširoval svoje praktické skúsenosti s modelovaním, simuláciou a analýzou Petriho sietí. Časťou modulu bolo aj diskusné fórum, kde sa o svoje názory mohli študenti podeliť, poprípade sa s nejakým problémom priamo obrátiť na učiteľa.

Záver

Po dôkladnej analýze tejto práce sa tím zhodol na tom, že napriek tomu, že práca je kvalitným materiálom pre výučbu Petriho sietí, v našom projekte sa sústredíme na testovanie teoretických vedomostí a praktických skúseností, takže táto práca nie je vhodným materiálom.

Zhodnotenie analýz

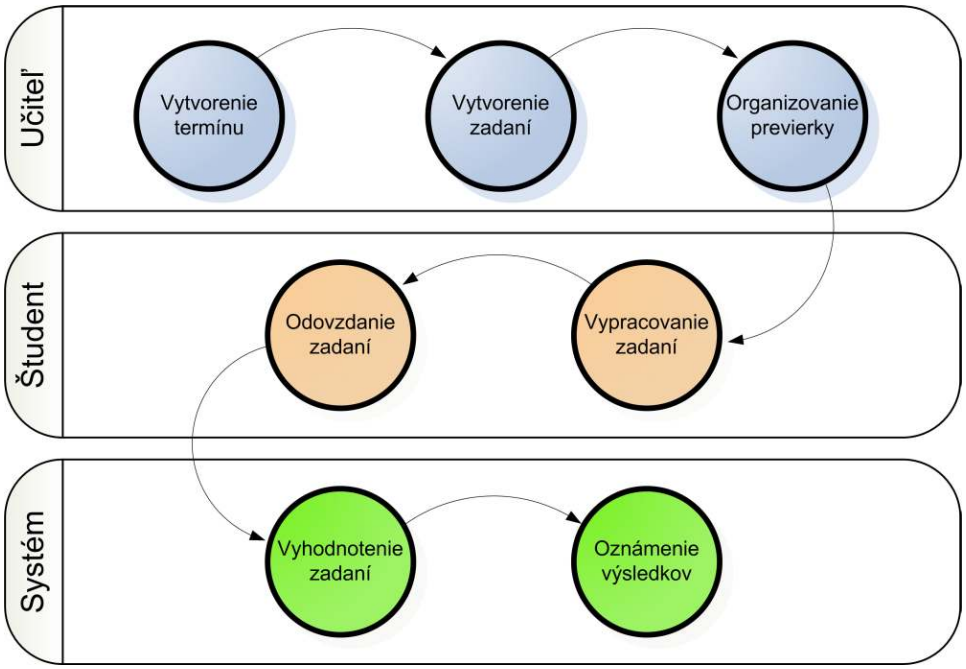
Všetky analyzované projekty sú jedinečné a obsahujú niekoľko vecí, ktorými sa náš tím môže inšpirovať pri riešení projektu. Z nášho pohľadu je ale jednoznačne najhodnotnejšia bakalárska práca Petra Husku, ktorá síce nerieši problematiku e-learningu, ale jeho aplikácia poslúži ako základný stavebný kameň nášho systému. Všetky ostatné práce však obsahujú niekoľko dobrých riešení, ktoré by sme mohli použiť v našom systéme. Samozrejme, žiadna práca nie je bezchybná, preto sme sa pri analýze snažili nájsť tieto chyby a poučiť sa z nich. Pre ďalšie pokračovanie v tímovom projekte budeme vychádzať z práce P. Husku, ktorého aplikáciu rozšírime o komunikáciu s databázovým serverom, generovanie testov, ich riešenie a automatické opravovanie riešení.

Špecifikácia

V nasledujúcej kapitole sme špecifikovali jednotlivé požiadavky, ktoré sú kladené na výsledný produkt projektu z pohľadu poskytovaných funkcií, vstupov, výstupov a grafického používateľského rozhrania. Tieto požiadavky na výsledok projektu vychádzajú zo zadania.

Špecifikácia funkcií systému

Hlavným cieľom nášho projektu je vytvorenie aplikácie, ktorá bude schopná automaticky vyhodnocovať zadania z tematiky Petriho sietí a tým prispeje k podpore vzdelávania v predmete Špecifikačné a opisné jazyky. Našim cieľom je vytvoriť systém ktorí môže byť použitý v reálnej prevádzke, pri zápočtových písomkách alebo skúškach. Preto je dôležité dôkladné overenie teoretických aj praktických znalostí študentov a vyžaduje sa, aby aplikácia bola schopná vyhodnotiť správnosť Petriho sietí navrhnutých skúšanými študentmi, ako aj automatické vyhodnotenie teoretickej časti, ktorá bude realizovaná formou testu.
Aplikácia bude teda podporovať tieto dva typy zadaní, pričom ich určenie ich počtu, maximálneho bodového ohodnotenia, textu zadaní, aj správnych odpovedí je v kompetencii učiteľa. Zjednodušený priebeh testovania a prípravy na testovanie je znázornené na nasledujúcom obrázku.



Obr. 6 – priebeh testovania a prípravy

Jednotlivé činnosti:

1. **Vytvorenie termínu** – učiteľ v systéme zadá časy pre jednotlivé termíny
2. **Vytvorenie zadaní** – učiteľ do systému zadá texty jednotlivých príkladov, správne riešenia a bodové ohodnotenie pre jednotlivé príklady.
3. **Organizovanie previerky** – učiteľ zaradí študentov pre jednotlivé termíny, vyberie príklady pre tieto termíny zo zbierky príkladov a nastaví všetky ostatné parametre testovania.
4. **Vypracovanie zadaní** – študenti dostanú náhodne pridelené príklady so zbierky určenej učiteľom. Na ich vypracovanie majú určený časový limit.
5. **Odovzdanie zadaní** – po vypracovaní všetkých príkladov alebo po uplynutí časového limitu sa zadania odovzdajú.
6. **Vyhodnotenie zadaní** – po odovzdaní zadania sú systémom vyhodnotené
7. **Oznámenie výsledkov** – výsledky previerky sa oznámia jednotlivým študentom a zároveň zapíšu do databázy. Učiteľ si môže pozrieť výsledky študentov a prípadne upraviť bodové hodnotenia.

Funkcie dostupné pre používateľa Učiteľ

Administrácia termínov

Táto funkcia zahŕňa všetky akcie ktoré sa týkajú termínov previerok. Zahŕňa funkcie ako vytvorenie, mazanie a modifikáciu termínov. Na tieto termíny budú následne priraďovaní jednotliví študenti a jednotlivé príklady.

Administrácia študentov

Táto funkcia slúži na vytvorenie kont pre jednotlivých študentov, ich mazanie, úprava, priradenie jednotlivých študentov do už vytvorených termínov, atď.

Administrácia testov

Učiteľ môže pomocou týchto funkcií vytvárať a modifikovať testy. Bude mať možnosť vytvoriť nový test, pridať alebo vymazať príklady, určiť alebo zmeniť text zadania, určiť správne odpovede pri jednotlivých príkladoch, a podobne. Vytvorené testy môže priradiť k ľubovoľnému termínu previerky, alebo určiť množinu príkladov z ktorých budú pre rôznych študentov náhodne vybrané rôzne príklady.

Nastavenie parametrov testovania

Učiteľ bude mať možnosť určiť rôzne parametre pre jednotlivé termíny, ako aj globálne pre všetky termíny. Bude môcť napríklad nastaviť časový limit ktorý budú mať študenti k dispozícii na vypracovanie testu, alebo pri náhodnom priradení testov študentom pomer testov s vyššou a nižšou náročnosťou.

Zobrazenie výsledkov testov a štatistika

Skúšajúci pedagóg bude mať možnosť prehliadať dosiahnuté bodové hodnotenia všetkých študentov, ako aj presné riešenia ktoré boli odovzdané. Systém bude taktiež poskytovať štatistiku v podobe grafov.

Obr. 7 – prípady použitia pre používateľa Učiteľ

Funkcie dostupné pre používateľa Študent

Výber termínu

V prípade že si môže študent vybrať na ktorom termíne previerky sa zúčastní, systém mu po prihlásení ponúkne túto možnosť.

Výber a vypracovanie príkladu

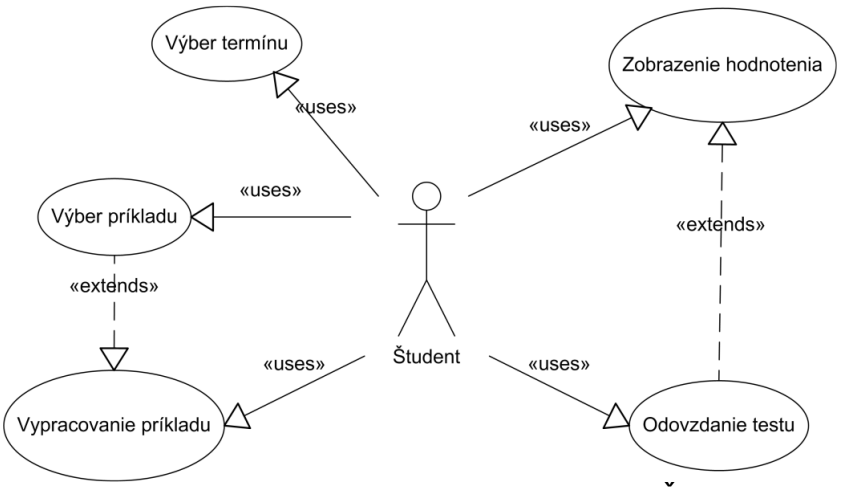
Po začatí testovania systém ponúkne študentovi test na vyplnenie, ktorý sa skladá z viacerých zadaní rôznej obtiažnosti. Každý študent dostane jedno z niekoľkých zadaní patriacich do jednej skupiny obtiažnosti. Takto sa pri viacerých obtiažnostiach zabezpečí približná rozdielnosť testov aby sa zabránilo opisovaniu. Medzi ponúknutými príkladmi bude mať študent počas trvania testu možnosť ľubovoľne prepínať. Môže tak napríklad do polovice vyriešený príklad prepnúť na iný a v jeho riešení pokračovať neskôr.

Odovzdanie testu

Študent má možnosť kedykoľvek odovzdať test. V prípade vypršania časového limitu určeného učiteľom sa test odovzdá automaticky.

Zobrazenie výsledkov a hodnotenia

Po odovzdaní testu a jeho vyhodnotení systémom bude študentovi zobrazené bodové (príp. percentuálne) hodnotenie jeho testu. V prípade že učiteľ pri administrácii testov povolí zobrazenie správnych výsledkov, zobrazia sa aj tieto a študent si môže skontrolovať na ktoré otázky odpovedal nesprávne, príp. kde spravil chybu pri riešení Petriho siete.



Obr. 8 – prípady použitia pre používateľa Študent

Požiadavky a ohraničenia

V tejto časti sú špecifikované hárdverové a softvérové požiadavky a s nimi súvisiace ohraničenia systému.

Hardvérové požiadavky

Tieto požiadavky sú rozdelené na požiadavky na server a požiadavky na klienta.

Na klientskej strane sa vyžaduje počítač, na ktorom spoľahlivo a dostatočne rýchlo (vzhľadom na interakciu) beží Java Virtual Machine (JVM a JRE 6.0). Minimálna odporúčaná konfigurácia:

- CPU Pentium III 800MHz
- RAM 256MB
- 10 MB voľného miesta na disku

Predpokladaná zátaž serveru je 200 používatelov v jednom čase. Preto je potrebný dostatočne výkonný server s väčším množstvom operačnej pamäte. Minimálna odporúčaná konfigurácia:

- CPU 2 GHz
- RAM 2 GB
- 2 GB voľného miesta na disku

Server aj klient musia byť pripojený do tej istej lokálnej siete, prípadne internetu.

Softvérové požiadavky

Na serveri bude bežať MySQL databáza. Na operačný systém nie sú kladené žiadne obmedzenia, okrem nutnosti podpory JVM a JRE. Na strane klienta je potrebný JDBC ovládač na komunikáciu s databázou.

Návrh

Architektúra systému

Kapitola architektúra systému je rozdelená na dve časti, a to architektúra na strane servera a na strane klienta. V architektúre servera je opísaná činnosť na strane servera. V klientskej architektúre sú stručne opísané knižnice, ktoré používa grafické rozhranie, práca s dátami, komunikácia po sieti, pripojenie k databáze a bezpečnosť.

Server

Všetky informácie o danom štúdiu v predmete sú uložené v databáze MySQL. Databázový server je zabezpečený heslom. Jedine administrátor dokáže manipulovať ručne s obsahom databázy. Ostatní používatelia (študent, učiteľ) budú vedieť pracovať s databázou len na tej úrovni, ktorú im poskytne grafické rozhranie a za podmienky, že ich meno sa nachádza v tabuľke učiteľov resp. študentov. Samozrejme každý má svoje konto chránené heslom.

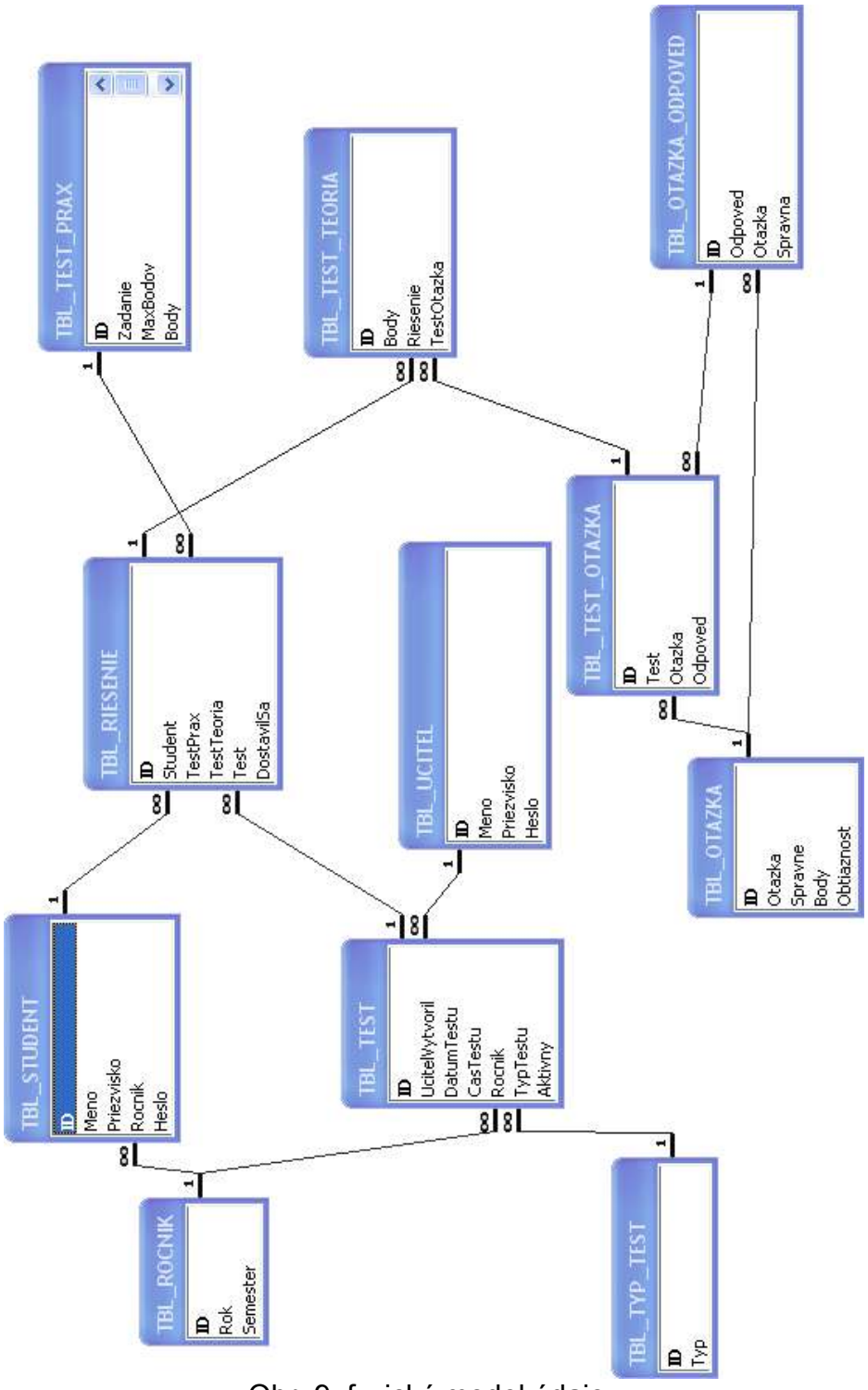
Klient

Klientská aplikácia bude program napísaný v jazyku JAVA. Grafické používateľské rozhranie sa skladá z dvoch častí. Hlavnou časťou je rozšírenie knižnice `JGraph` na také funkcie, ktoré sú potrebné pri modelovaní a simulácii Petriho siete. Ďalšou časťou sú komponenty z knižnice `Swing (javax.swing)`, z ktorých využijeme väčšinou dialógové okná a tabuľky. Modul na komunikáciu s databázovým serverom bude využívať funkcie `JDBC` ovládača (Java DataBase connectivity, knižnica `java.sql`). Medzi modulom na komunikáciu a samotnou aplikáciou bude jeden transparentný modul, ktorý sa bude starať o bezpečnosť. Tento modul bude využívať knižnicu `java.security`. Pre prácu s datami sa budú využívať XML dokumenty. Tieto sa budú spracovávať pomocou DOM (Document Object Model) – knižnica `org.w3c`. Na komunikáciu po sieti sa bude používať knižnica `java.net`.

Výsledná aplikácia bude jeden prenositeľný súbor `.jar`. Program nie je potrebné inštalovať, bude sa dať voľne stiahnuť z webového sídla nášho tímu a bude fungovať na ľubovoľnom počítači, na ktorom je nainštalovaná `JAVA 6 JRE`.

Fyzický model údajov

Databáza pre daný predmet bude obsahovať 11 tabuliek, ktoré sa musia na začiatku semestra naplniť potrebnými dátami. O tento proces sa postará garant predmetu. Postupne sa budú tabuľky dopĺňať systémom počas semestra podľa generovania otázok a zadaní na testoch a ich riešení.



Obr. 9. fyzický model údajov

Tabuľky

Daná kapitola popisuje jednotlivé tabuľky v dátovom modeli a ich obsah. Určuje, ktorý stĺpec v tabuľke čo reprezentuje resp. do akej tabuľky ukazuje.

TBL_STUDENT

Tabuľka bude na začiatku semestra naplnená študentmi, ktorý si daný predmet zapísali. **Meno** a **Priezvisko** sú identifikačné údaje študenta, **Rocnik** určuje ročník, v ktorom študent študuje. **Heslo** je zašifrovaná hodnota hesla pre daného študenta.

TBL_ROCNIK

Na začiatku semestra sa vytvorí záznam v tabuľke o novom roku. Položka **Rok** určuje aktuálny akademický rok, **Semester** určuje či je letný alebo zimný. S touto tabuľkou sú spojené tabuľky TBL_STUDENT a TBL_TEST.

TBL_TYP_TEST

V tejto tabuľke sú uvedené konkrétne typy testov, ktoré je možné študentmi riešiť. **Typ** určuje typ testu.

TBL_TEST

Záznam v tabuľke reprezentuje konkrétny test. **UcitelVytvoril** určuje, ktorý učiteľ vytvoril test, **DatumTestu** je dátum, kedy sa test koná. **CasTestu** určuje čas, koľko trvá daný test. **Rocnik** je ukazovateľom do tabuľky TBL_ROCNIK a určuje pre ktorý ročník je daný test. **TypTestu** určuje aký typ testu sa bude konať. **Aktivny** znamená či je daný test aktívny alebo nie, ak je aktívny, tak študent vie daný test riešiť, ináč nie.

TBL_UCITEL

Na začiatku semestra garant pridá do tabuľky nových učiteľov, resp. odstráni tých, ktorý už nevyučujú tento predmet. **Meno** a **priezvisko** sú identifikačné údaje učiteľa, **Heslo** je šifrovaná hodnota hesla učiteľa, ktorým sa prihlasuje do systému.

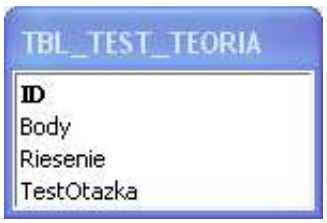
TBL_RIESENIE

Záznam v tabuľke reprezentuje riešenie jedného testu jedným študentom. **Test** určuje konkrétny test (v tabuľke TBL_TEST), ktorého riešením je daný záznam. **Student** je ukazovateľ do tabuľky TBL_STUDENT a určuje, ktorý študent riešil daný test. **DostavilSa** je stavová premenná áno/nie, ktorá určuje, či bol študent na danom teste prítomný alebo nie. **TestPrax** a **TestTeoria** sú ukazovateľmi do pomocnej väzobnej tabuľky TBL_TEST_PRAX a TBL_TEST_TEORIA.

TBL_TEST_PRAX

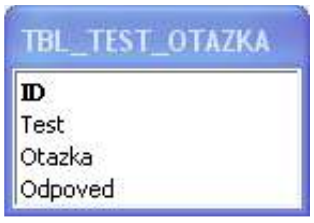
Jeden záznam v tejto tabuľke reprezentuje riešenie daného testu (praktická časť). **Zadanie** je textové znenie zadania praktického testu. **MaxBodov** určuje počet maximálne dosiahnuteľných bodov za daný test. **Body** – počet dosiahnutých bodov. **Riesenie** určuje absolútnu cestu na riešenie daného praktického testu (XML dokument).

TBL_TEST_TEORIA



Záznam reprezentuje riešenie daného testu (teoretická časť). **Body** – počet dosiahnutých bodov. **Riesenie** je ukazovateľom do tabuľky TBL_RIESENIE, ktorá reprezentuje dané riešenie testu daného študenta. **TestOtazka** určuje vzťah s väzobnou tabuľkou, v ktorej sú uložené odpovede a otázky študenta na danom teste.

TBL_TEST_OTAZKA



Jeden záznam v tabuľke určuje odpoveď jedného študenta na jednu otázku. **Test** je ukazovateľ do väzobnej TBL_TEST_TEORIA, ktorá je ďalej spojená s tabuľkou TBL_RIESENIE. **Otazka** určuje na ktorú otázku je dané riešenie. **Odpoved** určuje akú odpoveď si študent zvolil na danú otázku.

TBL_OTAZKA

Na začiatku roka bude táto tabuľka naplnená teoretickými otázkami. **Otazka** je znenie otázky, **Spravne** ukazuje do tabuľky TBL_OTAZKA_ODPOVED a určuje, ktorá odpoveď je správna. **Body** určujú počet bodov, ktoré môže študent za otázku získať. **Obtiaznost** je úroveň zložitosti danej otázky.

TBL_OTAZKA_ODPOVED

Jeden záznam v tabuľke reprezentuje otázku a možnú odpoveď. **Otazka** je ukazovateľ do tabuľky TBL_OTAZKA a určuje, na ktorú otázku je táto odpoveď možná, **Odpoved** je textové znenie odpovede na danú otázku. **Spravna** je stavová premenná áno/nie, určuje či je daná odpoveď na otázku správna.

Komunikácia s databázou

Komunikácia s databázou bude realizovaná pomocou JDBC ovládača (Java DataBase Connectivity). Tento ovládač poskytuje funkcie pripojenia k databáze, odpojenia z databázy, vykonávanie SQL príkazov a získavanie dát z databázy alebo ich modifikáciu.

Implementácia

Nami vyvíjaný projekt na testovanie teoretických vedomostí a praktických zručností študentov v oblasti Petriho sietí bude implementovaný v jazyku Java. Použitá verzia JDK je 1.6.0_11-b03. Na konci zimného semestra sme odovzdali funkčný prototyp. Používateľ vie pomocou myšky modelovať vlastnú sieť a túto následne odsimulovať. V prototype sú implementované 2 funkcie učiteľa, a to: prezeranie zoznamu študentov a prezeranie všetkých teoretických otázok a k nim zodpovedajúcich otázok.

Implementácia prototypu

V tejto kapitole je opísaná implementácia najdôležitejších logických a funkčných blokov, ktoré tvoria aplikáciu.

Reprezentácia Petriho siete

Reprezentácia Petriho siete je určená triedou `PetriNetMatrix`, ktorá v sebe nesie zoznam všetkých miest, prechodov a hrán v danej Petriho sieti. Rozhranie danej triedy nám umožňuje volať metódy na prácu s maticou. Dôoležité sú `add*` a `remove*` metódy. Metódy `addPlace(Object)`, `addTransition(Object)` a `addArc(Object)` slúžia na pridávanie miest, prechodov a hrán do matice. Analogicky metódy `removePlace(Object)`, `removeTransition(Object)` a `removeArc(Object)` sa používajú na odstránenie objektov z matice.

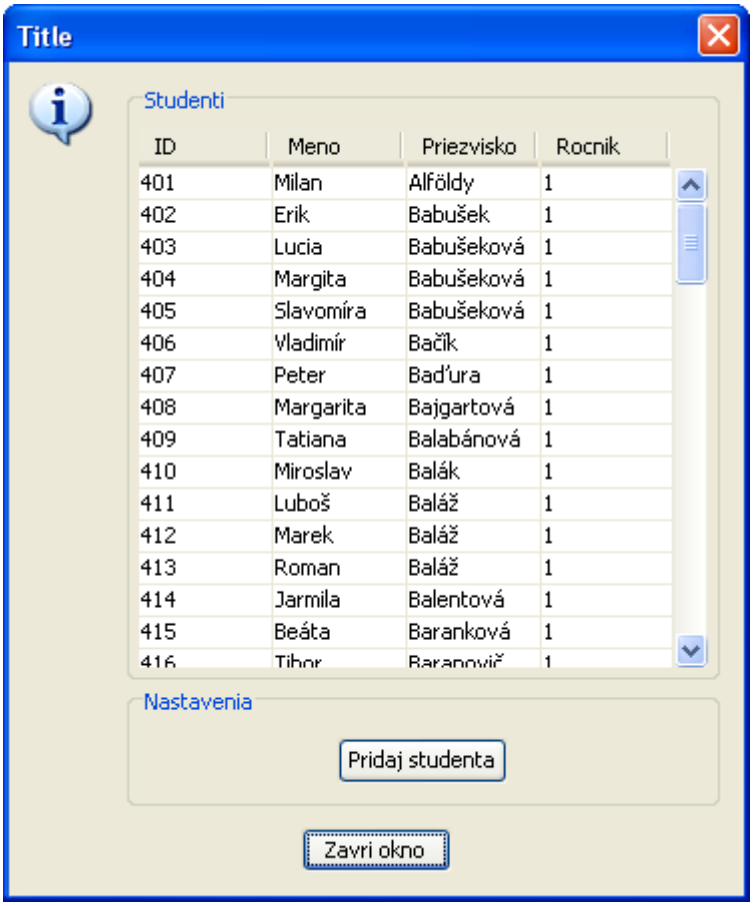
Práca s databázou

Na prácu s databázou sme vytvorili pomocnú statickú triedu `DBHelper`, ktorá nám poskytuje statické metódy na výber prvkov z tabuľky, napr `DBGetStudents()`, `DBGetTests()`, `DBGetQuestions()` a podobne. Ku každej z 11 tabuliek máme takúto metódu. Tieto vracajú typ zoznam objektov, ktoré reprezentujú jeden riadok v danej tabuľke. Každá tabuľka má jeden typ objektu, ktorý dedí od triedy `TBL` (abstraktná trieda, ktorá reprezentuje jeden riadok v tabuľke). Pre efektívnu prácu s databázou sme vytvorili filtrovacie metódy, ktoré na vstupe očakávajú typ long a na výstup posielajú len jeden záznam z tabuľky, ktorého ID je rovné s ID získané na vstupe.

Na korektnú prácu s databázou je potrebné použiť externú knižnicu, ktorá v sebe nesie implementáciu rozhraní a tried `java.sql`. Konkrétne sme použili `mysql-connector-java-3.1.14-bin.jar`.

Grafické rozhranie

Grafické rozhranie je vytvorené pomocou grafických komponentov z knižnice `javax.swing`. Príklad dialógového okna použitého v aplikácii.



Obr. 10. dialógové okno na prezeranie zoznamu študentov

Časový plán

Dátum	Úloha	Stav
29.10	analýza existujúcej literatúry, stanovené ciele projektu, spôsob implementácie	splnené
	práca na dátovom modeli a modeli prípadov použitia	splnené
	vypracovanie kapitoly analýza	splnené
03.11.	dokončený model prípadov použitia a dátový model	splnené
	práca na texte špecifikácie a hrubého návrhu	splnené
13.11.	**odovzdanie dokumentácie analýzy problému, špecifikácie požiadaviek a návrhu riešenia**	splnené
	vypracovanie posudku	-
21.11.	**odovzdanie posudku analýzy, špecifikácie a návrhu iného tímu**	-
	používateľská príručka	-
	príprava príkladov pre prototyp	-
	dokončená implementácia používateľského rozhrania	-
	implementácia prototypu + užívateľská príručka	rozpracované
	testovanie prototypu	-
08.12.	finalizácia dokumentácie a prototypu, formálna aj obsahová	-
15.12.	**používateľská prezentácia prototypu**	-
	vypracovanie posudku na prototyp iného tímu	-
18.12	**odovzdanie posudku prototypu iného tímu**	-

Záver

Tento dokument je opisom tvorby aplikácie na preverovanie teoretických znalostí a praktických skúseností v modelovaní, simulácii a analýze Petriho sietí. Dokument je len prvou časťou výsledného dokumentu, ktorý bude výsledkom aktivít nášho tímu na predmete Tímový projekt. Dokument bude výsledkom práce piatich študentov za dva semestre inžinierskeho štúdia.

V prvej časti dokumentu sme sa venovali analýze dvoch tímových projektov z akademického roka 2007/2008 a dvoch bakalárskych prác t toho istého ak. roka. Na základe týchto analýz sme sa rozhodli ktorým smerom bude pokračovať vývoj našej aplikácie. Výsledkom analýzy bolo rozhodnutie pokračovať vo vývoji aplikácie, ktorú implementoval Peter Huska v rámci svojej bakalárskej práce.

V ďalšej časti dokumentu sme špecifikovali požiadavky na vytváraný systém. Stanovili sme všetky funkcie, ktoré bude vedieť študent a učiteľ využívať a takisto aj softvérové a hardvérové požiadavky na klientský počítač a server.

V poslednej časti tohto dokumentu sme sa navrhli architektúru systému a fyzický dátový model. Je celkom možné, že tento model sa v ďalšom semestri bude meniť.

Dôležité časti dokumentu – implementácia a testovanie – budú spísané priebežne a na konci letného semestra tohto akademického roka bude celý dokument odovzdaný ako výsledok namáhavej práce členov nášho tímu počas dvoch semestrov.

Literatúra

[1] Tím 3PSS, Podpora vzdelávania v predmete Špecifikačné a opisné jazyky, Tímový projekt, FIIT STU Bratislava, máj 2008
[2] Tím 4PSS, Podpora vzdelávania v predmete Špecifikačné a opisné jazyky, Tímový projekt, FIIT STU Bratislava, máj 2008
[3] Bc. Peter Huska, Programovanie algoritmu pre simuláciu správania sa udalostného systému, Bakalárska práca, Bratislava, máj 2008
[4] Bc. Richard Varga, Multimediálny výučbový modul pre Petriho siete, Bakalárska práca, Bratislava, máj 2008