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 najdôležitejších 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

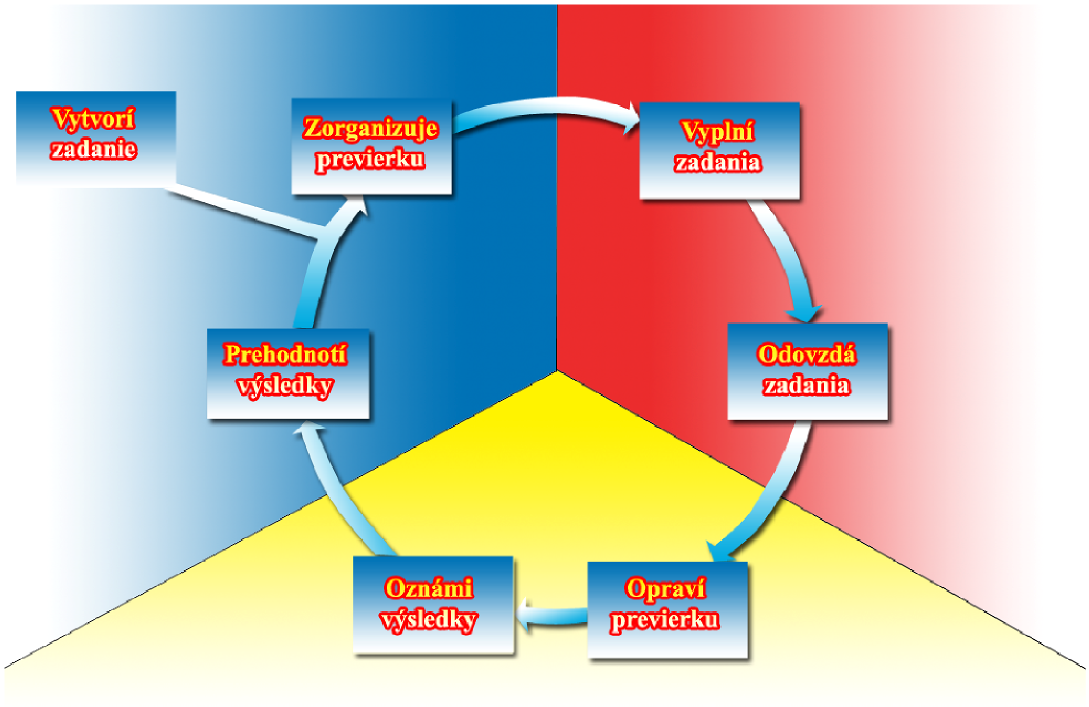
(Tím 4 PSS, Tímový projekt 2008)

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

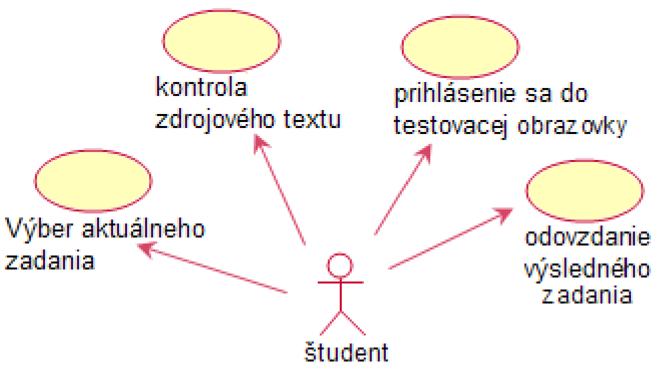
Podobne ako pri predchádzajúcom projekte, aj tu bolo cieľom autorov navrhnúť a implementovať e-learningový systém na podporu vzdelávania v predmete Špecifikačné a opisné jazyky. Tím mal za úlohu analyzovať existujúce riešenia a dostupné e-learningové systémy a na základe týchto vedomostí vytvoriť moduly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

Špecifikácia požiadaviek a opis riešeného problému

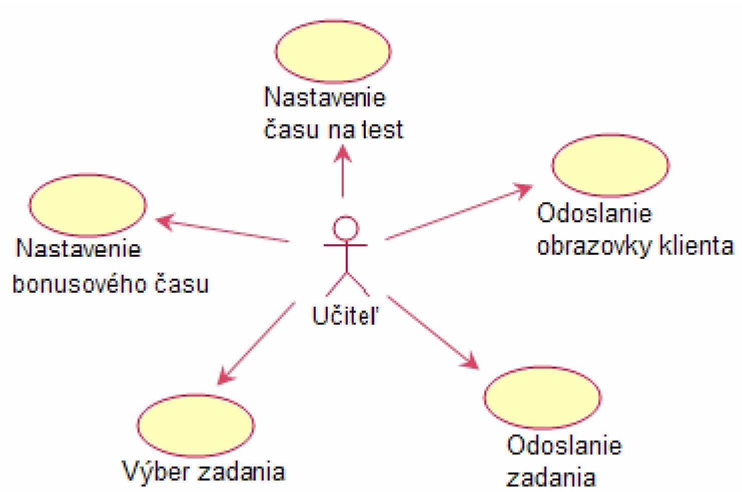
Po vyčerpávajúcej analýze existujúcich e-learningových systémov a softvérov na podporu vzdelávania sa tím rozhodol rozdeliť úlohy v zadaní na dve samostatné časti. Keďže v zadaní projektu sa vyžadovalo aj vytvorenie modulov na zabezpečenie učebných materiálov pre študentov a v predchádzajúcich ročníkoch 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]

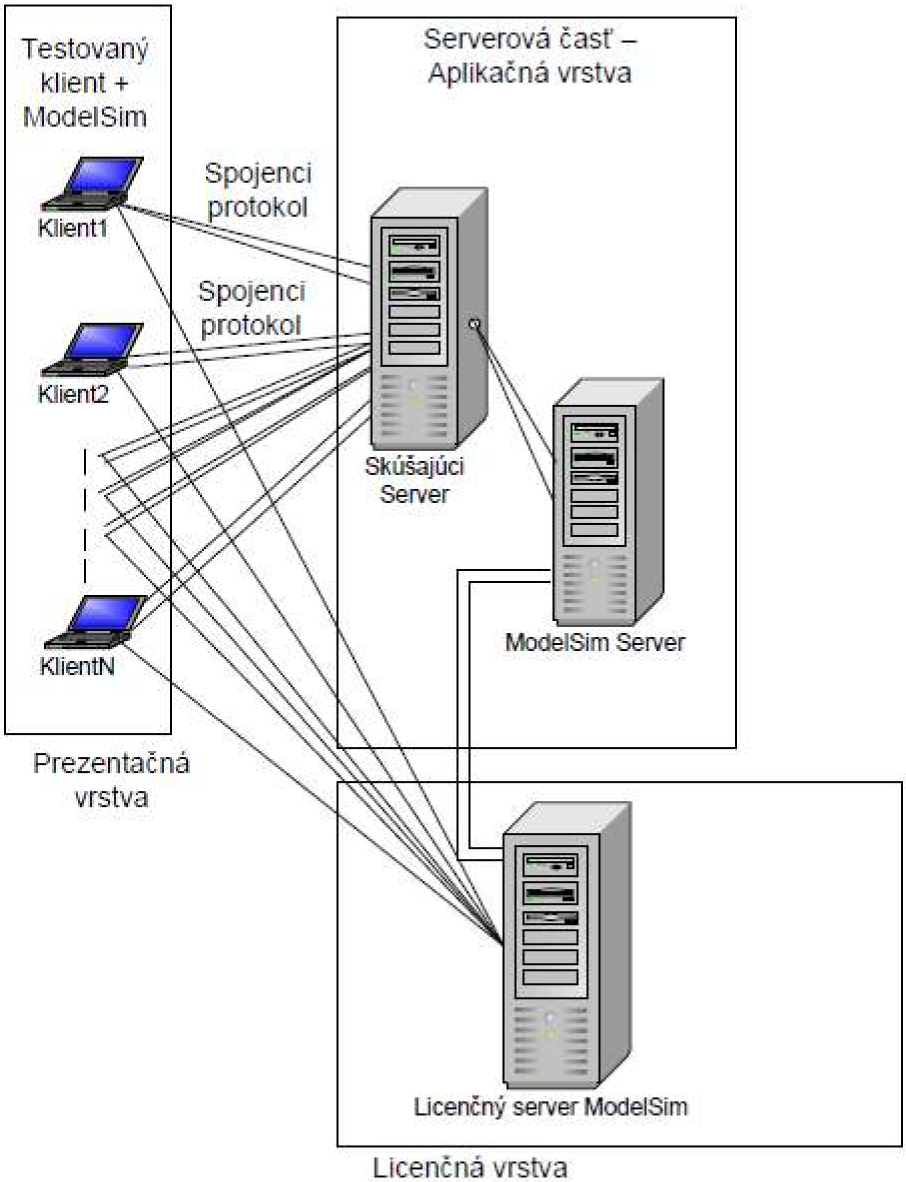


Obr. 3 - Diagram prípadov použitia pre učiteľa [zdroj]

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.

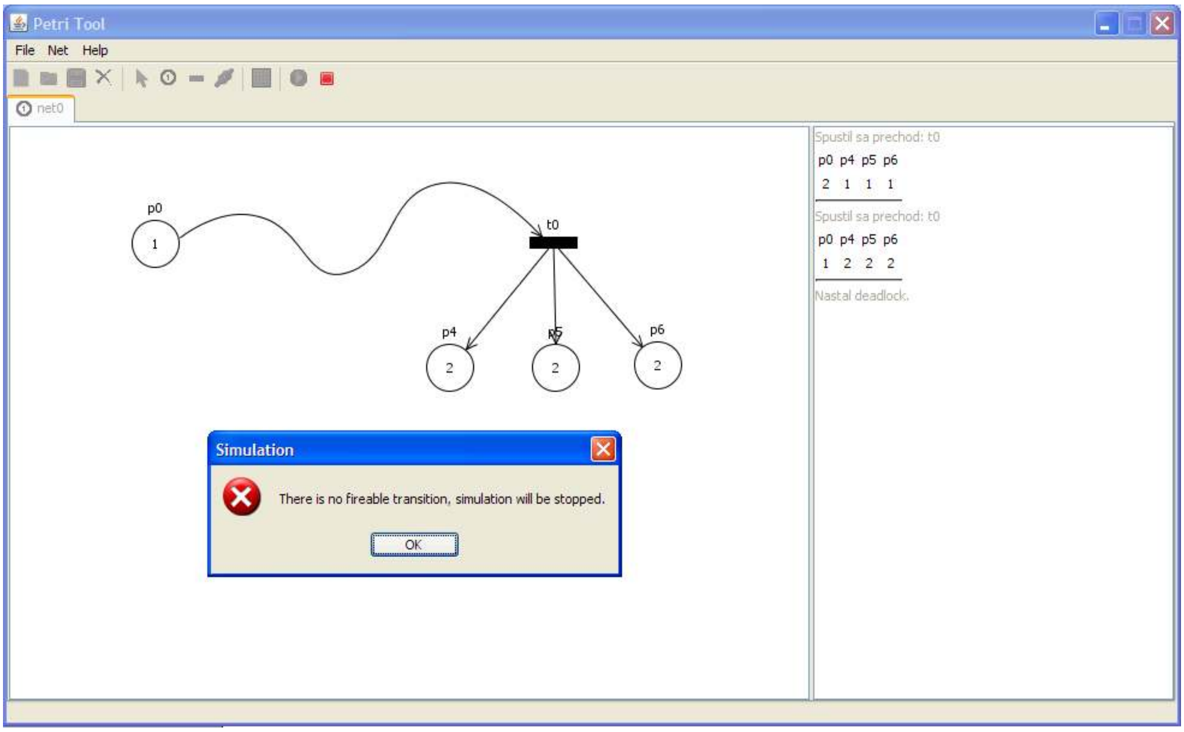
Obr. 5 – prvky ovládania

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. 6 – 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.

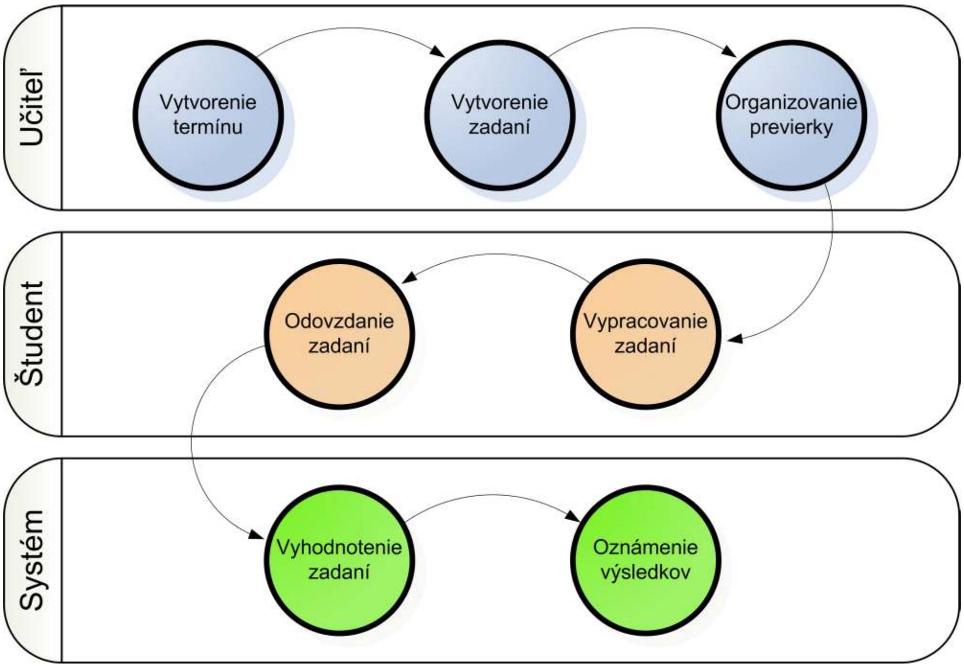
Špecifikácia

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. 7 – 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. 8 – 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

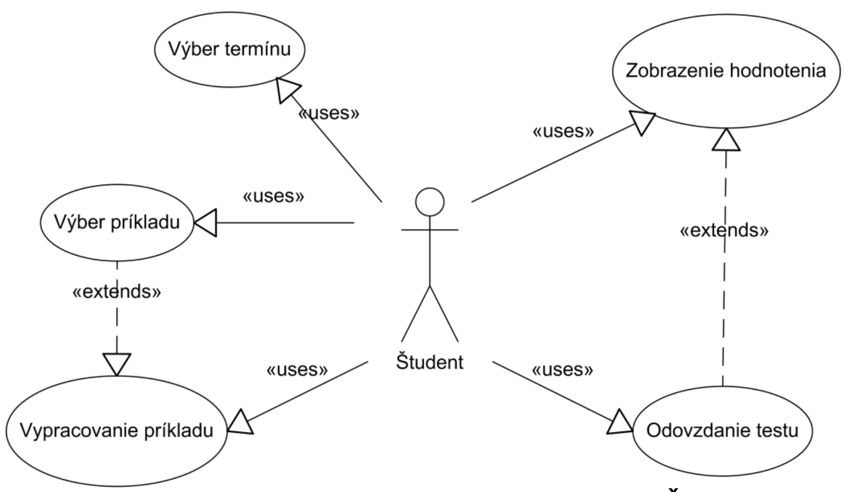
Po začatí testovania systém ponúkne študentovi test na vyplnenie, ktorý sa skladá z viacerých zadaní rôznej obtiažnosti. Každý študent dostane jedno z niekoľkých zadaní patriacich do jednej skupiny obtiažnosti. Takto sa pri viacerých obtiažnostiach zabezpečí približná rozdielnosť testov aby sa zabránilo 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. 9 – 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.

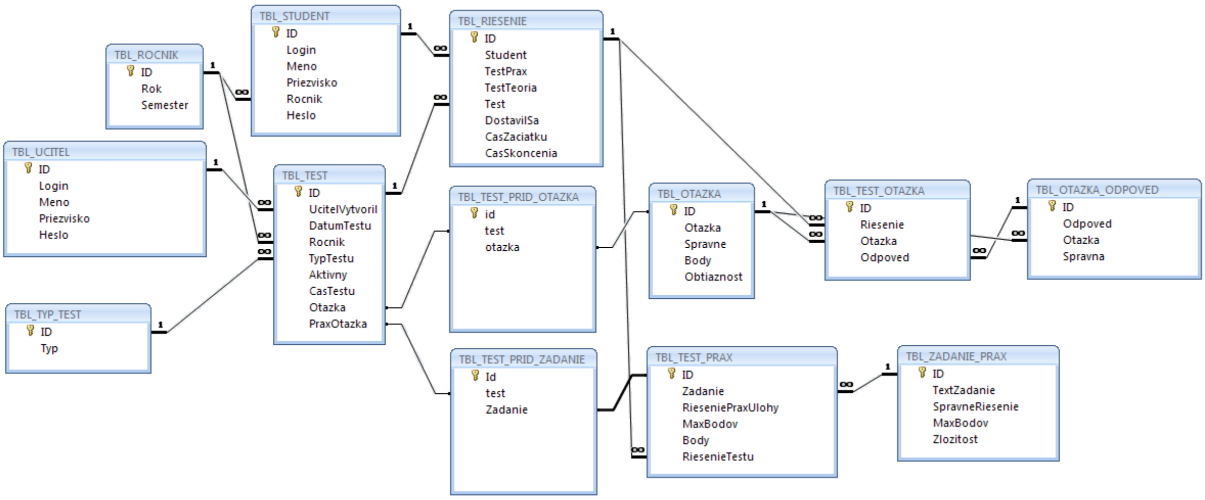
Klient

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, z ktorých niektoré sa musia na začiatku semestra naplniť potrebnými dátami. O tento proces sa postará garant predmetu, alebo niekto, komu túto úlohu garant pridelil, podľa toho sa pridelia potrebné používateľské privilégiá. Postupne sa budú tabuľky dopĺňať systémom počas semestra podľa generovania otázok a zadaní na testoch a ich riešení, ktoré budú vypracovávať študenti.

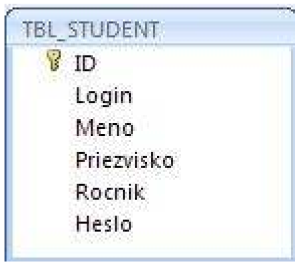


Obr. 10. fyzický model údajov

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. Túto tabuľku bude napĺňať zrejme garant predmetu, alebo osoba tým poverená. **Login** je prihlasovacie meno študenta do systému. **Meno** a **Priezvisko** sú identifikačné údaje študenta, **Rocnik** určuje ročník, v ktorom študent študuje. **Heslo** je zašifrovaná hodnota hesla pre daného študenta.

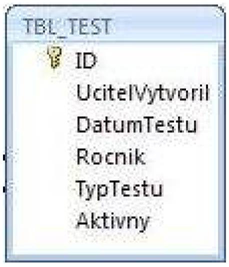
TBL_ROCNIK

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

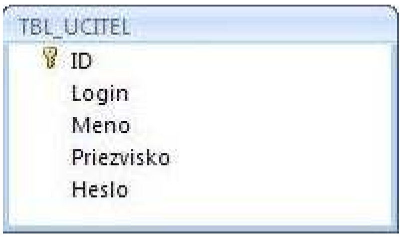
V tejto tabuľke sú uvedené konkrétne typy testov, ktoré je možné študentmi riešiť. **Typ** určuje typ testu (praktický, teoretický).

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 o aký typ testu sa jedná. **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. **Login** reprezentuje prihlasovacie meno, konkrétneho učiteľa. **Meno** a **priezvisko** sú identifikačné údaje učiteľa, **Heslo** je šifrovaná hodnota hesla učiteľa, ktorým sa prihlasuje do systému.

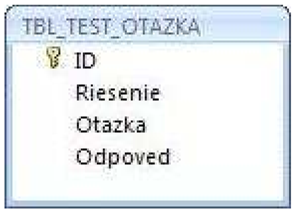
TBL_RIESENIE

Záznam v tabuľke reprezentuje riešenie jedného testu jedným študentom. **Test** určuje konkrétny test (z tabuľky 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. **CasZaciatku** a **CasSkoncenia** sú údaje o čase kedy daný študent začal riešiť resp. odovzdal riešenie v systéme.

TBL_TEST_PRAX

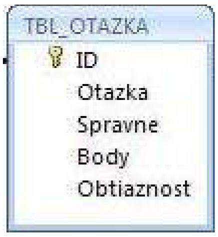
Jeden záznam v tejto tabuľke reprezentuje riešenie daného testu (praktická časť). **Zadanie** je textové znenie zadania praktického testu. **MaxBodov** určuje počet maximálne dosiahnuteľných bodov za daný test. **Body** – počet dosiahnutých bodov. **RieseniePraxUlohy** určuje absolútnu cestu na riešenie daného praktického testu (XML dokument). **RiesenieTestu** určuje, ktorému celkovému riešeniu patrí toto riešenie praktickej úlohy.

TBL_TEST_OTAZKA



Jeden záznam v tabuľke určuje odpoveď jedného študenta na jednu otázku. **Riesenie** je ukazovateľom do tabuľky TBL_RIESENIE, ktorá reprezentuje dané riešenie testu daného študenta. **Otazka** ukazuje do tabuľky TBL_OTAZKA a určuje na ktorú otázku študent odpovedal. **Odpoved** ukazuje do tabuľky TBL_OTAZKA_ODPOVED a určuje akú odpoveď pridelil študent otázke.

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.

TBL_ZADANIE_PRAX

Tabuľka obsahuje praktické zadania na realizáciu petriho sietí. **TextZadanie** slovne opisuje vlastnosti požadovanej petriho siete. **SpravneRiesenie** obsahuje správny model petriho siete. **MaxBodov** určuje bodovú hodnotu zadania. **Zlozitost** určuje stupeň zložitosti zadania

TBL_TEST_PRID_OTAZKA

Tabuľka slúži na priradenie jednotlivých otázok ku konkrétnym testom. **Test** a **Otazka** sú identifikátory v záznamoch tabuliek TBL_TEST resp. TBL_OTAZKA.

TBL_TEST_PRID_ZADANIE

Tabuľka slúži na priradenie jednotlivých zadaní na petriho siete ku konkrétnym testom. **Test** a **Zadanie** sú identifikátory v záznamoch tabuliek TBL_TEST resp. TBL_TEST_PRAX.

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. Dôležité 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.

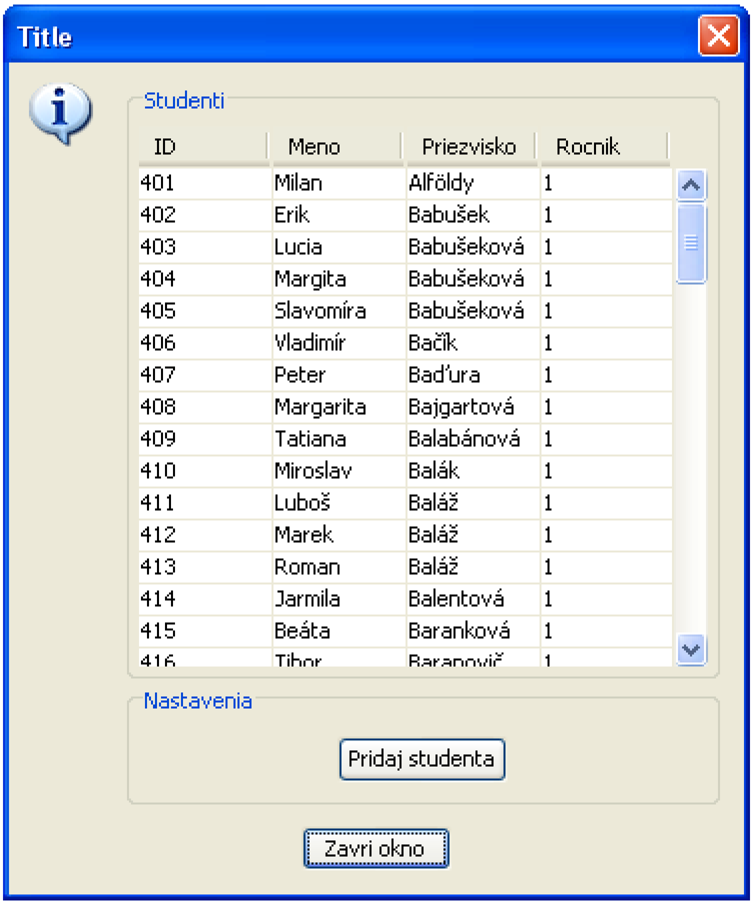
Práca s databázou

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. 11. dialógové okno na prezeranie zoznamu študentov

Implementácia

Simulácia

Miesta, prechody a spojenia medzi nimi sa ukladajú všetky do vlastných polí. Keď užívateľ zvolí simuláciu. Spustí sa metóda *generateMatrix,* v ktorej sa vytvoria matice váh spojení miest a prechodov. Tieto matice sa používajú na zistenie spustiteľnosti prechodov a určovanie značkovaní po spustení prechodov.

Po spustení simulácie a vygenerovaní matíc sa zo zvoleného značkovania, kapacít a váh spojení vypočíta pre každý prechod spustiteľnosť. Ak je prechod spustiteľný zobrazí sa používateľovi červenou farbou. Ak nie je žiadny prechod spustiteľný simulácia na to upozorní používateľa a skončí. O zisťovanie spustiteľnosti prechodov sa stará metóda *isFireable.* Táto metóda sa na začiatku spustí iteratívne pre každý prechod.

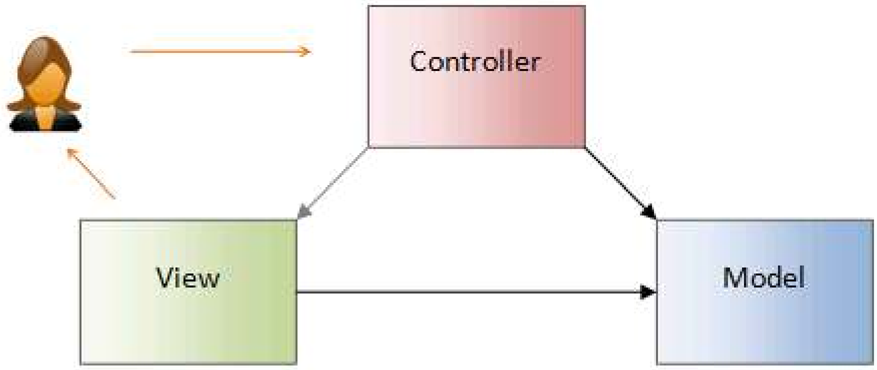
Ak používateľ spustí prechod, zavolá sa metóda *fireTransition.* Táto metóda vypočíta podľa váhových matíc a aktuálneho značkovania nové značkovanie, ktoré sa následne aktualizuje do grafického rozhrania. Keďže po zmene značkovania sa nemusia niektoré prechody dať spustiť, zavolá sa nad všetkými prechodmi znova metóda *isFireable* aby používateľ videl, ktoré prechody sú spustiteľné. Takýmto spôsobom sa tiež zabezpečuje konzistencia siete.

Design aplikácie

Dizajn našej aplikácie je postavený na princípe návrhového vzoru MVC (Model-View-Controll). Je to je ustálený návrhový vzor, ktorý pretvorí aplikáciu na udržateľný, modulárny a rýchlo sa vyvíjajúci balík. Nové vlastnosti sú pridávané jednoducho a staré rysy dostávajú novú tvár. Modulárny a oddelený návrh taktiež umožňuje developerom a návrhárom pracovať súčasne, zahŕňa taktiež možnosť rychlého prototypovania. Separácia taktiež umožňuje developerom robiť zmeny v jednej časti aplikácie bez ovplyvnenia zvyšných častí.

Model je doménovo špecifická reprezentácia informácií, s ktorými sa pracuje
View je pohľad, ktorý predáva dáta reprezentované modelom do podoby vhodnej
k interaktívnej prezentácii používateľovi

Controller alebo radič reaguje na udalosti (typicky pochádzajúce od používateľa) a zaisťuje zmeny v modely alebo v pohľade.



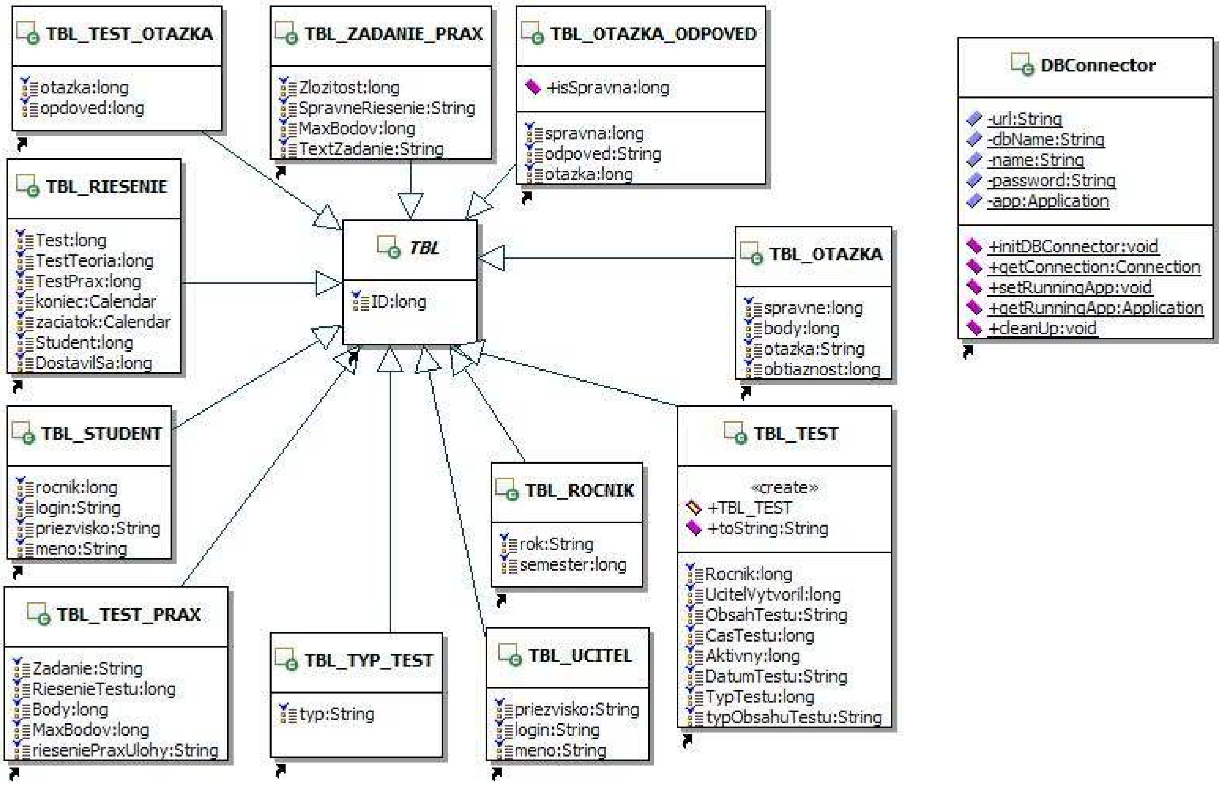
Obr. 14 Model-View-Controller

Implementácia zahŕňa nasledovné kroky:

- vytvorenie databázy
- vytvorenie modelu s validačnými pravidlami
- implementovanie UI so zoznamom a detailmi dát použitím Controllers/Views
- umožnenie CRUD (Create, Update, Delete) operácií
- použitie ViewModel vzoru na posunutie informácie z Controller-a do View

Model

Model predstavuje aplikačnú logiku a údaje. Údajesú síce uložené v databáze, ale na ich používanie je treba mať mapovanie týchto údajov do aplikačnej logiky. Nasledujúci diagram tried znázorňuje implementáciu mapovania tried v našej aplikácii.



Obr. 15 - Model

Trieda TBL predstavuje abstraktnú reprezentáciu záznamu v tabuľke. Pretože každý záznam musí mať ID, je tento atribút v triede TBL.

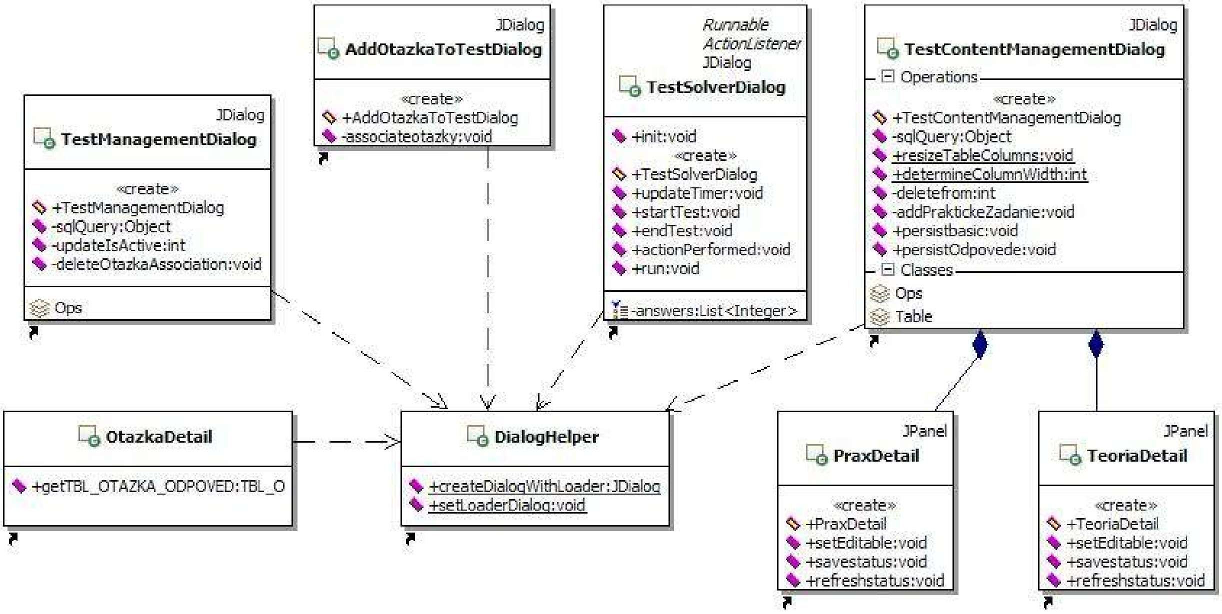
Od tejto triedy dedia všetky ostatné triedy reprezentujúce záznam v tabuľke. Toto mapovanie je vytvorené jedna k jednej- čiže pre každú tabuľku jedna trieda.

Trieda DBConnector zabezpečuje vytvorenie a uchovávanie pripojenia do databáze a základné inicializačné metódy ako je vyvolanie prihlasovacieho formulára pri spustení aplikácie.

View

Pohľad zabezpečuje prezentáciu dát v grafickom používateľskom rozhraní. Nasledujúci diagram tried zobrazuje našu implementáciu. Diagram obsahuje len zjednodušený náhlad pre zlepšenie prehľadnosti.

Obr,O



Obr. 16 - View

Centrálnym bodom je DialogHelper, ktorý zabezpečuje vytvorenie nového dialógového okna s ikonou priebehu načítavania, keby bol databázový server pomalý alebo zahltený a pomaly by odpovedal, používateľ vidí, že aplikácia pracuje správne. Od tejto triedy potom ďalej dedia všetky ostatné dialógové okná.

TestManagementDialog

Je to gui pre manažment testov. Umožňuje vytváranie nových testov a priraďovanie otázok k teoretickým a zadaní k praktickým testom. Umožuje tiež aktivovať a deaktivovať test pre študentov. Poskytuje aj prehľad, koľký a ktorí študenti už test vykonali a umožňuje taktiež zobraziť detail riešení jednotlivých študentov.

TestContentManagementDialog

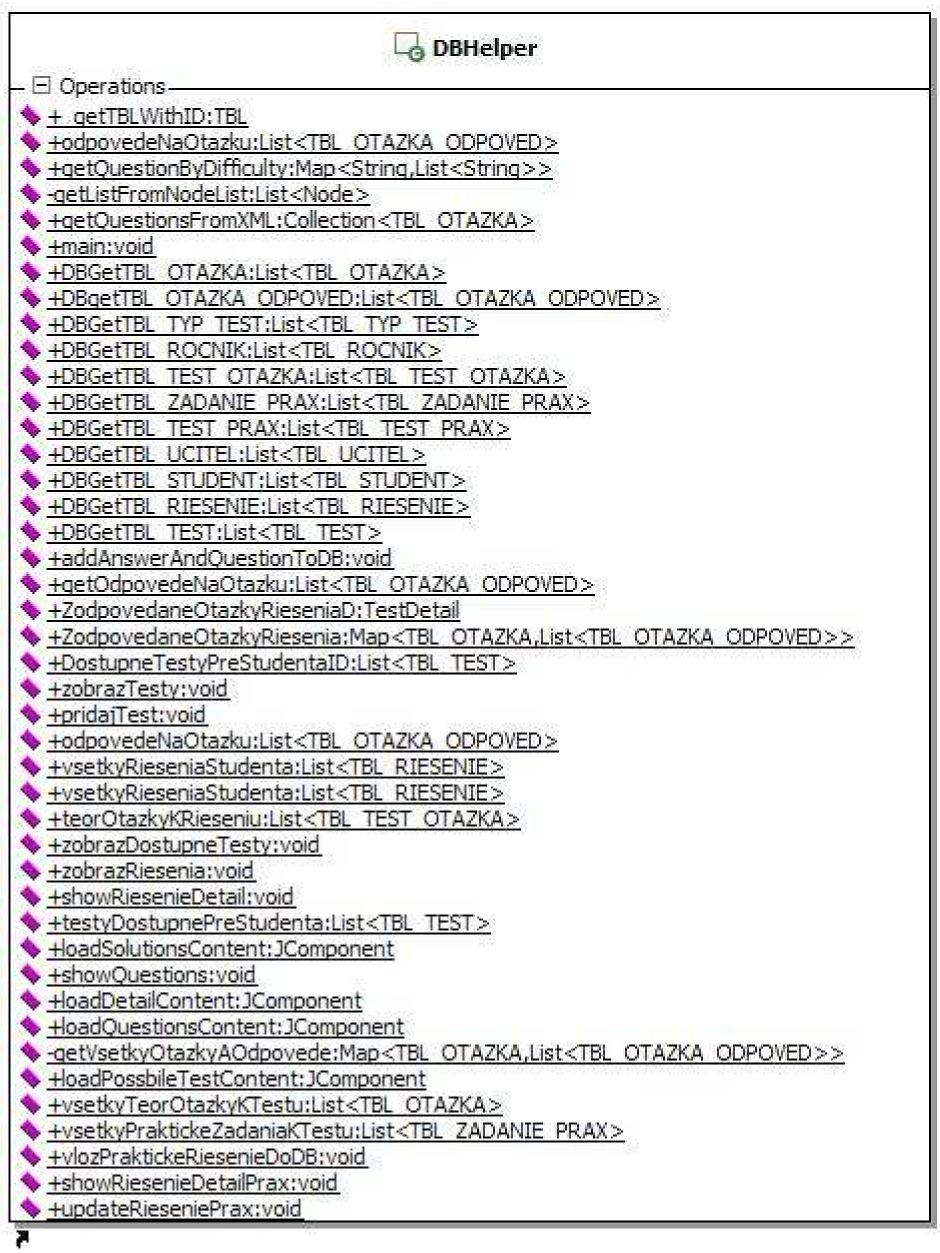
Táto trieda implementuje manažment otázok a zadaní. Umožňuje vytváranie, mazanie a editovanie otázok a zadaní. Taktiež pridávanie, mazanie odpovedí k otázkam a samozrejme aj nastavovanie, ktorá odpoveď je správna. Poskytuje aj jednoduché filtrovanie otázok podľa toho, či sú praktické alebo teoretické.

TestSolverDialog

Toto dialógové okno je na vypracovávanie testov študentmi. Zobrazujú sa v ňom otázky testov a možnosti, ktoré môže študent označiť za správne. Zobrazuje sa mu v tomto okne aj čas ostávajúci na vypracovanie testu.

Control

Majoritná časť riadiacej časti našej aplikácie sa nachádza v triede DBHelper. Táto trieda obsahuje množstvo metód na riadenie grafického interfejsu a taktiež na vykonávanie operácií nad Modelom- čiže databázou. Nasledujúdi diagram znázorňuje obsah tejto triedy.



Obr. 17 - Controller

getTBLwithID:TBL
Metóda slúzi na získanie tabuľky so zadaným identifikátorom. Po získaní tabuľky môžme priamo pristupovať k dátam z tabuľky.

odpovedeNaOtazku:List<TBL_OTAZKA_ODPOVED>
Táto metóda vráti zoznam všetkých odpovedí k otázke zadanej ako vstupný parameter.

getQuestionsByDifficulty:Map<String,List<String>>
Vracia map otázok a obtiažnosti, aby sa ľahko vyberali otázky konkrétnej obtiažnosti.

getQuestionsFromXML:Collection<TBLOTAZKA>
Metóda vyberie z parametrom zadaného XML súboru zoznam všetkých otázok, ktoré sa v XML súbore nachádzajú.

DBGetTBL_OTAZKA
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_OTAZKA.

DBGetTBL_OTAZKA_ODPOVED
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_OTAZKA_ODPOVED.

DBGetTBL_TYP_TEST
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_TYP_TEST.

DBGetTBL_ROCNIK
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_ROCNIK.

DBGetTBL_TEST_OTAZKA
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_TEST_OTAZKA.

DBGetTBL_ZADANIE_PRAX
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_ZADANIE_PRAX.

DBGetTBL_TEST_PRAX
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_TEST_PRAX.

DBGetTBL_UCITEL
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_UCITEL.

DBGetTBL_STUDENT
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_STUDENT.

DBGetTBL_RIESENIE
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_RIESENIE.

DBGetTBL_TEST
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_TEST.

addAnswerAndQuestionToDB
Pomocou tejto metódy sa do databázovej tabulky TBL_OTAZKA a TBL_OTAZKA_ODPOVED pridávajú otázky a odpovede.

getOdpovedeNaOtazku
Metóda vracia odpoveď z tabulky TBL_OTAZKA_ODPOVED na otázku, ktorej identifikátor bol zadaný ako parameter.

ZodpovedaneOtazkyRiesenia
Táto metóda vracia zoznam všetkých odpovedí, ktoré študent označil v rámci zadania. Identifikátor zadania a študenta sa zadávajú ako vstupné parametre.

DostupneTestyPreStudentaID
Metóda vracia všetky testy z tabulky TBL_TEST, ktoré sú pridelené študentovi konkrétneho ročníka a sú aktívne.

zobrazTesty
Metóda vracia zoznam všetkých prvkov priamo z databázovej tabulky TBL_TEST.

pridajTest
Metóda slúži na pridávanie testov do databázovej tabuľky TBL_TEST.

showRiesenieDetail
Metóda vráti podrobné informácie o tom ako študent vypracoval test na základe vstupných identifikátorov študenta a testu.

Testovanie

Testovanie aplikácie sme realizovali posledné 3 týždne pred ukončením semestra. Testovanie prebiehalo v dvoch fázach. Prvá fáza bola funkčná, kde sme testovali funkčnosť aplikácie a druhá bola dátová. V tejto sme testovali správnosť dát.

Funkčné testovanie

Testovanie funkčnosti aplikácie odhalilo veľké množstvo chýb, ktoré sme následne opravili. Medzi tie najzávažnejšie patrili chyby typu nesprávne zobrazovanie okien, zlé časovanie testov a pod.

Dátové testovanie

V tomto testovaní sme robili analýzu obsahu tabuliek po vykonaní jednotlivých operácií s databázou. V tomto testovaní sme odhalili len malé množstvo chýb, napr. zlý formát dátumu začiatku alebo konca testu.

Výsledky testovania

Testovanie nám odhalilo pomerne veľa chý b, čo je pre nás pozitívny výsledok, pretože sme všetky chyby opravili a aplikáciu odladili.

Možné vylepšenia

Aplikácia je funkčná a otestovaná, no nie je dokonalá. Tak ako každá iná, aj táto sa dá orzšíriť o ďalšiu funkcionalitu.

Nápady na vylepšenie

Manažment aplikácie

- Zmena hesla študenta samotným študentom
- Zmena hesla študenta učiteľom (pre zabudnutí)
- Import štdentov z externého súboru
- Import otázok a zadaní z externého súboru

Ovládanie aplikácie

- Ešte jednoduchšie vytváranie Petirho siete (zjednodušiť ovládanie pridávania prechodov)
- Viac klávesových skratiek

Hodnotenie testov

- Pri hodnotení nakreslenej siete by mal učiteľ možnosť pripísať k bodovému hodnoteniu aj tetový komentár, čo bolo v riešení zle alebo dobre
- Filtrovanie praktických testov podľa hodnotených alebo nehodnotených

Inštalácia

Aplikácia je distribuovaná ako JAR súbor. Všetky knižnice sú fyzicky pripojené do tohto súboru. Nie je potrebné žiadne explicité pridávanie knižníc do classpath-u.

Offline mód

Spustenie aplikácie v offline móde sa vykonáva pomocou runOffline.bat súboru, ktorý spustí Java Virtual Machine a následne v nej spustí danú aplikáciu.

Online mód

Spustenie aplikácie v online móde vyžaduje mať k dispozícii MySQL databázu s danou štruktúrou tabuliek. SQL príkaz na vytvorenie prázdnyh tabuliek je uložený v súbore initSQL.sql. Aplikácia v offline móde sa spúšťa so 4 vstupnými parametrami, a to:

- URL k databáze
- Meno databázy
- Meno používateľa
- Heslo používateľa

Po spustení SQL dopytu sa vytvoria všetky potrebné tabuľky. Tabuľka TBL_UCITEL obsahuje len 1 záznam, a to: ADMIN s heslom ‚admin'. Toto heslo je možné po otvorení aplikácie zmeniť.

Používateľská príručka

Táto aplikácia je určená na overenie znalostí študentov predmetu Špecifikačné a opisné jazyky. Grafické rozhranie aplikácie je prispôsobené tak, aby čo najviac uľahčilo prácu vyučujúcemu pri vytváraní a vyhodnocovaní testov. Na druhej strane poskytuje študentom prostriedky na vypracovanie testov nielen teoretických ale aj praktických. Pod teoretickým testom rozumieme otázky s výberom odpovede a pod praktickým testom rozumieme vypracovanie petriho siete podľa daných požiadaviek na vlastnosti petriho sietí. Zo spomenutých faktov je zrejmé logické rozdelenie grafického rozhrania na dve nezávislé skupiny ponúk funkcionalít. Jedna skupina funkcionalít je určená pre vyučujúceho a druhá skupina je určená pre študentov. Najprv si však ukážeme ako sa prihlásime do samotnej aplikácie a následne opíšeme ponuku aplikácie z pohľadu rozdielnych používateľov.

Prihlasovacie menu

Prihlasovacie menu je riešené úplne intuitívne s klasickým rozhraním, ktoré sa nám spustí hneď po naštartovaní aplikácie (obrázok 1). Používateľ vloží svoje prihlasovacie meno a heslo a aplikácia vyhodnotí či sa jedná a študenta alebo učiteľa a podľa toho prispôsobí zobrazené menu. Najskôr je však nutné aby sa používateľ nachádzal v databáze, čo treba zabezpečiť z učiteľovho menu.

Obr. 18 - prihlasovací dialóg

Potom, ako používateľ zadá svoje prihlasovacie údaje odpovie mu aplikácia úspechom alebo zlyhaním prihlasovania (obrázok 2).

Obr. 19 - oznámenie o korektnom prihlásení

Ponuka pre učiteľa

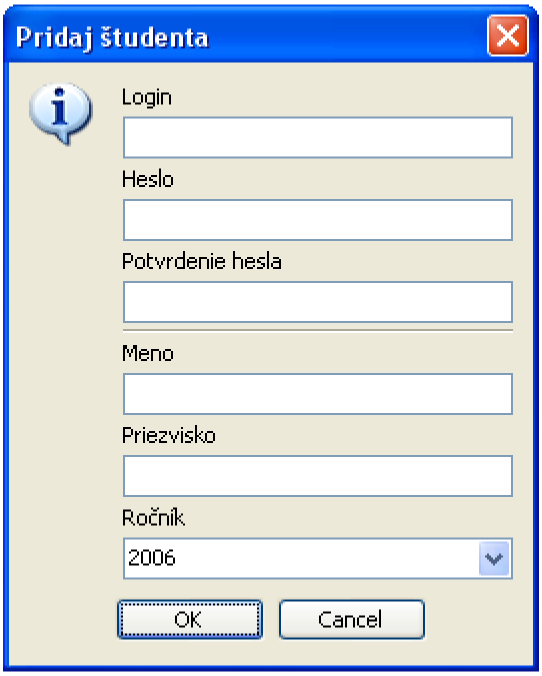
Aplikácia ponúka používateľovi jednoduchú komunikáciu s databázou, a ponúka všetky potrebné funkcionality v ľavej časti okna aplikácie (obrázok 3).

Manažment študentov

V ponuke pre učiteľa je tlačidlo s názvom Manažment študentov. Po stlačení tlačidla sa nám zobrazí dialógové okno. V tomto dialógovom okne je zoznam študentov. každý študent je reprezntovaný ID číslom, loginom, menom priezviskom a jeho ročníkom. ID je generované automaticky. Pri pridávaní študenta do databázy je login ten údaj, ktorý je potrebný pre prihlásenie študenta. Meno a priezvisko a ročník sú údaje, ktoré nie je potrebné vysvetlovať.

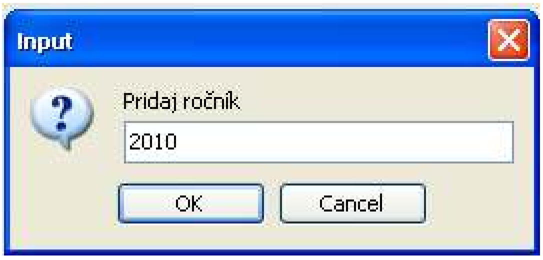
Obr. 20 - Manažment študentov

Pre pridanie študenta je potrebné stlačiť tlačidlo v okne pre manažment študentov. Následovať bude ďalšie dialógové okno, v ktorom vyučujúci pridá do databázy študenta. Pri pridávaní študenta je potrebné zadať aj jeho prihlasovacie údaje ako meno a heslo.



Obr. 21 - pridanie študenta

Pre pridanie ročníku učiteľ musí stlačiť tlačidlo pridať ročník. Následuje jednoduché dialógové okno kde učiteľ dopíše ročník podľa potreby a potvrdí tlačidlom OK. Ak je ročník pridaný, bude zobrazený v ponuke pre pridanie študenta v položke ročník.



Obr. 22 - pridanie ročníku

V prípade, že vyučujúci zadal ročník, ktorý už existuje a je uložený v databáze, bude na túto skutočnosť upozornený výstražným oknom.

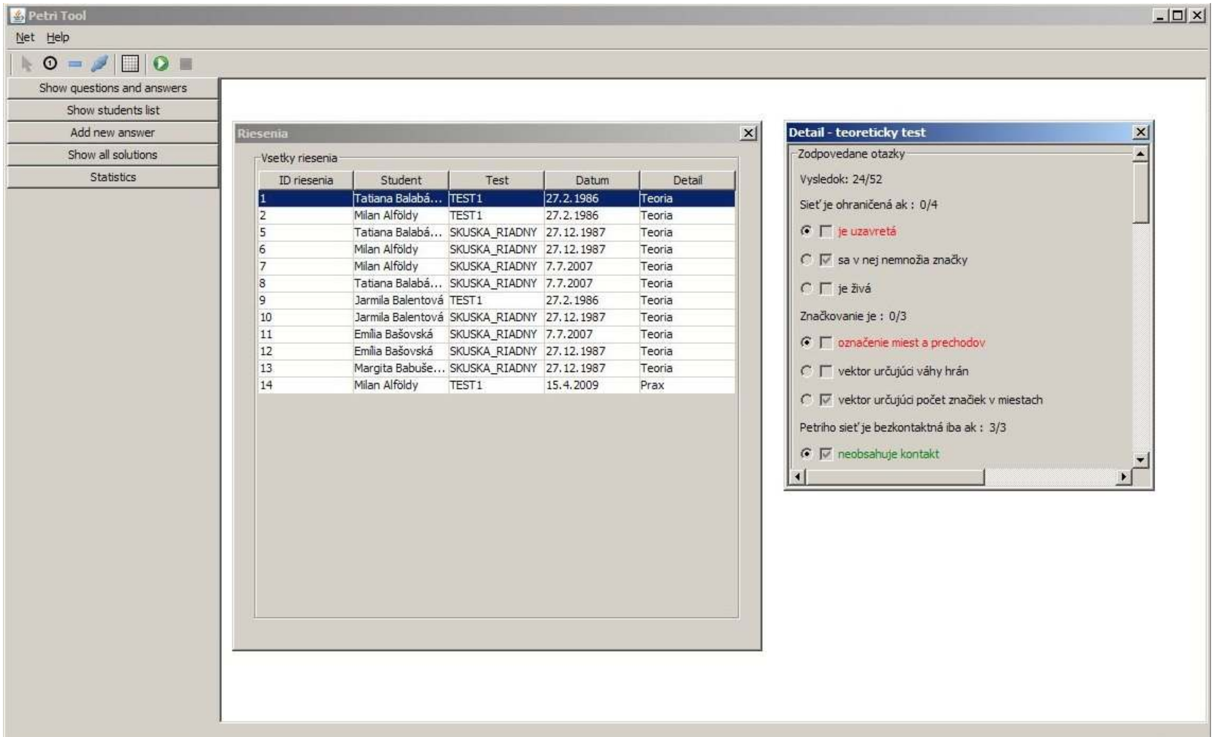
Obr. 23 - výstražné okno

Riešenia

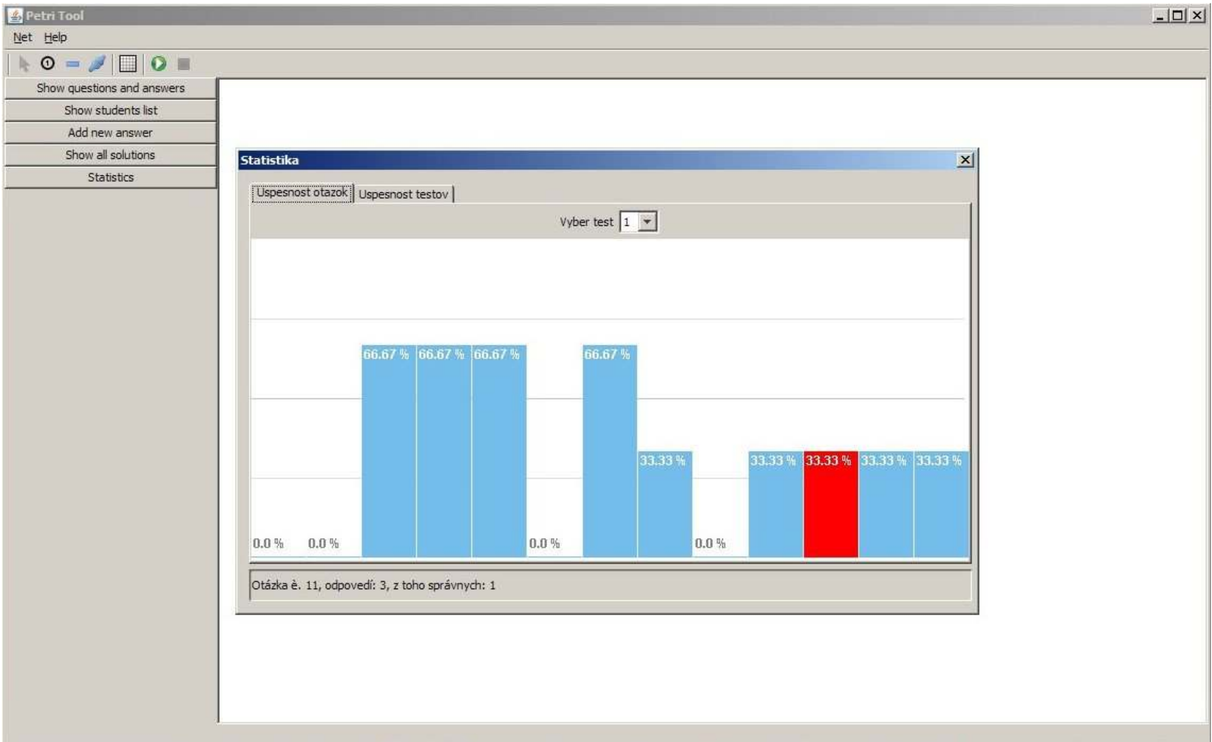
Pod štvrtým tlačidlom ponuky aplikácie („Zobraz všetky riešenia“) sa nachádzaju všetky odovzdané študentmi vypracované testy (obrázok 6). Používateľ si zobrazí test kliknutím na konkrétneho študenta a následne sa zobrazí nové okno so zobrazenými odpoveďami študenta a správnymi odpoveďami. Správne zodpovedané otázky sú zobrazené zelenou farbou a nesprávne červenou. Zároveň je bodkou označený študentov výber odpovede a fajkou je označená správna odpoveď. Pokiaľ je zvolené riešenie k praktickému testu, načíta sa riešenie z databázy, do hlavného panela sa vykreslí sieť, ktorú študent odovzdal a učiteľ ju vie ohodnotiť.

Štatistika

Pod posledným tlačidlom ponuky aplikácie sa ukrýva funkcia na zobrazenie štatistického prehľadu úspešnosti jednotlivých otázok ako aj testov. Po kliknutí na tlačidlo sa zobrazí nové okno s intuitívnym ovládaním (obrázok 7).



Obrázok 24


Petri Tool
Net Help
Show questions and answers
Show students list
Add new answer
Show all solutions
Statistics
Statistika
Uspesnost otazok
Uspesnost testov
Vyber test 1
0.0 %
0.0 %
66.67 %
66.67 %
66.67 %
0.0 %
66.67 %
33.33 %
0.0 %
33.33 %
33.33 %
33.33 %
33.33 %
Otázka č. 11, odpovedí: 3, z toho správnych: 1

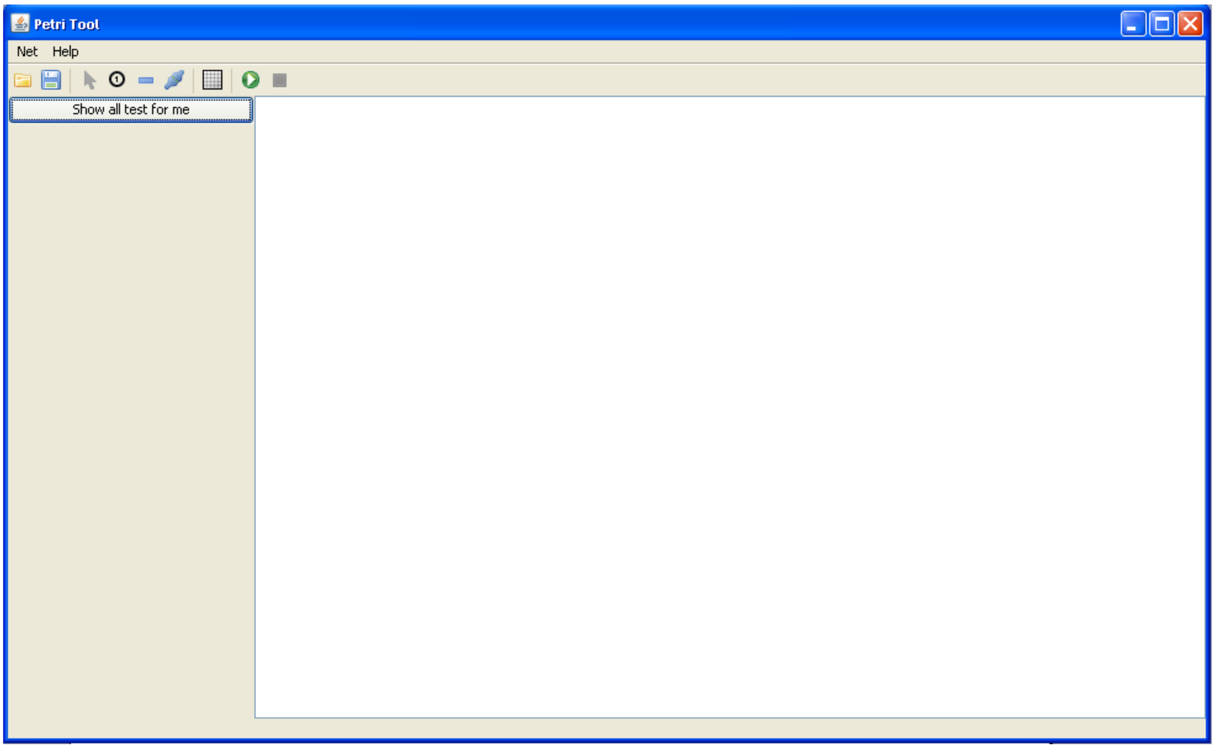
Obrázok 25

Ponuka pre študenta

Po spustení aplikácie sa zobrazí dialogové okno pre prihlásenie. Každý študent má svoje prihlasovacie meno a heslo. Po zadaní a potvrdení systém rozpozná, či sa jedná o učiteľa alebo žiaka.

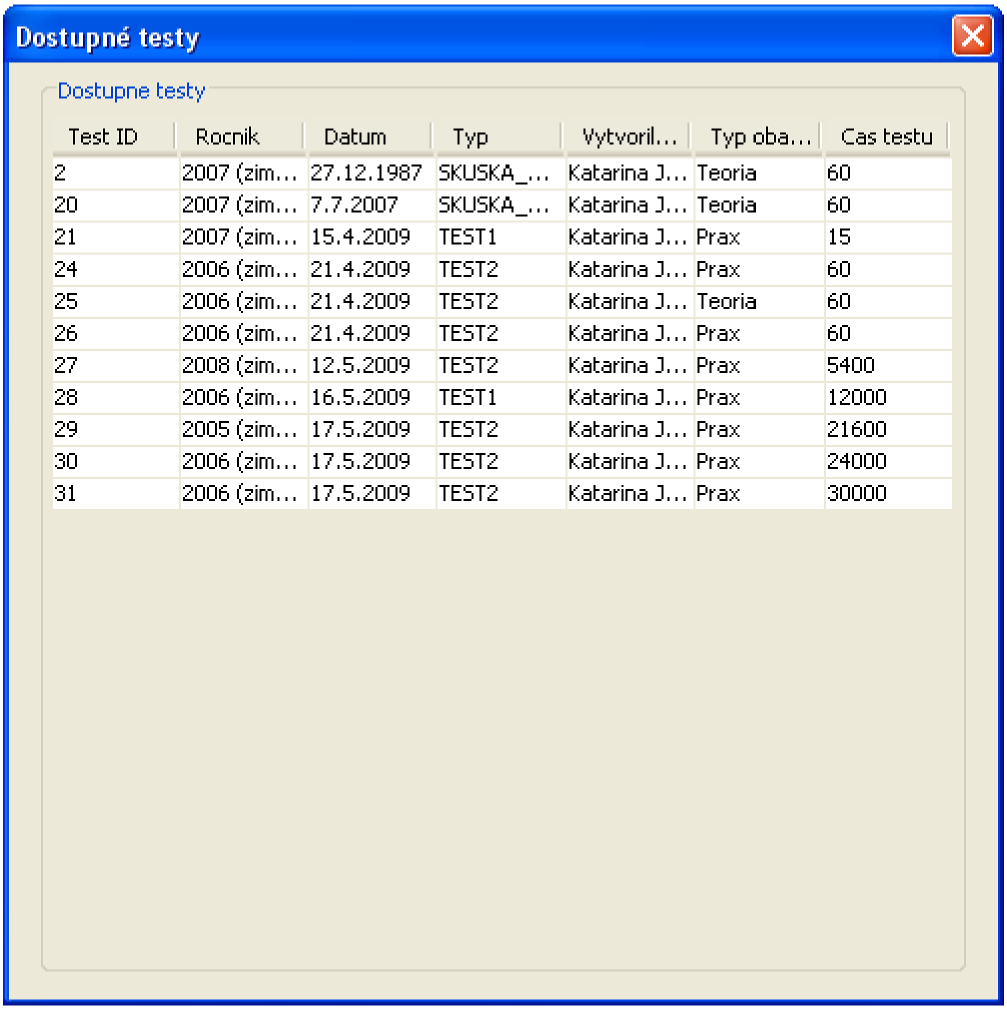
Obr. 26 - prihlasovacie menu

Po prihlásení sa zobrazí úvodná obrazovka. Tu má študent viacej možností. Môže si kresliť a navrhovať Petriho siete (používateľská príručka pre simuláciu). Druhá možnosť je tlačidlo Dostupné testy.



Obr. 27 - okno po prihlásení

Stlačením sa vyvolá dialogové okno, ktoré ponúka všetky testy, ktoré študent môže absolvovať.



Dostupné testy

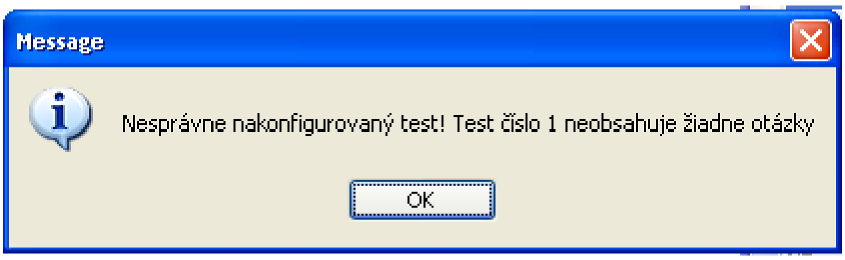
Dostupne testy

Test ID	Rocnik	Datum	Typ	Vytvoril...	Typ oba...	Cas testu
2	2007 (zim...	27.12.1987	SKUSKA_...	Katarina J...	Teoria	60
20	2007 (zim...	7.7.2007	SKUSKA_...	Katarina J...	Teoria	60
21	2007 (zim...	15.4.2009	TEST1	Katarina J...	Prax	15
24	2006 (zim...	21.4.2009	TEST2	Katarina J...	Prax	60
25	2006 (zim...	21.4.2009	TEST2	Katarina J...	Teoria	60
26	2006 (zim...	21.4.2009	TEST2	Katarina J...	Prax	60
27	2008 (zim...	12.5.2009	TEST2	Katarina J...	Prax	5400
28	2006 (zim...	16.5.2009	TEST1	Katarina J...	Prax	12000
29	2005 (zim...	17.5.2009	TEST2	Katarina J...	Prax	21600
30	2006 (zim...	17.5.2009	TEST2	Katarina J...	Prax	24000
31	2006 (zim...	17.5.2009	TEST2	Katarina J...	Prax	30000

Obr. 28 - dostupné testy

Teoretické testovanie

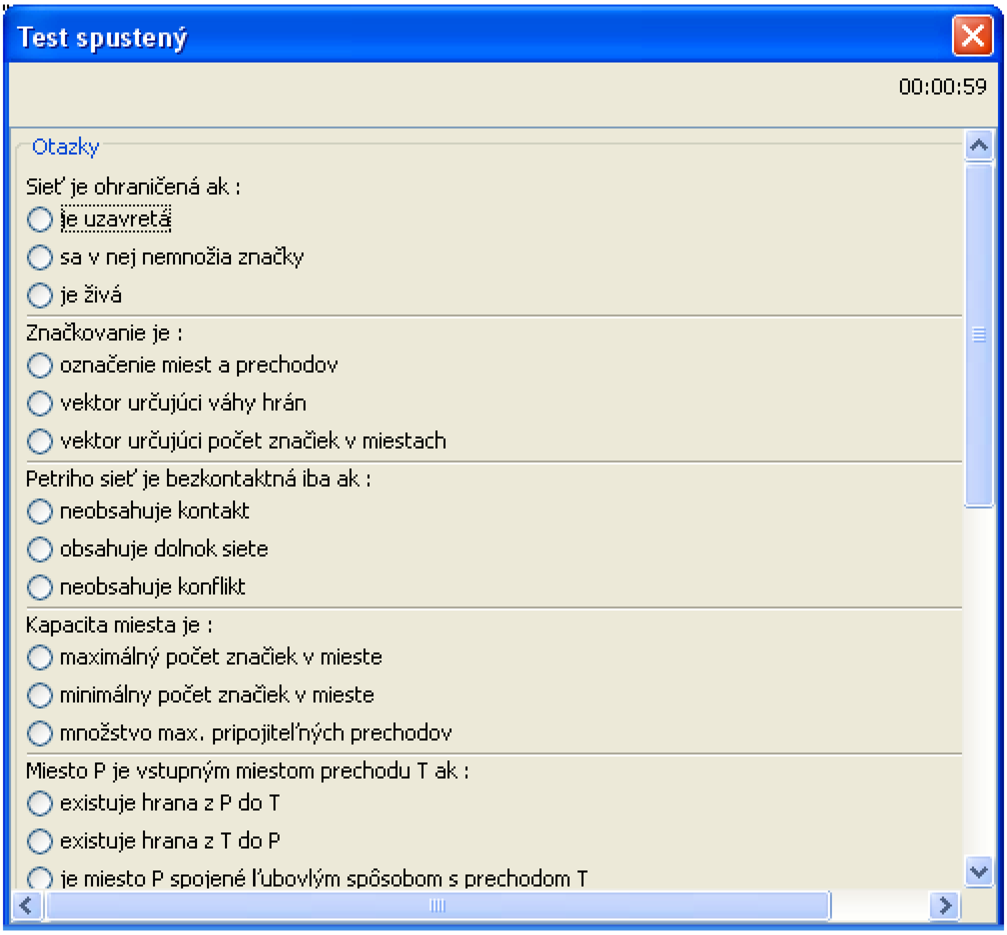
Ak študent chce, môže dvojklikom aktivovať test. Pokiaľ test nie je dostupný, študent o tom bude informovaný príslušným varovným oknom



Obr. 29 - varovné okno neaktívneho testu

V prípade, že test je aktívny, dvojklikom ho študent spustí. Od spustenia testu plynie časový limit do ktorého študent musí vypracovať všetky otázky a test odovzdať. Tento časový limit je dopredu uvedený pri ponuke testov v poslednom stĺpci. Tento údaj je uvedený v sekundách. Pokiaľ študent test po spustení neodovzdá, test bude odovzdaný automaticky. Po odovzdaní testu bude študent informovaný o dosiahnutých výsledkoch dialogovým oknom.

Obr. 30 - informácia o výsledku testu

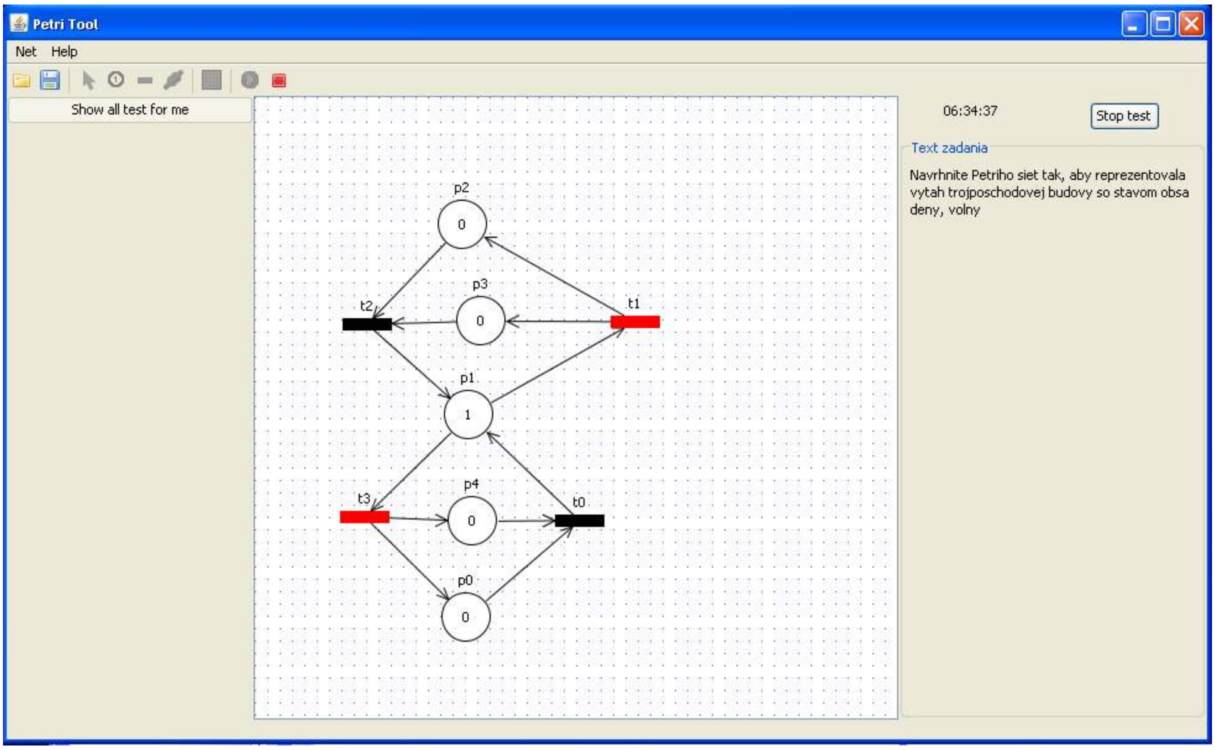


Obr. 31 - test po vypracovaní a odovzdaní

Pokiaľ študent vypracoval a odovzdal test, tento je z ponuky dostupných testov odstránený a nie je možné ho absolvovať znovu. Testy sú viacero kategorií. Teoretický test pozostáva z otázok a ku každej otázke je niekoľko odpovedí.

Praktické testovanie

Pre vypracovanie praktického testu sa prepne systém do módu kde sa kreslia Petriho siete. Tieto praktické zadania nie je možné opravoviť a vyhodnotiť okamžite, pretože správnych riešení môže byť viac. Preto sa tieto zadania iba vypracujú a vyučujúci si ich dodatočne pozre a vyhodnotí na základe vlastného uváženia.



Obr. 32 - príklad praktického testu

Text zadania sa nachádza v pravej strane okna. Pre overenie je možné si spustiť simuláciu Petriho siete tak aby sme si mohli overiť funkčnosť pred odovzdaním testu. Na obrázku je príklad vypracovaného zadania počas simulácie.

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

V letnom semestri sme doplnili kapitoly implementácia, testovanie a používateľská príručka. V implementácii je opísaný model aplikácie, ktoré komponenty s čím súvisia a ako komunikujú. Kapitola testovanie veľmi krátko opisuje priebeh testovania aplikácie. Používateľská príručka ukazuje čitateľovi ako používať aplikáciu.

Tento dokument je výsledkom našej práce na tímovom projekte za dva semestre.

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