Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Software pro odhad počtu studentů na rozvrhových akcích

Ladislav Ježek

Bakalářská práce

2008

SOUHRN

Tato práce se zabývá problematikou určení počtů studentů na rozvrhových akcích. Pro dosažení optimálního počtu studentů na předmětu jsou porovnány hodnoty z předešlých 4 let. Pomocí popisné statistiky jsou zpracována získaná data a pro odhad počtu studentů je navržen vlastní algoritmus.

KLÍČOVÁ SLOVA

popisná statistika, algoritmus, odhad počtu, software, Java

TITLE

Software for estimation of the number of students on schedule actions

ABSTRACT

This work is concerned with the definition of the number of students in schedule actions. For achieving optimum number of students in a subject there is a need to compare rates from last four years. Data are processed by descriptive statistics and self algorithm is designed for estimating the number of students.

KEYWORDS

descriptive statistics, algorithm, estimate of the number, software, Java

Obsah

Seznam obrázků a tabulek

Seznam zkratek

API (Application Programming Interface) – aplikační programové rozhraní

C++ – objektový programovací jazyk

C# – programovací jazyk

DCL (Data Control Language) – jazyk pro ovládání dat

DDL (Data Definition Language) – jazyk pro definici dat

DML (Data Manipulation Language) – jazyk pro manipulaci s daty

EPL (Eclipse Public License) – publikační licence od Eclipsu

GUI (Graphical user interface) – grafické uživatelské rozhraní

IBM – firma pracující v informační technice

JRE (Java Runtime Environment) – rozhraní pro java aplikace

PHP (Personal Home Page) – skriptovací programovací jazyk

PL (Procedural Language) – procedurální jazyk

SQL (Structured Query Language) – strukturovaný dotazovací jazyk

STAG – studijní agenda

SWT (Standard Widget Toolkit) – soubor nástrojů pro grafické rozhraní

1 Úvod

Před začátkem každého semestru musí studijní oddělení každé fakulty co nejpřesněji odhadnout počet studentů, kteří si zapíšou danou rozvrhovou akci. Povinností studijních oddělení je také zajištění potřebného počtu učeben pro všechny studenty zapsané na daný předmět a vypsání dostatečného počtu cvičení nebo seminářů pro jednotlivé rozvrhové akce.

Tato bakalářská práce si klade za cíl zpracovat údaje z předešlých čtyř let, které jsou uloženy ve školní databázi STAG, a přispět tak k přesnějšímu odhadu předpokládaného počtu studentů pro následující semestry a k zajištění dostatečných výukových prostor.

Na zpracování těchto údajů je použito popisné statistiky, sloužící k odhadu počtu studentů, kteří se budou chtít přihlásit v nadcházejícím roce na rozvrhovou akci. Je navržen algoritmus pro vyhodnocování údajů, který je poté implementován do programu, jenž je součástí bakalářské práce. Za cíl je stanoveno dosáhnout správnosti odhadu nad 90% při kontrole se skutečností.

Navrhnutý software spolupracuje se školní databází STAG, kde získává potřebná data, která analyzuje pro odhad počtu studentů na rozvrhových akcích. Aplikace je uživatelsky přátelská, jednoduchá na ovládání a nabízí grafické zobrazení údajů o předmětu a statistice.

2 Popisná statistika

2.1 Historie

První zmínky o uchovávání dat, které měly statistickou povahu, pocházejí z oblasti Sumeru. Sumerové zaznamenávali úrodu, počet osob a kusů domácího dobytka. Od těch dob se statistika dále rozvíjela. Hojně se zaznamenávalo sčítaní lidí a hodnoty se dále zpracovávaly.

Za zmínku stojí rok 1589, kdy Ital Girolamo Ghilini použil jako první termín statistika, ale v původním smyslu se jednalo o stav státu.

Významnou osobností byl belgický matematik, statistik a astronom Adolphe Lambert Quételet (1796-1874), který zavedl některé základní pojmy jako průměr, střední hodnota, rozptyl a rozdělení.

Statistika 19. a počátku 20. století byla především o vytváření a zkoumání vlastností rozsáhlých souborů dat. Začalo se uvažovat, že hodnocení takového množství informací je časově i finančně náročné. Objevily se názory, zda by nestačilo zkoumat jen část dat, tak zvaný reprezentativní vzorek. Na základě této myšlenky se zrodila statistika matematická, která tvoří teoretickou část statistiky. Jedná se o matematickou vědu aplikovanou přímo na problémy spojené se sběrem a pozorováním náhodných dat. Matematická statistika se začala považovat za samostatný obor vyšší matematiky.

2.2 Úvod do popisné statistiky

Popisná statistika zjišťuje a poskytuje číselné i slovní údaje (informace) o jevech hromadné povahy. Statistickým zjišťováním zpravidla provádíme měření nebo popis hmotných objektů (např. lidí, aut, měst), čímž získáme velké množství údajů, které nazýváme hromadná data. Nezpracovaná data jsou pouze chaotické a neuspořádané údaje, ze kterých se nedají vyčíst prakticky žádné užitečné informace, pokud neprovedeme další zpracování. A zpracování takových hromadných dat, které vedou k získání informací a zákonitostí těchto dat, se zabývá právě popisná statistika.

2.3 Základní pojmy

2.3.1 Statistický soubor

Statistický soubor je množina všech prvků, které jsou předmětem daného statistického zkoumání a každý prvek tvoří jednu statickou jednotku. Prvky mají společné identifikační znaky, které umožňují určit, zda prvek patří nebo nepatří do statistického souboru. Na prvcích statistického souboru sledujeme jednu nebo více vlastností z hlediska cílů statistického zkoumání. Při sledování pouze jednoho znaku získáme jednorozměrný statistický soubor, pokud sledujeme více znaků, dostáváme vícerozměrný statistický soubor.

2.3.2 Sledované statistické znaky

Sledované znaky dělíme na kvantitativní a kvalitativní. Kvantitativní znaky nabývají číselných hodnot (např. počet, cena, doba, délka, pevnost,

…), narozdíl od kvalitativních, které nemají číselný charakter a lze je vyjádřit slovně (např. tvar, barva, jakostní třída, …).

Kvantitativní znaky dělíme na diskrétní, jestliže nabývají pouze oddělených číselných hodnot, nebo spojité, pokud nabývají všech hodnot z nějakého intervalu reálných čísel. Mezi diskrétní znaky patří kusová výroba, počet vad apod. Rozměr výrobku, cenový index, doba do poruchy apod. patří do spojitých znaků.

Kvalitativní znaky se rozdělují na ordinální a nominální. Ordinální znaky jsou slovní hodnoty mající smysl, podle kterého se dají uspořádat (jakostní třídy, klasifikace apod.). Slovní hodnoty nominální znaků postrádají význam pořadí a převažují nad ordinálními znaky. Patří do nich barva, tvar, dodavatelé apod.

2.3.3 Aritmetický průměr

Aritmetický průměr je statistická veličina, která nám určuje průměrnou hodnotu všech hodnot ze statistického souboru, obvykle se značí vodorovným pruhem nad názvem proměnné. Hodnota se získává součtem všech prvků ve statistickém souboru a vydělením jejich počtem.

$$\overline{x} = \frac{1}{n}\left(x_1 + x_2 + \ldots + x_n\right) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Přiklad:

Statistický soubor

x	17	15	16	14	23	17

Aritmetický průměr je roven: $\overline{x} = 17$

2.3.4 Vážený průměr

Vážený průměr zobecňuje aritmetický průměr a používá se, pokud hodnoty ve statistickém souboru mají různou důležitost (váhu). Využívá se hlavně při počítání celkového aritmetického průměru, který se skládá z dílčích souborů, nebo pokud máme statistický soubor o n hodnotách a k němu odpovídající množinu vah o n prvcích. Hodnota se spočítá součtem součinu prvku a příslušné váhy a vydělením součtu vah.

$$\overline{x} = \frac{w_1 x_1 + w_2 x_2 + \ldots + w_n x_n}{w_1 + w_2 + \ldots + w_n} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Přiklad:

Statistický soubor a k němu odpovídající váhy

x	17	15	16	14	23	17
w	2	2	2	2	1	2

Vážený průměr je roven: $\overline{x} = 16{,}45$

2.3.5 Klouzavý průměr

Klouzavý průměr nám poskytuje přesnější průměrnou hodnotu. Používá se, pokud se ve statistickém souboru nachází několik hodnot, které se výrazně liší od ostatních hodnot. Klouzavý průměr nám tyto rozdíly sníží (vyhladí) výpočtem aritmetického průměru v okolí. Celý klouzavý průměr se spočítá tak, že si zvolíme délku (počet hodnot) - zpravidla se volí lichý počet, aby okolí kolem hodnoty bylo stejně velké na obou stranách. Problém je, že nemůžeme vyhladit hodnoty na začátku a konci statistického souboru, protože nesplňují podmínku okolí. Pokud se zvolí délka klouzavého průměru $d = 5$, tak nemůžeme vyhladit dvě hodnoty na začátku a na konci statické-

ho souboru. Ve statistickém souboru postupujeme vždy o jednu hodnotu n+1, tím získáme o d - 1 méně hodnot.

$$c = \frac{d-1}{2}$$

$$\overline{x} = \frac{1}{n-(d-1)} \sum_{i=c}^{n-c} \left(x_{i-c} + \ldots + x_{i-1} + x_i + x_{i+1} + \ldots + x_{i+c} \right)$$

Přiklad:

Statistický soubor

x	17	15	16	14	23	17

Délka klouzavého $d = 3$ průměru:

Soubor po vyhlazení

x_2	16	15	17,6	18

Klouzavý průměr je $\overline{x} = 16{,}65$ roven:

2.4 Ukázkový příklad

Je dán datový soubor se známkami z matematiky dvaceti studentů, spočítá se relativní četnost jedničkářů, relativní četnost úspěšných studentů, tečkový diagram. Dále se spočítá aritmetický průměr, průměrná odchylka, rozptyl a směrodatnou odchylku.

Datový soubor:

x = {2, 1, 4, 1, 1, 4, 3, 3, 1, 1, 4, 4, 2, 4, 2, 4, 1, 4, 4, 1}

n = 20

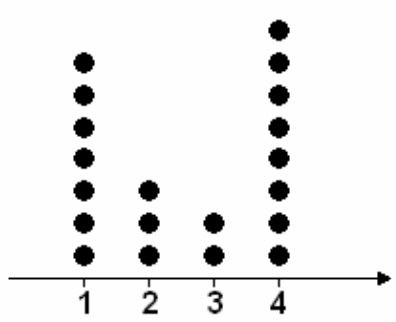
Relativní četnost jedničkářů:

$$p(x=1)=\frac{7}{20}=0{,}35$$

Relativní četnost úspěšných studentů:

$$p(x\le 3)=\frac{11}{20}=0{,}55$$

Tečkový diagram:



Aritmetický průměr:

$$m=\frac{1}{n}\sum_{i=1}^{n}x_i=\frac{51}{20}=2{,}55$$

Průměrná odchylka:

$$o=\frac{1}{n}\sum_{i=1}^{n}\left|x_i-m\right|=\frac{25}{20}=1{,}25$$

Rozptyl:

$$s^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-m\right)^2=\frac{34{,}95}{20}=1{,}75$$

Směrodatná odchylka:

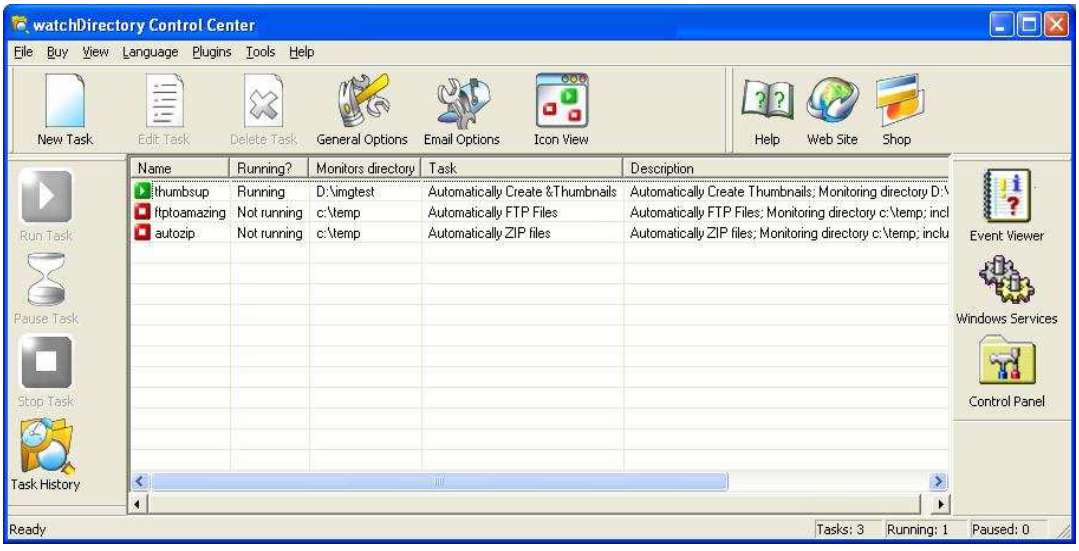
$$s=\sqrt{s^2}=1{,}32$$

3 Software

3.1 User-friendly software

Do českého jazyka přeloženo jako uživatelsky přátelský software. Naprogramovat takovýto software není nic jednoduchého, protože ve světě je mnoho uživatelů a každý z nich očekává od používaného programu něco jiného.

Základem pro splnění této podmínky je dobrá volba rozhraní pro komunikaci mezi uživatelem a počítačem. Velmi příjemné je grafické uživatelské rozhraní (anglicky *Graphical User Interface* – Obr. 1., známé pod zkratkou GUI). Takové rozhraní je pro uživatele rozhodně příjemnější než příkazový řádek v konzole. Aplikace s GUI komunikuje s počítačem pomocí interaktivních grafických prvků a celý program je přehlednější.



Obr 1. Ukázka grafického uživatelského rozhraní

Další věcí, kterou si uživatelé na softwaru chválí, je nápověda. Pokud je ale nápověda příliš rozsáhlá, stává se pro uživatele nepříjemnou, zejména z hlediska dlouhého vyhledávání potřebné informace. Vhodné je do nápovědy doplnit vyhledávač, klávesu F1 pro spuštění nápovědy a přidat tlačítka k některým grafickým komponentům, pro které je vhodné použít krátkou nápovědu na její použití.

Hodně také záleží na detailech zobrazení. Menu bar (řádkové menu) už obsahuje každý program, dále ho doplňuje tool bar (řádek nástrojů), který zobrazuje obrázkové ikonky vystihující službu nástroje. Dále je vhodné umísťovat tlačítka na jejich obvyklé místo. Uživatelé jsou zvyklí u nastavujících dialogů (oken) hledat tlačítka „Ok“, „Storno“ a popřípadě „Použít“ v pravém dolním rohu. Tlačítka by měla být dostatečně veliká a čitelná. Veškerá tlačítka, která se zaškrtávají nebo mačkají, by měla být dostatečně velká, aby uživateli nedělalo problém se s myší na ně trefit.

Celý software je nutné zabezpečit po stránce programátorské, aby nebyl přítěží pro uživatele. Uživateli nesmí být povolena taková operace, která by vedla k pádu programu. Je zapotřebí veškeré tyto stavy podchytit, aby k nim nedošlo. Software by byl nestabilní a uživatelé nespokojeni. Také není dobré, je-li této situaci zabráněno způsobem, že operace sice proběhne v pořádku, ale nenastane požadovaná změna. Přátelsky uživatelský software je stále stabilní a kontroluje uživatele, zda vše správně vyplňuje, případně ho upozorní varovnou hláškou.

Pro programátora je příjemný interface velmi důležitý, můžeme uvést například Eclipse pro vývojáře v programovacím jazyce Java. Program splňuje veškeré požadavky výše uvedené a dále zvyšuje pracovní komfort uživateli jako už i ostatní podobné programy. Kontroluje právě psaný zdrojový

kód, pokud programátor zapomene jen středník nebo špatný typ proměnné, program hned celý řádek červeně podtrhne a na straně se zobrazí chybový znak. Mírnější stupeň je varování, pro které je charakterizovaná žlutá barva. Červená barva pro chybové a žlutá pro varovné hlášení je užita téměř ve všech programech. Program dokončuje slova a nabízí vhodné funkce po stisknutí kombinace kláves Ctrl+mezerník. A nabízí plno dalších příjemných funkcí.

3.2 Oracle Database 10*g*

Školní databáze STAG je provozována právě na Oracle Database 10g. Tento databázový server vyvíjí společnost Oracle, která nabízí kompletní řešení podnikové informační infrastruktury a založila u nás pobočku v roce 1994 s cílem přiblížit produkty a služby zákazníkům v České republice.

Databáze poskytuje automatizované časově náročné administrativní úkoly, které jsou náchylné na chyby. Dále nabízí širokou škálu bezpečnostních mechanismů, takže ji předurčují k nasazení na prostředí s vysokými nároky na bezpečnost informací.

Oracle Database 10*g* je relační databáze, která je založená na relačním modelu dat a relační algebře. Data jsou uspořádaná do tabulek (relací) a nad tabulkami jsou definovány operace. Pro ovládání databáze slouží standardní relační dotazovací jazyk SQL podle normy SQL92, dále nabízí možnost využít programovací jazyk PL/SQL, který umožňuje tvořit procedury, uživatelské funkce, triggery atd. Programovací jazyk SQL obsahuje čtyři základní skupiny příkazů.

Základní skupiny příkazu:

- DML – Data Manipulation Language slouží pro manipulaci s daty,
- DDL – Data Definition Language jsou na vytváření struktury databáze (tabulky),
- DCL – Data Control Language se používají pro nastavení přístupových práv a řízení transakcí,
- Ostatní příkazy – zde patří příkazy pro správu databáze, přídávaní uživatelů, nastavovaní systémových parametrů atd.

Pro naši potřebu na získání dat ze školní databáze STAG se zaměříme na skupinu příkazů DML, do této skupiny patří příkazy SELECT (vybírá data z databáze), INSERT (vkládá nová data), UPDATE (mění data v databázi), DELETE (odstraňuje záznamy z databáze) a jiné. Nám postačí příkaz SELECT pro vybírání dat z databáze.

Ukázka složení příkazu pro výpis počtu studentů zaspaných na předmětu IJC+2 v jednotlivých letech:

```
SELECT ROK_VARIANTY, SUM(POCET_STUDENTU) FROM INSTALL2.UPA_FEI_PREDMETY WHERE ZKR_PREDM='IJC+2' AND SEMESTR='ZS' GROUP BY ROK_VARIANTY ORDER BY ROK_VARIANTY
```

Rozebrání jednotlivých částí příkazu:

- SELECT ROK_VARIANTY, SUM(POCET_STUDENTU) – zde je vybráno, které sloupečky chceme vypsat,
- SUM(POCET_STUDENTU) – funkce slouží pro sčítaní hodnot, které musí být rozděleny do skupin podle určitého požadavku viz dále

v příkazu, funkce je použita, protože si některý předmět může zapsat více oborů a záznam je rozdělený do více řádků,

- FROM INSTALL2.UPA_FEI_PREDMETY – výběr tabulky, ve které se nachází data, před tečkou je název uživatele, který vlastní tabulku,
- WHERE ZKR_PREDM='IJC+2' AND SEMESTR='ZS' – podmínky výběru podmnožiny řádků z tabulky,
- GROUP BY ROK_VARIANTY – vytvoření skupin podle roku pro funkci SUM,
- ORDER BY ROK_VARIANTY – pro seřazení řádku vzestupně od nejstaršího roku.

3.3 Programovací jazyk JAVA

Programovací jazyk vyvinula firma Sun Microsystems. Jedná se o objektově orientovaný jazyk. Ve svém vzniku se podobal jazyku C++, ke kterému má syntakticky nejblíže, pokud nebereme v úvahu C#, který vznikl později.

Oproti předchůdcům programu Java je programovaní v Javě lehčí a neobsahuje některé konstrukce, které způsobovaly při programování největší potíže. Odpadla starost s přidělováním a uvolňováním paměti, vše je vyřešeno automaticky. Objekt se ruší například, jen přiřazením neplatné reference null. Dále jsou zcela odstraněny ukazatele, jsou nahrazeny referencemi, tím mizí chyby při programování, když docházelo nechtěnému zápisu ukazatelem mimo data.

Jazyk nabízí implementaci mechanismu vláken, lze spouštět více úloh v rámci jednoho programu a je pamatováno i na jejich synchronizaci pomocí

tzv. monitorů. Další výhodou je používání výjimek, takže lze odchytit a zpracovat runtime chyby (chyby vzniklé při běhu programu). Značným ulehčením pro programátory je obsáhlost standardně dodávaných knihoven. Největší výhodou tohoto programovacího jazyka je přenositelnost zdrojového kódu, například programátor napíše program na počítači pod Windows a pustit půjde na jakémkoliv jiném operačním systému. Zkrátka všude, kde je aspoň k dispozici JRE (Java Runtime Environment), prostředí pro běh programů v Javě.

„**REDWOOD SHORES, Kalifornie, 25. července 2005** - Společnost Oracle oznámila, že stvrzuje svůj závazek ke komunitě vývojářů v jazyce Java a nabízí jim bezplatně nástroj Oracle JDeveloper 10g. Oracle se také uchází o vedení nástrojového programu JavaServer Faces (JSF) v open source komunitě Eclipse Foundation a chce se jako klíčový přispěvovatel zúčastnit projektu Apache MyFaces.“

Zdroj: webový server Oracle [8]

Společnost Oracle nejen že nabízí svoje vlastní vývojové prostředí, ale nabízí i knihovny pro lepší komunikaci Javy s jejími produkty. Hlavně pro komunikaci s databází, proto jsem si vybral tento programovací jazyk a vývojové prostředí Eclipse.

3.4 Vývojové prostředí Eclipse

Eclipse je open source vývojová platforma, určená hlavně pro programování v jazyce Java. Toto vývojové prostředí je flexibilní a za pomocí pluginů umožňuje rozšířit seznam programovacích jazyků, například o C++ nebo PHP. V základní verzi obsahuje pouze integrované prostředí pro vývoj Javy (kompilátor, debugger atd.). Všechny další nástroje

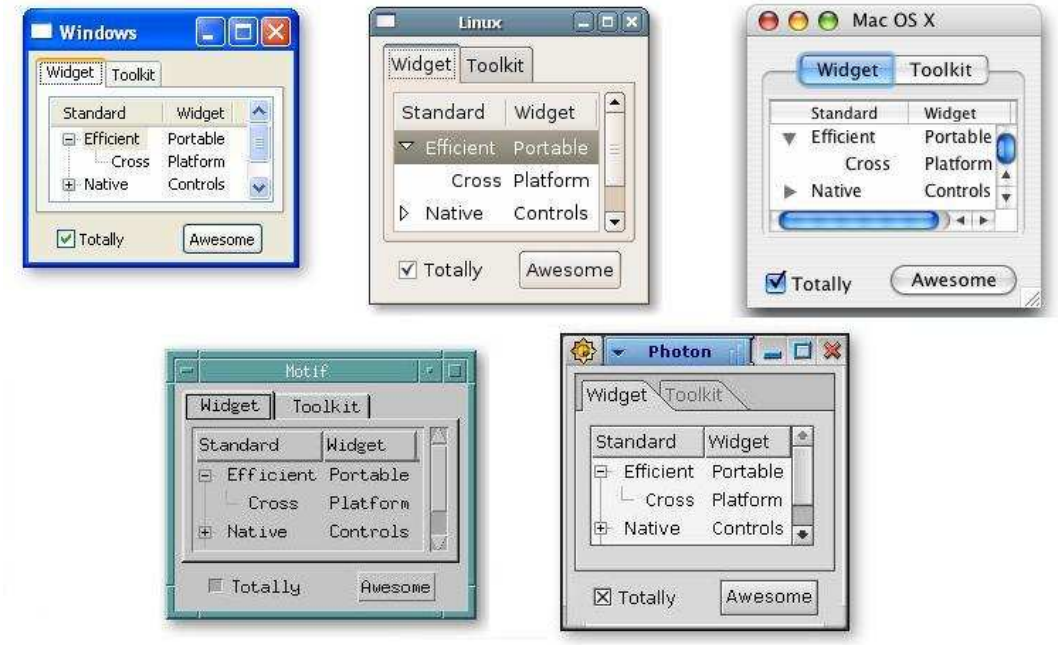
například vizuální návrh grafického uživatelského rozhraní nebo aplikační server si musí uživatel přidat formou pluginů. Díky tomu vznikly takzvané subprojekty, které jsou pod záštitou Eclipse. Tyto subprojekty se zaměřily na jednotlivé oblasti softwarového vývoje v Javě a usnadňují integraci potřebných rozšíření do samotného vývojového prostředí.

Vývojové prostředí vyvíjí Eclipse Foundation a projekt Eclipse vnikl díky uvolnění kódu od IBM pod EPL (Eclipse Public License) licencí. Na základě tohoto projektu byl vyvinut grafický rámec SWT (Standard Widget Toolkit), který používá nativní prvky operačního systému.

3.5 SWT (Standard Widget Toolkit)

SWT je grafická knihovna v jazyce Java. Používá nativní prvky z prostředí operačního systému, na kterém je program spuštěn, v tom je její hlavní výhoda. Na základě této výhody je na platformě nezávislá. Pro její spuštění jsou potřebné jen knihovny pro určitý operační sytém, např. pro Windows knihovny typu *.dll podle použité verze externího souboru swt*.jar (*.jar – Java Archiv několik souborů *.class).

Uživatel vidí známé prostředí svého operačního systému (viz Obr. 2.), narozdíl od grafické knihovny Swing, která používá svoje vlastní grafické komponenty a je vzhledově slabá. Když Microsoft uvolnil XP verzi Windows, SWT automaticky převzalo jeho nový vzhled, Swing toho však není schopen. Tímto je grafická knihovna SWT pro uživatele příjemnější a lépe se mu bude pracovat s programem.



Obr. 2. Ukázka SWT aplikace na různých platformách [10]

Grafická knihovna SWT má pro programátory pár nevýhod. Pro tvorbu formulářů neexistuje žádný funkční návrhář, ale pomocí správců rozvržení se toto dá zvládnout a není k dispozici kvalitní komponenta pro zobrazení většího množství dat v mřížce. Jediné řešení je napsat si vlastní komponentu sám. SWT není možné u alokovaných zdrojů jako jsou fonty a barvy se spolehnout na automatickou správu paměti pomocí garbage collection, ale tyto zdroje je nutné uvolňovat ručně. Většinu SWT tříd nelze dědit.

SWT je základní grafická knihovna pro Javu, která nemá žádnou závislost na standardním grafickém API (Application Programming Interface) Javy. Jeho výkladní skříň je vývojové prostředí Eclipse a je IBM iniciativou.

4 Algoritmus odhadující počet studentů na předmětu

4.1 Určení základních informací

Zápis rozvrhových akcí probíhá před začátkem každého semestru. Předměty se rozlišují na povinné pod statutem „A“, dále předměty povinně volitelné „B“ a volitelné „C“. Povinné předměty musí být splněny, proto si je student v případě neúspěchu zapíše podruhé. U ostatních předmětů si nemůžeme být jisti, protože si student při nezvládnutí povinně volitelného předmětu může zvolit příští rok jiný předmět. Nejmenší pravděpodobnost, že si student zapíše shodný předmět i příští rok, je u předmětů volitelných, které si student většinou zapisuje pouze pro získání dostatečného počtu kreditů.

Nejjednodušší pro odhadování počtu studentů by bylo, kdyby vyučovali roboti. Učili by stále za stejných podmínek, nezaujatě, a odhad počtu neúspěšných studentů by se nemusel měnit. Realita je ale složitější. Musíme uvažovat, že každý předmět většinou vyučuje více než jeden vyučující a každý vyučující má odlišné požadavky pro zvládnutí předmětu. Tím se nám mění pravděpodobnost neúspěšnosti studenta. Významný je i vliv času, hlavně u technicky zaměřených předmětů, kde se poznatky rychle vyvíjí. Pokrok lidstva nelze zastavit.

Dalším probléme je, že Fakulta elektrotechniky a informatiky (dříve Ústav informatiky) vznikla před čtyřmi lety, tudíž není k dispozici mnoho uchovaných dat a nemůžeme zaručit dostatečně věruhodné statistické výsledky. Fakulta se dále rozvíjí a nabírá více studentů, dobře patrné je to

na předmětu IJC+2 (Jazyk C++ 2), kde v roce 2004 bylo zapsáno 23 studentů a v roce 2007 už 119 žáků.

4.2 Získání hodnot

Školní databáze STAG obsahuje více než 50 tabulek, kde jsou zaznamenány informace o všech předmětech, fakultách, zaměstnancích, studentech atd. Pro získaní potřebních údajů byly vytvořeny dva pohledy nad tabulkami. Jeden pohled je UPA_FEI_PREDMETY, který obsahuje sloupce:

- PRAC_ZKR – zkratka fakulty, na které se vyučuje daný předmět,
- ZKR_PREDM – zkratka předmětu
- ROK_VARIANTY – rok, pro který jsou údaje platné,
- CISOBORU – číslo oboru, pro který je daný předmět určen,
- SEMESTR – semestr, ve kterém se vyučuje předmět,
- DOPORROC – doporučený ročník zápisu předmětu,
- STATUT – statut předmětu,
- POCET_STUDENTU – počet studentů zapsaných na předmětu,
- OPAKUJE_PREDMET_V_ROCE – počet opakujících z počtu zapsaných studentů v oboru.

Pokud si daný předmět může zapsat více oborů, tak je záznam uveden ve více řádcích, pro každý obor zvlášť. Tento fakt musíme zohlednit při odhadu počtu studentů v dalším roce.

Druhý pohled tabulky je UPA_FEI_POCTY_STUDENTU, který obsahuje sloupce:

- ROK_PLATNOSTI – rok, který jsou údaje platné,
- CISOBORU – číslo oboru,

- ROCNIK – ročník oboru,
- POCET – počet studentů na oboru.

Pomocí databázového dotazu pro lepší zpracovaní vyfiltrujeme a setřídíme hodnoty podle roku varianty vzestupně (viz. Tab. 1. a Tab. 2.).

Tab. 1. Výpis jednoho předmětu z pohledu UPA_FEI_PREDMETY

ZKR_PREDM	ROK_VARIANTY	POCET_CELKEM	Z_TOHO_OPAKUJE
IJC+2	2004	23	0
IJC+2	2005	46	1
IJC+2	2006	84	2
IJC+2	2007	119	29

Zdroj: Školní databáze STAG

Tab. 2. Výpis počtu studentů v 3. ročníku z pohledu UPA_FEI_POCTY_STUDENTU

rok	2003	2004	2005	2006	2007
počet v 2. ročníku	46	76	91	114	128
počet v 3. ročníku	7	22	48	87	129

Zdroj: Školní databáze STAG

Z těchto hodnot spočítáme procentuální účast opakujících studentů na předmětu (Tab. 3.) tak, že spočítáme podíly mezi počtem opakujících a zapsaných studentů na předmětu IJC+2.

Tab. 3. Procentuální počet opakujících studentů

rok	2004	2005	2006	2007
poměr [%]	0	2,17	2,38	24,37

Dále zjistíme procentuální účast studentů zapsaných na předmětu z celkového počtu studentů z ročníku (Tab. 4.), pomocí podílu mezi počtem studentů zaspaných na předmět IJC+2 a počtem studentů zapsaných ve 3. ročníku.

Tab. 4. Procentuální počet zapsaných studentů

rok	2004	2005	2006	2007
poměr [%]	104,55	95,83	96,55	92,25

Ještě potřebujeme zjistit procentuální počet studentů, kteří se zapíšou do ročníku z minulého roku (Tab. 5.) tak, že spočítáme podíl mezi počtem zaspaných studentů v 3. ročníku a počtem zapsaných studentů ve 2. ročníku minulý rok.

Tab. 5. Procentuální zápis studentů z 2. ročníku do 3. ročníku

rok	2004	2005	2006	2007
poměr [%]	47,83	63,16	95,6	113,16

Teď máme získané všechny údaje o předmětu v jednotlivých letech. Abychom z těchto údajů zjistili více, můžeme spočítat aritmetické nebo vážené průměry.

4.3 Vlastní algoritmus

Navrhnout výkonný a spolehlivý algoritmus není jednoduché, někdy je potřeba celý tým lidí. Při návrhu algoritmu vycházíme z předešlých informací o systému. Poté se algoritmus testuje, dosažené výsledky se hodnotí se skutečnými hodnotami v předešlých letech.

Při návrhu algoritmu jsem nebyl na první pokus úspěšný, můj odhad počtu studentů na rozvrhové akci byl nízký, okolo 30 %, které jsou pro stanovený cíl nevyhovující.

Spočítal jsem odhad počtu studentů, kteří budou opakovat předmět, tak, že jsem spočítal průměrnou hodnotu procentuálních hodnot opakujících studentů na předmětu v jednotlivých letech a získanou průměrnou hodnotou jsem vynásobil počet zapsaných studentů. Odhad počtu studentů na předmětu jsem zjistil tak, že jsem k počtu studentů zapsaných v nižším ročníku přičetl odhad opakujících studentů z předmětu a hodnotu jsem vynásobil průměrnou hodnotou z procentuálních hodnot zapsaných studentů na předmět v jednotlivých letech.

Problém mi činily lišící se počty studentů v jednotlivých letech, což ovlivňovalo průměrnou hodnotu. Dosahoval jsem nízké procentuální hodnoty správnosti odhadu.

Začal jsem aplikovat vážený průměr, tím se mi začala procentuální pravděpodobnost odhadu zvyšovat. Větší váhu jsem kladl na hodnoty z posledních let. Tím jsem vyřešil vzrůstající počet zapsaných studentů v jednotlivých letech.

Zjistil jsem, že čím větší důraz kladu na hodnoty z posledních let, tím získávám vyšší pravděpodobnost správného odhadu Spokojil jsem se se správností odhadu okolo 90%. Při kontrolním testování na předmětu z třetího ročníku, následně po vyzkoušení na předmětu z druhého ročníku procentuální správnost odhadu prudce klesla na hodnoty okolo 45%. S výsledkem jsem opět nebyl spokojen.

Začal jsem zjišťovat , kde se vyskytl problém, proč na předmět v třetím ročníku je o 40% lepší odhad než na předmět ve druhém ročníku. Zjišťoval jsem, kolik se procentuálně zapíše studentů z minulého ročníku do dalšího. Dopátral jsem se toho, že z druhého ročníku se zapíše do třetího ročníku okolo 98% studentů, ale z prvního ročníku do druhého jen okolo 53% studentů. Tím jsem dospěl k závěru, že když je student v prvním ročníku a nedaří se mu, tak studium na vysoké škole ukončuje.

Problém jsem začal řešit: spočítal jsem odhad studentů zapsaných do ročníku z předchozího ročníku. Z předešlé skutečnosti jsem už používal vážený průměr s větší váhou na hodnoty z posledních let.

V konečném řešení si nejdříve spočítám odhad studentů, kteří se zapíšou do dalšího ročníku. Hodnotu si zapamatuji a spočítám odhad neúspěšných studentů na předmětu. Tyto dvě hodnoty sečtu a vynásobím průměrnou hodnotu z procentuálních hodnot zapsaných studentů na předmět v jednotlivých letech. Tím získám odhad počtu studentů na rozvrhové akci. Algoritmus funguje na předměty v druhém a třetím ročníku.

Pro první ročník vynechávám odhad počtu zápisu z minulého ročníku, protože není možné počet studentů odhadnout. Řeším to tak, že pro první ročník se zadává počet přijatých studentů do oboru.

4.4 Algoritmem dosažené výsledky

Po zjištění, že je potřeba používat váženého průměru, jsem nasbíral výsledky rozdílných vah na poslední dva roky, abych rozhodl o optimálních hodnotách, které budou použity v softwaru pro odhad počtů studentů na rozvrhových akcích. Výsledky některých vybraných předmětů jsem zaznamenal, pro přehlednost, do tabulek (viz Tab. 6., Tab. 7., Tab. 8.).

Tab. 6. Procentuální správnost odhadů

Procentuální správnost odhadů na letošní rok						
váha na r.		zkratka předmětu				
př. r.	p. r.	IJC+3	ISOSY	ISPWE	IPNIN	IDAS2
1	6	93,87	100	96,77	88,3	98,04
2	4	90,82	97,83	97,85	84,04	94,12
0	4	92,86	98,91	96,77	82,23	99,02
1	3	89,3	98,91	97,84	82,97	95,1
aritm. pr.		80,61	95,65	98,92	72,34	90,2

Tab. 7. Procentuální správnost odhadů

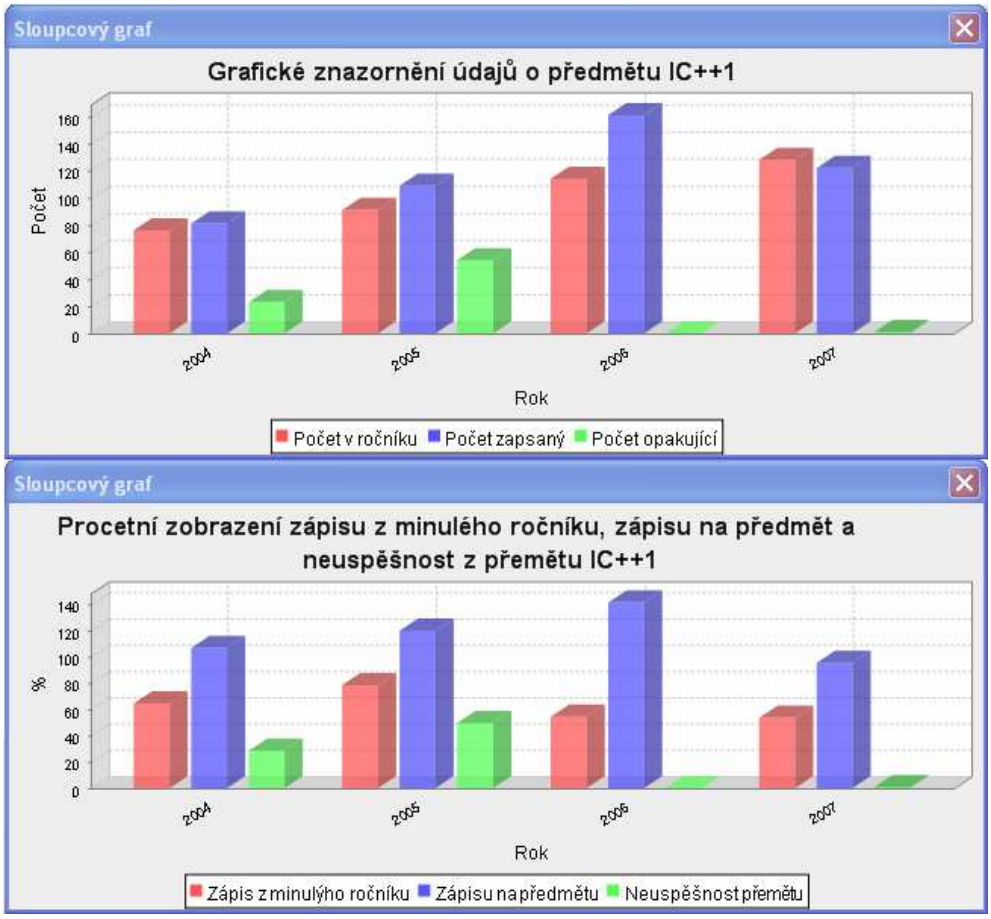
Procentuální správnost odhadů na letošní rok						
váha na r.		zkratka předmětu				
př. r.	p. r.	IEKPO	IJC+2	IOSYS	IWWW	IC++
1	6	98,95	100	100	93,33	70,49
2	4	98,95	95,8	98,95	93,33	61,48
0	4	98,95	97,48	100	93,33	66,39
1	3	98,95	94,12	98,95	93,33	59,01
aritm. pr.		92,63	78,99	92,63	94,44	27,86

Tab. 8. Procentuální správnost odhadů

Procentuální správnost odhadů na letošní rok						
váha na r.		zkratka předmětu				
př. r.	p. r.	IDAS1	IJAV	IZMAR	IOOP	IUJC
1	6	88,1	98,43	95,86	98,84	99,31
2	4	86,51	96,88	95,87	95,93	98,61
0	4	88,09	98,44	94,21	97,09	97,22
1	3	86,51	98,44	95,04	97,09	100
aritm. pr.		83,33	95,31	90,08	93,02	96,52

Protože kontrolujeme odhad počtu studentů v letošním roce, lze z výsledků vypozorovat, že u většiny předmětů záleží na váze z posledního roku, který nás zejména zajímá. Ale nesmíme zvolit příliš velkou váhu na poslední rok, protože by ostatní hodnoty byly zanedbané. Zvolil jsem proto váhu na předposlední rok jedna a na poslední rok tři. Váha spočívá v tom, že daná hodnota je použita vícekrát (tzn. že předposlední rok zaznamenám ještě jednou a poslední rok třikrát a vydělím celkovým počtem hodnot).

S dosaženými výsledky, za použití vah na předposledním roku jedna a posledním roku tři, jsem spokojený. Považuji je za dostatečné, neboť se u většiny předmětů dostanu nad 90% správnosti odhadu a nejsou zanedbány ani hodnoty z dalších let. Pouze u několika málo předmětů tomu tak není, protože je buď málo záznamů o předmětu (třeba jen dva roky), nebo jejich aritmetický průměr je pod 50%, což znamená velké výkyvy v jednotlivých letech. Tohoto si můžeme povšimnout na předmětu IC++ (viz Obr. 3.).



Obr. 3. Grafické znázornění předmětu IC++1 z 2. ročníku
Zdroj: Navržený vlastní software

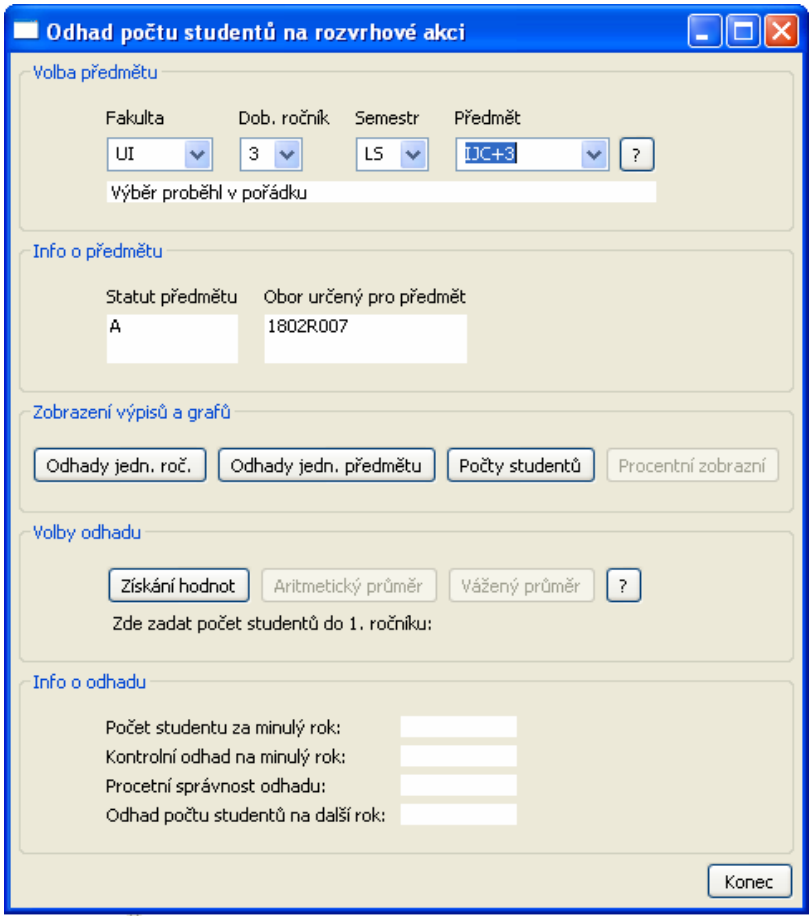
Z grafů lze vyčíst, jaké nastaly změny po akademickém roku 2005/2006. Tato změna je důsledkem nízké správnosti odhadu počtu studentů na tomto předmětu. Problém nastal změnou vyučujícího, který má jiné nároky nebo dosavadní vyučující změnil zásadně svoje nároky na studenty. Protože v ak. roce 2005/2006 opakovala předmět skoro polovina studentů zapsaných na předmětu, v ak. roce 2006/2007 už nikdo neopakoval a hodně studentů se jej zapsalo poprvé až ve třetím ročníku (to ukazuje vyšší počet zapsaných studentů než studentů v ročníku). Poslední rok je patrné, že předmět je méně náročný na zvládnutí, protože se přihlásilo méně zapsaných studentů než je počet studentů v ročníku.

Dále je patrné, že univerzita nabírá do prvních ročníků stále více studentů, ale do druhého ročníku se zapíše jen něco přes 50% studentů. Toto jsem už zmiňoval výše u problému při návrhu algoritmu.

5 Software pro odhad počtů studentů

5.1 Popis programu

Software je navržen jednoduše a s user-friendly rozhraním. Program má grafickou podobu viz. Obr. 4.



Obr. 4. Zobrazení programu

Na obrázku lze vidět, že program používá nativní prvky operačního sytému, kterým je Windows XP. Program neobsahuje menu bar ani tool bar, protože nejsou pro jednoduchost programu třeba. Můžeme si všimnout, jak jsou vhodně použity boxy pro přehlednost programu a umístění tlačítka na ukončení programu v dolním pravém rohu, kde by ho většina uživatelů hledala.

Program nabízí grafické zobrazení údajů o předmětu a statistice (viz Obr. 4.1 výše), ve kterém jsou vidět názorně jednotlivé roky. Dále je možnost výpisu seznamu předmětu pro celkové zjišťování počtů studentů. Tyto možnosti dělají program uživatelsky příjemnější. V horní části programu je textové pole, které informuje uživatele, co je potřeba provést.

Z bezpečnostního hlediska jsou využity vlastnosti ovládacích prvků, které můžou být aktivní nebo neaktivní. Tím je zabráněno, aby uživatel nemohl chtít operaci, která by právě vedla k pádu programu. Tím, že se ovládací prvky postupně stávají aktivní, je program pro uživatele příjemnější, protože si všimne jaké operace má právě k dispozici. Pokud se vyskytne chyba při běhu programu, je programem zachycena a uživatel upozorněn dialogem.

Všechny tyto aspekty, které jsou výše poznamenány, se snaží, aby interface programu byl pro uživatele nejpříjemnější a nejpřátelštější. Program je pro uživatele jednoduchý k ovládání a popis tlačítek vystihuje činnost, která je pod nimi ukryta.

5.2 Zdrojový kód algoritmu

Níže uvedený kód slouží pro získání procentuálního odhadu neúspěšných studentů na předmětu pomocí váženého průměru. VectorNeuspech obsahuje procentuální počty opakujících studentů v jednotlivých letech. Podobně se spočítá odhadZapisu a odhadZapDoDalRoku.

```
…
odhadNeuspechu = 0;
for (int i=0; i<vectorNeuspech.size(); i++){
        DoubleHodnota hodnota =
        (DoubleHodnota)vectorNeuspech.get(i);
        odhadNeuspechu = odhadNeuspechu + hodnota.getHodnota();
        if (i+2 == vectorNeuspech.size()){
                odhadNeuspechu = odhadNeuspechu +
                (hodnota.getHodnota() * 1);
                }
                if (i+1 == vectorNeuspech.size()){
                        odhadNeuspechu = odhadNeuspechu +
                        (hodnota.getHodnota() * 3);
                }
        }
odhadNeuspechu = odhadNeuspechu / (vectorNeuspech.size() + 4);
…
```

Po získání všech tří odhadů se spočítá odhad počtu studentů, kteří se zapíšou na předmět.

```
…
double pocdalsroce = pocMinRok * odhadZapDoDalRoku;
double pocneusp = zapMinRok * odhadNeuspechu;
double pocstud = (pocdalsroce + pocneusp) * odhadZapisu;
odhadPoctu = (int)pocstud;
…
```

Nejdříve se spočítá odhad počtu studentů, kteří se zapíšou do dalšího ročníku, a odhad počtu studentů, kteří si předmět zapíšou podruhé. Tyto dvě hodnoty se sečtou a vynásobí hodnotou odhadZapisu. Tak se získá odhad počtu studentů zapsaných na předmětu.

6 Závěr a hodnocení

6.1 Program

Program je napsaný v programovacím jazyce Java, pro lepší komunikaci s databází od společnosti Oracle, která tento jazyk podporuje. Využívá grafické knihovny SWT, aby program zobrazoval nativní prvky operačního systému a tím se stal pro uživatele příjemnější, protože uživatel vidí interface, který zná. Společně s výhodami, o niž jsem se zmínil výše, dělají program uživatelsky příjemný.

Do programu jsem přidal výpis odhadu počtů studentů do všech ročníku a výpis odhadu zápisů na předmět, filtrovaný podle ročníku a semestru, aby program nabídl více služeb pro studijní oddělení, které mělo zájem o informace ohledně této služby. Vidíme tedy, že každý uživatel má jiné

nároky a nelze vyhovět všem. Závěrem bych dodal, že program je uživatelsky přátelský a další vhodné doplňky ukáže až jeho používaní.

6.2 Algoritmus pro odhad počtů studentů

Na začátku jsme si stanovili cíl, že správnost odhadu bude nad 90%. Kontrola správnosti se provádí tak, že odhad provedeme z hodnot z předešlého roku a porovnáme, jak se moc liší hodnota od skutečnosti. Tohoto cíle jsem u většiny předmětu docílil, u některých dokonce i s úspěšností nad 95%. U některých předmětů se mi ale cíle dosáhnout nepodařilo. Tento problém vzniká tím, že u některých předmětů je méně záznamů ve STAGU, buď se předmět nevyučuje nebo se teprve začal vyučovat. Výsledek je také ovlivněn změnou nároků pro absolvování předmětů během posledních dvou let.

Problém se nachází v tom, že záznamy o předmětu jsou k dispozici jen za poslední čtyři roky a zjišťovat statistiku ze čtyř hodnot je málo, neboť každá hodnota více odlišná od průměru nám značně ovlivňuje výsledek. Řešením by mohlo být například použití klouzavého průměru, který tyto hodnoty vyhlazuje, ale pro jeho použití potřebujeme více hodnot.

Z dosaženými výsledky algoritmu jsem spokojený. Čas ukáže, jak se program osvědčí v praxi, zda bude dobře odhadovat počty studentů a zda se bude zlepšovat správnost odhadu, protože budou v databázi přibývat každým rokem další údaje.

6.3 Celkový pohled

Pro zvládnutí bakalářské práce jsem podrobněji nastudoval popisnou statistiku, abych zvládnul zpracovat hodnoty z databáze a navrhnout algoritmus pro odhad počtu studentů na rozvrhové akci. Díky vývoji programu jsem se naučil lépe programovat v jazyku Java, musel jsem nastudovat komunikaci Javy a databáze od společnosti Oracle a kompletně se naučit, jak se programují SWT aplikace, které využívají prvky nativního prostředí operačního systému.

Program jsem se snažil naprogramovat tak, aby byl uživatelsky přátelský (user-friendly). Nikde jsem nenašel přesnou definici nebo popis, kde by byly popsány zásady pro tvorbu user-friendly programu, proto jsem program tvořil podle svého cítění tak, abych zadání dodržel. Algoritmus pro odhad počtů studentů poskytuje dobré výsledky u většiny předmětů. Zadání bakalářské práce jsem dodržel a výsledek algoritmu odpovídá stanovenému cíli – správnosti odhadu počtu studentů nad 90 %.

7. Použita literatura

1) BUDÍKOVÁ, Marie, MIKOLÁŠ, Štěpán, PAVEL, Osecký. *Popisná statistika*. Brno : Masarykova univerzita, 1996. ISBN 80-210-1201-3. s. 49.

2) KUNDEROVÁ, Pavla. *Základy pravděpodobnosti a matematické statistiky : Popisná statistika*. 1. vyd. Olomouc : Univerzita Palackého, 2004. ISBN 80-244-0813-9. s. 116-128.

3) SPELL, Brett. *Java Programujeme profesionálně*. Praha : Computer Press, 2002. ISBN 80-7226-667-5. s. 1022.

4) KISZKA, Bogdan. *1001 TIPŮ A TRIKŮ PRO programování v jazyce Java*. Brno : Computer Press, 2003. ISBN 80-7226-989-5. s. 519.

5) ŽÁK, Libor. *Popisná statistika : úvod* [online]. 30. 10. 2006 [cit. 2008-04-27]. Dostupný z WWW: <http://mathonline.fme.vutbr.cz/Popisna-statistika/sc-1146-sr-1-a-139/default.aspx>.

6) ŽÁK, Libor. *Popisná statistika : zavedení pojmů* [online]. 23. 10. 2006 [cit. 2008-04-27]. Dostupný z WWW: <http://mathonline.fme.vutbr.cz/Popisna-statistika/sc-1146-sr-1-a-139/default.aspx>.

7) ŽÁK, Libor. *Statistika* [online]. 14. 11. 2006 [cit. 2008-04-27]. Dostupný z WWW: <http://mathonline.fme.vutbr.cz/Uvod-a-historie/sc-1145-sr-1-a-138/default.aspx>.

8) *Oracle stvrzuje závazek k vývojářům v jazyce Java prostřednictvím bezplatného vývojového nástroje a open source projektů : Oracle zjednodušuje vývoj složených aplikací projektem Java Server Faces* [online]. [2005] [cit. 2008-04-27]. Dostupný z WWW:
<http://www.oracle.com/global/cz/corporate/pressroom/2005/050725_jdevelop_free.html>.

9) KOUBA, Tomáš. Java.NET : Swing versus SWT [online]. 23. 5. 2004 [cit. 2008-05-03]. Dostupný z WWW:
<http://www.neo.cz/~tomas/java.net/2004/05/swing-versus-swt.html>.

10) MALÉŘ, Jakub. Eclipse RCP (Rich Client Platform) [online]. prosinec 2005 [cit. 2008-05-03]. Dostupný z WWW:
<http://nb.vse.cz/~zelenyj/it380/eseje/xmalj35/EclipseRCP.htm>.

11) KUČERA, Petr. Tvorba grafického uživatelského rozhraní v jazyce Java [online]. podzim 2005 [cit. 2008-05-03]. Dostupný z WWW: <http://is.muni.cz/th/51573/fi_m/Examples/Windows/SwtTutorial/html/index.html>.

12) Java : SWT JFace Eclipse [online]. c2003-2008 [cit. 2008-05-03]. Dostupný z WWW: <http://www.java2s.com/Code/Java/SWT-JFace-Eclipse/CatalogSWT-JFace-Eclipse.htm>.

13) Wikipedia : Otevřená encyklopedie [online]. 2002 , 26. 2. 2008 v 11:15 [cit. 2008-05-03]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Hlavn%C3%AD_strana>.