Fakulta riadenia a informatiky
Žilinská univerzita

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie OSSConf 2009

2.–5. júla 2009
Žilina, Slovensko

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
2.–5. júla 2009, Žilina, Slovensko

Organizátori:	Michal Kaukič, Peter Palúch, Helena Šamajová, Žilinská univerzita Milan Žmindák, VTS pri Žilinskej univerzite Peter Mráz, Kremnica Slavko Fedorik, Kežmarok Peter Štrba, Turčianske Teplice Miloš Šrámek, Univerzita Komenského v Bratislave Ladislav Ševčovič, Technická univerzita v Košiciach
Editori:	Michal Kaukič Miloš Šrámek Slavko Fedorik Ladislav Ševčovič
Recenzenti:	Mgr. R. Blaško, CSc. RNDr. P. Czimmermann, PhD. Ing. S. Fedorik Ing. K. Grondžák, CSc. Ing. J. Janech Mgr. M. Kaukič, CSc. doc. Ing. M. Šrámek, CSc. doc. RNDr. I. Turek, CSc.

Vydavateľ: Vedecko-technická spoločnosť pri Žilinskej univerzite
ISBN 978-80-89276-16-5

Sadzba programom pdfTEX Ladislav Ševčovič

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

Editori: Michal Kaukič
Miloš Šrámek
Slavko Fedorik
Ladislav Ševčovič

SPONZORI KONFERENCIE

PODPORILI NÁS

Obsah

ÚVOD 7

Tomáš Bača
Zásuvné moduly v nástroji UML .FRI 9

Peter Fodrek
Bugzilla pri riadení vývoja softvéru 15

Peter Fodrek
Využitie/zneužitie subversion v pedagogike
na sledovanie pribežnej prípravy počas semestra a pri hodnotení 23

Ján Janech
UML/DSM nástroj UML .FRI 29

Ján Karabáš, Jozef Siláči, Ondrej Šuch
WEB2PY FRAMEWORK na tvorbu webovských aplikácií 39

Blanka Kozáková
LMS MOODLE jako nástroj pro vzdělávání učitelů v Moravskoslezském kraji 49

Pavel Kříž, Václav Sebera
Terminálové řešení v univerzitním prostředí 55

Robert Mařík
Mathematical assistant on web – skládanka ze svobodných matematických programů . . . 61

Milan Moravčík
Multiplatformový vývoj interaktívnej aplikácie – hry Python a Pygame 69

2046
Software was just the beginning 81

Michal Páleník
OpenStreetMap – slobodná wiki mapa sveta 89

Štefan Peško
Najlacnejšie úplné párenie vo všeobecných grafoch s použitím oss nástroja GLPK 97

Miloš Šrámek
VTK – vizualizačný nástroj s otvoreným kódom 105

Andrej Trnka
Voľne dostupné LMS 117

Slavomír Tuleja
Vyučovanie fyziky na gymnáziu
s využitím vzdelávacej stratégie just-in-time teaching 123

František Zachoval
Open source skrze mikrospoločnosti 137

ABSTRAKTY 147

ÚVOD

Vážení čitatelia,

tento zborník je jedným z výstupov prvej samostatnej konferencie na Slovensku, ktorá je venovaná slobodnému a otvorenému softvéru a jeho využitiu vo vzdelávaní, vede, ale aj v ostatných oblastiach, kde sa využívajú informačné technológie.

V celosvetovom meradle, obzvlášť v štátoch EU, pozorujeme veľký nárast používania otvoreného softvéru (OSS). Dokonca aj viaceré komerčné firmy ponúkajú riešenia, kde je základom tento softvér. Veľké firmy v oblasti IT, ako Google, HP, IBM, Red Hat, Suse nielen deklarujú podporu otvoreného softvéru, ale ho aj vyvíjajú a dávajú ďalej k dispozícii vývojárskej komunite. Tým sa výrazne rozširujú možnosti využitia OSS doma, vo vzdelávaní, výskumnej činnosti a tiež aj v softvérových firmách pre nekomerčné i komerčné riešenia.

Napriek týmto faktom, OSS sa dostáva do povedomia verejnosti ťažšie, ako komerčné operačné systémy a softvér. Šíri sa hlavne v menších komunitách, „ústnym podaním" a osobnými príkladmi blízkych osôb. Existuje mnoho učiteľov, študentov, výskumníkov alebo prosto nadšencov, ktorí OSS vo svojej práci využívajú, nahrádzajú či dopĺňajú ním drahé komerčné riešenia a pritom získavajú cenné skúsenosti, z ktorých by mohli mať úžitok aj ostatní. Problémom je, ako tieto skúsenosti odovzdať. Jednou z príležitostí môže byť aj naša konferencia.

Sme presvedčení, že otvorený softvér sa môže veľmi dobre uplatniť v oblasti výučby a výskumu na všetkých stupňoch škôl. Zatiaľ čo niektoré softvérové produkty s nedostupným zdrojovým kódom sú široko propagované zjavnou i skrytou reklamou, otvorený softvér a jeho možnosti sú oveľa menej známe. Je to škoda, pretože otvorený softvér v tejto oblasti poskytuje plnohodnotnú náhradu viacerých finančne náročných softvérových produktov (kancelárske balíky, matematický softvér, sieťové služby a administrácia, programy na výučbu rôznych predmetov na základných a stredných školách). Viaceré z týchto otvorených alternatív sú prezentované v materiáloch konferencie (konferenčné DVD a tento zborník).

Výhody otvoreného softvéru sa neprejavujú len v ekonomickej oblasti, ale majú aj významný dopad na myslenie žiakov, študentov i učiteľov. Podporujú tvorivý prístup k softvéru, ktorý sa hĺbavému používateľovi už nejaví ako čierna skrinka ale umožňuje mu prispôsobiť si jeho funkcionalitu vlastným potrebám. Zároveň majú používatelia veľký vplyv na jeho vývoj, pretože ich ohlasy, želania, kritika a chybové hlásenia pomáhajú vývojárom pri ďalšom zdokonaľovaní a šírení príslušného softvérového systému.

Priamymi podnetmi na usporiadanie tohto podujatia boli úspešné dva ročníky špecializovanej sekcie *Otvorený softvér vo vede a vzdelávaní*, ktorú sme organizovali v rámci medzinárodnej konferencie Aplimat v Bratislave v r. 2007 a 2008. Ďalším podujatím, ktoré ukázalo záujem hlavne učiteľov a študentov o používanie otvoreného softvéru, bol *Víkend s Linuxom* v Kremnici v júni 2008.

Podujatie *Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach* chce pokračovať v dobrej tradícii vytvorenej predchádzajúcimi podujatiami. Nie je to len konferencia v klasickom, akademickom ponímaní, ale hlavne neformálne stretnutie ľudí, ktorých spája záujem

o otvorený softvér (jeho tvorbu, využitie, komplexnejšie riešenia na základe existujúceho softvéru). Popri recenzovaných príspevkoch, ktoré sú publikované v tomto zborníku, na podujatí budú predvedené ukážky riešení, kde je základom otvorený softvér. Stručné abstrakty týchto ukážok a prezentácií sú uvedené v závere tohto zborníka.

Všetky príspevky, uvedené v tomto zborníku, ako aj prezentácie, ktoré na konferencii odznejú, budú voľne prístupné na stránke podujatia, `http://frcatel.fri.uniza.sk/OSS09`. Okrem toho, na tejto stránke nájdete aj odkaz na konferenčné DVD (DVD médium dostane každý účastník konferencie), ktoré okrem spomínaných materiálov konferencie obsahuje distribúciu GNU/Linux Ubuntu 9.04 s ďalším pridaným softvérom, ktorého sa budú týkať príspevky konferencie, s príkladmi a videotutoriálmi.

Táto konferencia je jednou z aktivít na propagáciu otvoreného softvéru. Sme toho názoru, že čím skôr sa používateľ počítača s otvoreným a slobodným softvérom zoznámi, tým lepšie bude vedieť oceniť jeho prednosti, z ktorých bude neskôr profitovať, či už profesionálne alebo v súkromí. Zároveň sa rozširuje jeho všeobecný rozhľad, keď sa odváži nakuknúť za oponu, mimo javiska, ktoré mu prezentuje svet reklamy a obchodných záujmov softvérových gigantov. Na podporu a propagáciu alternatívneho vyučovania informatiky a ostatných predmetov na základných, stredných a vysokých školách prostredníctvom bezplatného otvoreného softvéru slúži tiež projekt „Slobodný a otvorený softvér pre školy“ s domovskou stránkou `http://sospreskoly.org`. Cieľom projektu je vytvoriť komunitu záujemcov a používateľov otvoreného softvéru na školách všetkých úrovní. Je pokusom o spojenie tých, ktorí vedia ako na to a rovnako aj tých, ktorí si uvedomujú možnosti, ktoré otvorený softvér pre školstvo a spoločnosť poskytuje, ale nevedia ako a kde začať.

Nič sa neurobí samo od seba, takže tento projekt je príležitosťou aj pre vás. Môžeme postupne budovať vzorové učebne s voľne prístupným edukačným softvérom, pracovať na jeho prípadných úpravách a lokalizácii, vytvárať návody a príručky pre správcov učebne a používateľov, aby sme čoraz viac cítili, že v našich snahách nie sme osamotení. Súčasťou snáh projektu *SOS pre školy* je aj presadzovanie (na úrovni vládnej i regionálnych samospráv) otvoreného softvéru v použití pre školské administratívne a ekonomické účely.

Tento zborník, materiály konferencie, aj samotný fakt, že sa toto podujatie mohlo uskutočniť, je kolektívnym dielom organizátorov, editorov, autorov príspevkov a prezentácií, recenzentov i všetkých účastníkov. Myšlienka samostatnej OSS konferencie sa mohla realizovať len vďaka príkladnej spolupráci a dobre odvedenej práci všetkých zainteresovaných. Ďakujeme im a dúfame, že sa aj v budúcnosti budeme stretávať na podobných podujatiach a že nás bude čoraz viac, aby naše hlasy nezanikli v lomoze každodenného zhonu a agresívnej reklamy softvérových firiem.

Vďaka sponzorom, HP Slovensko a Slovenskej informatickej spoločnosti sa mohlo uskutočniť vydanie tohto zborníka a podporili sme účasť viacerých účastníkov z radov učiteľov čiastočným uhradením nákladov.

Organizačný výbor OSSConf 2009

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

ZÁSUVNÉ MODULY V NÁSTROJI UML .FRI

BAČA, Tomáš, (SK)

1 Úvod

Aplikácia UML .FRI je vizuálny editor diagramov, aké bežne používajú študenti a pracovníci v oblasti IT. Vyvíjajú ho študenti inžinierskeho štúdia na Fakulte riadenia a informatiky Žilinskej univerzity v rámci svojej projektovej výučby.

Nástroj má síce v názve *UML*, radíme ho však medzi *DSM*[1] modelovacie nástroje. To znamená, že nie je zameraný na jeden modelovací jazyk, ale obsahuje mechanizmus, ktorý interpretuje *metamodel*[2]. Preto je v ňom možné modelovanie napríklad v UML, vývojových diagramoch, DFD, vytváranie grafov či schém zapojenia počítačov do počítačovej siete. Okrem toho, pridanie nového modelovacieho jazyka spočíva iba v pomerne jednoduchom vytvorení nového metamodelu.

Pre profesionálne modelovacie nástroje je charakteristické, že okrem modelovania samotného, obsahujú aj pokročilejšie funkcie. Napríklad umožňujú generovanie zdrojových kódov na základe UML diagramu tried alebo automatické usporiadanie prvkov diagramu podľa určitých pravidiel. Väčšina týchto funkcií je však špecifická pre jeden typ alebo malú skupinu typov diagramov. Do spomínaných profesionálnych nástrojov mohli byť teda zaradené relatívne ľahko hlavne preto, že nástroje samotné sú orientované iba na vybrané modelovacie jazyky.

Tím, starajúci sa o vývoj UML .FRI, považuje takéto funkcie za potrebné a rád by ich do svojho nástroja zaradil. Ak sú však závislé na modelovacom jazyku, musia byť dodávané spoločne s príslušným metamodelom a teda vo forme *zásuvných modulov*.[3]

[1] Domain Specific Modeling

[2] Model definujúci model, alebo tiež gramatika modelovacieho jazyka

[3] Zásuvný modul je počítačový program, ktorý spolupracuje s hlavnou (hostiteľskou) aplikáciou. Poskytuje jej konkrétnu a obyčajne aj veľmi špecifickú funkcionalitu „na požiadanie používateľa“.

Z toho dôvodu som si pre svoju diplomovú prácu vybral tému *Návrh a tvorba systému zásuvných modulov pre nástroj UML .FRI*. [4] V rámci jej riešenia som vytvoril prvú verziu, ktorú budem postupne rozširovať. V článku opisujem princípy, na ktorých je tento systém postavený a možnosti, ktoré to do budúcnosti prináša.

2 Požiadavky na systém podpory zásuvných modulov

Aplikácia UML .FRI je vyvíjaná v programovacom jazyku Python, pre grafické rozhranie používa knižnicu GTK+, informácie ukladá pomocou XML. Všetky tieto technológie boli vybrané tak, aby aplikácia mohla fungovať na rôznych operačných systémoch a hardvérových platformách. Z toho vyplýva najpodstatnejšia podmienka, ktorú musí aj nový systém zásuvných modulov spĺňať – musí fungovať rovnako univerzálne ako aplikácia samotná.

Aplikácia je vyvíjaná ako otvorený projekt a najmä oddelenie metamodelov od aplikácie umožňuje pomerne jednoduché rozširovanie zo strany používateľov. Rovnako, zásuvné moduly sú zamerané na možnosť rozšírenia funkcionality programátormi nezávislými na vývojovom tíme UML .FRI. Nedostatočne otestovaný zásuvný modul však môže predstavovať riziko pre stabilitu samotnej aplikácie. Systém zásuvných modulov musí byť preto odolný voči chybám zásuvných modulov.

Jazyk Python má mnoho nesporných výhod voči iným programovacím jazykom a preto je aj použitý pre naprogramovanie aplikácie UML .FRI. Napriek tomu však nie je medzi programátormi príliš rozšírený. Ak by mal byť vývoj zásuvných modulov obmedzený iba na Python, pravdepodobne by to odradilo veľmi veľkú časť potencionálnych tvorcov a to by bola škoda. Musí byť preto umožnené pripájať k aplikácii zásuvné moduly naprogramované v rôznych programovacích jazykoch.

3 Implementácia systému podpory zásuvných modulov

V záujme uspokojenia všetkých požiadavok a inšpirovaný novým webovým prehliadačom Google Chrome, som sa rozhodol, že každý zásuvný modul bude vykonávaný v samostatnom procese. Dôjde tak k dostatočnej izolácii aplikácie od chýb, ktoré by mohli v zásuvných moduloch vzniknúť. Navyše, univerzálny systém medziprocesovej komunikácie umožní implementáciu zásuvných modulov principiálne v ľubovoľnom programovacom jazyku.

V prvej fáze vývoja som navrhol, že zásuvný modul sa bude k aplikácii pripájať pomocou TCP/IP *socketu*. Je to pomerne jednoducho implementovateľný spôsob. Má však jednu závažnú nevýhodu. Niektorí používatelia môžu mať blokovanú IP komunikáciu na väčšine portov, čím by znemožnili fungovanie tohto systému. Do druhej fázy preto plánujem implementovať komunikáciu aj pomocou rúr (*pipes*).

[4] Systém umožňujúci pripájanie zásuvných modulov k aplikácii a vytvárajúci komunikačné rozhranie medzi aplikáciou a zásuvnými modulmi.

Pre zachovanie maximálnej prenositeľnoti systému medzi platformami ale tiež medzi programovacími jazykmi, komunikačný protokol je kódovaný textovo. Konkrétne, vychádza zo špecifikácie *RFC 2822* [8]. Formát správ sa preto podobá na protokoly známe z internetovej komunikácie (*HTTP*, *SIP*, *SMTP*).

Správy možno rozdeliť na tri typy:

- požiadavka – zaslaná zásuvným modulom,
- odpoveď – zaslaná aplikáciou ako reakcia na požiadavku zásuvného modulu,
- notifikácia – zaslaná aplikáciou v prípade nejakej udalosti

Zásuvný modul môže zasielať požiadavky na tri rôzne časti aplikácie:

- zmena stavu *GUI*[5] aplikácie – aby si pridal svoje vlastné položky do menu aplikácie,
- získanie informácie o aktuálne načítanom metamodeli,
- vzdialené volanie metódy nad objektmi samotného modelu.

Knižničné rozhranie, ktoré môžu použiť programátori zásuvných modulov, je navrhnuté tak, aby abstrahovalo od komunikačného kanála medzi aplikáciou a zásuvným modulom. Celé je to spravené tak, aby mal programátor dojem, že pracuje s objektmi aplikácie. V skutočnosti tu však dochádza k vzdialenému volaniu metód.

4 Možnosti použitia systému zásuvných modulov

Pre lepšiu predstavu ešte raz pripomeniem základné schopnosti aplikácie UML .FRI. Aplikácia samotná umožňuje vytváranie diagramov, zložených z elementov a spojení. Obsahuje iba všeobecné funkcie pre prácu s týmito diagramami. Sú to napríklad preskupovanie elementov, ukladanie projektu, či tlačenie diagramu na tlačiarni.

Pokročilejšie funkcie, ako napríklad generovanie zdrojových kódov z UML, však neobsahuje a ani obsahovať nebude. Je to najmä preto, lebo takéto funkcie sú špecifické pre konkrétny modelovací jazyk, čiže pre konkrétny metamodel. Metamodely sú však k aplikácii dodávané samostatne a navyše si ich môžu vytvárať aj samotní používatelia.

Zásuvné moduly preto predstavujú vhodný spôsob, ako rozšíriť funkcionalitu aplikácie o rôzne špecifické algoritmy a zachovať pritom jadro aplikácie malé a univerzálne. Konkrétne príklady použitia zásuvných modulov v aplikácii UML .FRI sú:

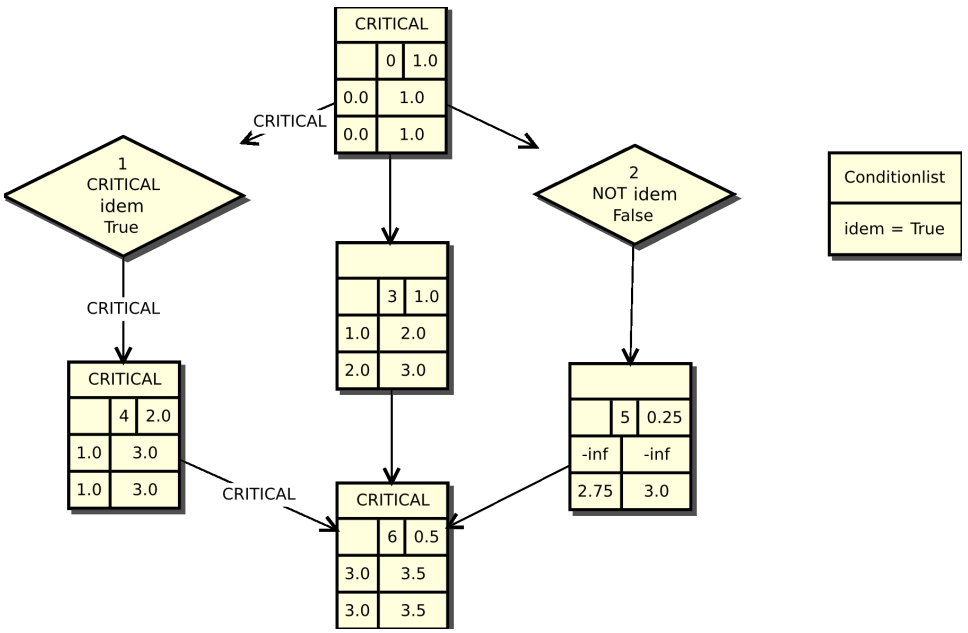
- generovanie zdrojových kódov z diagramov,
- spätné vytváranie diagramov zo zdrojových kódov,
- transformácie model2model,
- implementácia a vizualizácia rôznych algoritmov z teórie grafov,

[5] Graphical User Interface – grafické používateľské rozhranie.

- generovanie metamodelov,
- automatické usporiadanie prvkov diagramu,
- procesne orientovaná simulácia,
- a iné.

Pre úplnosť treba povedať, že nie všetky návrhy sú realizovateľné so súčasnou verziou UML .FRI. V nástroji chýba napríklad ešte podpora pre animáciu, ktorá by bola potrebná pre potreby simulácie.

5 Demonštračný zásuvný modul



Obr. 1: Príklad sieťového grafu s podmienenými vetvami a vyznačenou kritickou cestou

Napriek tomu, že na realizáciu niektorých nápadov ešte nie je aplikácia pripravená, iné môžu začať vznikať už teraz. Jeden z nich som implementoval ako súčasť svojej diplomovej práce, aby som ním demonštroval funkčnosť svojho riešenia. Zásuvný modul slúži na hľadanie kritickej cesty v sieťovom grafe, ktorý obsahuje podmienené vetvy. Teória ohľadom tejto problematiky je popísaná v [3, 4, 6, 7] a preto sa jej venovať nebudem. Popíšem však v niekoľkých krokoch, ako prebiehal samotný vývoj zásuvného modulu.

V prvom rade bolo potrebné vytvoriť nový metamodel. V tomto prípade šlo o definovanie metamodelu s jedným typom diagramu – sieťovým grafom. Sieťový diagram obsahuje tri typy

elementov (činnosť, podmienka, zoznam hodnôt podmienok) a jeden typ spojenia (náväznosť činností). Pre každý typ elementu a spojenia je potrebné vytvoriť definíciu logickej štruktúry dát, ktorá nesie (atribúty), vzhľad a okrem toho treba vymenovať, ktorými typmi spojenia možno spájať jednotlivé typy elementov.

Následne na to som mohol začať programovať samotný zásuvný modul. Zvolil som preň štruktúru triedy, ktorá obsluhuje udalosti vznikajúce v GUI. To znamená, že metóda, ktorá je volaná pri inicializácii zásuvného modulu, pridá do menu aplikácie svoju položku a pre udalosť jej aktivácie zaregistruje ďalšiu svoju metódu. Zásuvný modul potom prechádza do stavu čakania. V prípade, že používateľ aktivuje túto položku, bude vyvolaná obslužná metóda.

Obslužná metóda sa najskôr musí presvedčiť, či je v aplikácii práve otvorený sieťový graf, v opačnom prípade nemá totiž význam niečo počítať. Následne na to si načíta celý diagram a interpretuje ho do svojej internej štruktúry. Potom sa pokúsi o monotónne očíslovanie vrcholov a výpočet najskorších a najneskorších možných začiatkov a koncov činností. Napokon označí vrcholy so zhodnou hodnotou najskoršieho aj najneskoršieho možného začiatku za kritické činnosti, a hrany, ktoré ich prepájajú za kritické hrany. Na záver sú výsledky zobrazené v aplikácii.

6 Záver

Úlohou práce bolo umožniť rozširovanie aplikácie UML .FRI pomocou zásuvných modulov. Keď vedúci mojej diplomovej práce vymyslel túto tému, očakával, že na jej základe bude môcť vypísať celú sériu ďalších záverečných prác s rôznou tématikou. Dá sa povedať, že každý z bodov štvrtej kapitoly predstavuje námet na záverečnú prácu minimálne jedného študenta. A treba povedať, že nové nápady sa postupne vynárajú aj teraz.

Moja práca sa však ešte stále nekončí. Súbežne vytváral svoju diplomovú prácu aj môj kolega na tému „Návrh a implementácia systému Undo/Redo[6] do aplikácie UML .FRI“. Obaja sme vyvíjali svoje vetvy oddelene od hlavnej aj od seba navzájom. Našou spoločnou úlohou v najbližšom čase bude zaradenie týchto nových systémov do hlavnej vetvy a ich vzájomné prepojenie. Spoločným termínom celého tímu UML .FRI na vydanie novej verzie je september 2009. Vtedy začne nasledujúci akademický rok a my všetci dúfame, že si v ňom UML .FRI získa pevné postavenie medzi učiteľmi aj študentmi Fakulty riadenia a informatiky.

Poďakovanie

Táto práca vznikla s podporou grantovej agentúry KEGA v rámci riešenia projektu 3/2158/04 „Využitie open source softvéru vo výučbe na vysokých školách“.

[6]Operácia *Undo* znamená vrátenie predchádzajúcej používateľovej akcie. Operácia *Redo* zruší predchádzajúcu *Undo* operáciu.

Literatúra

[1] BAČA, T.: *Návrh a tvorba systému zásuvných modulov pre nástroj UML .FRI*. 2009

[2] BAČA, T. – JURÍČEK, M. – ODLEVÁK, P. *Case tool development* Journal of Information, Control and Management Systems 2008. vol.6, ISSN 1336-1716.

[3] BACHRATÝ, H. – SADLOŇ, Ľ.: *Modifikácia zovšeobecnených sieťových grafov*. 2005. nepublikované.

[4] BACHRATÝ, H. – SADLOŇ, Ľ.: *Výpočet prevádzkových intervalov pomocou sieťových grafov*. INFOTRANS 2005. Pardubice : DPJF Pardubice, september 2005. ISBN 80-7194-792-X

[5] JANECH, J. – BAČA, T. – ODLEVÁK, P – a i.: *Projektová dokumentácia*. 2008. nepublikované.

[6] LUKÁČ, M.: Softvérový nástroj na návrh a výpočty sieťových grafov s podmienkovými hranami. 2005.

[7] PALÚCH, S.: *Skriptá z teórie grafov*. 2001. Dostupné online: `http://frcatel.fri.uniza.sk/users/paluch/grafy-pdf-pics.zip`

[8] RFC 2822 Internet Message Format. 2001. Dostupné online: `http://www.ietf.org/rfc/rfc2822.txt`

Kontaktná adresa

Tomáš Bača,
peter.fodrek@stuba.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

BUGZILLA PRI RIADENÍ VÝVOJA SOFTVÉRU

FODREK, Peter, (SK)

1 Úvod

Zúčastnil som sa na akcii „Dni príležitostí 2009“, kde sa stretali firmy so študentami a pedagógmi a ponúkali pracovné miesta. V diskusiách s firmami mi bolo povedané alebo potvrdené, že jedným z najväčších nedostatkov absolventov je práca v tímoch. Časť problémov riešia Source Code Management (SCM) systémy, ktorých vedľajšie efekty sme sa rozhodli použiť aj v hodnotení. SCM, a teda zdieľaním kódov, sa zaoberám v inom článku. Druhá časť problémov je organizácia práce v tíme. Na to slúžia programy hlásenia chýb a plánovania úloh. Jedným z programov na riešenie tímovej synchronizácie je OSS systém Bugzilla. Samotný názov napovedá pôvod systému. Bol totiž vytvorený „na lovenie vší“ (bugov) v programoch vytvorených Mozilla Foundation (odtiaľ koniec názvu).

2 Inštalácia

Bugzilla je súbor Perl skriptov. Z toho vyplýva, že rozhranie je primárne webové, hoci sa môže použiť aj lokálne. Na svoju činnosť potrebuje Bugzilla databázu Oracle, MySQL alebo PostgreSQL a k tomu príslušné Perl moduly, ktoré sa spúšťajú ako CGI skripty. Inštalačný skript s parametrom `--check-modules`, zistí potrebné a voliteľné moduly Perl-u potrebné na beh Bugzilly. Napriek tomu, že východzie nastavenie je pre MySQL, zvolil som PostgreSQL, lebo mi pripadala menšia a ľahšie administrovateľná ako MySQL a navyše som ju musel na BSD server s menom fuzzy inštalovať lokálne, lebo tam nebola inštalovaná

a s inštaláciou MySQL som nemal skúsenosti. Po inštalovaní modulov Perl je, pre inú databázu ako MySQL, nutné prepísať položku Driver v konfiguračnom súbore na driver pre nainštalovanú databázu, a to taký, ktorý sme doinštalovali ako modul Perlu. Pre PostgreSQL je to ovládač „Pg“ s dodržaním veľkého písmena P. Problém môže nastať ak nainštalujeme len jeden ovládač k databáze a chceme použiť inú databázu. Inštalačný skript s vyššie uvedeným parametrom prestane zobrazovať zoznam možných ovládačov k databázam, ak je v systéme nainštalovaný aspoň jeden z troch ovládačov pre databázy v Perle. Potom treba nastaviť databázu, a keďže na serveri Fuzzy nemám práva administrátora, musel som urobiť lokálnu inštaláciu PostgreSQL.

Z rovnakého dôvodu som použil lokálnu inštaláciu Perlu, keďže ten pôvodný mi nedovoľoval doinštalovať moduly a ovládače databáz bez práv užívateľa root (Unix administrátor). Lokálna inštalácia navyše zvyšuje bezpečnosť systému, ktorá je na fuzzy až paranoidná, čo ale nie je na škodu. Kvôli tomu treba použiť shell skript s využitím editora sed na zmenu cesty k interpereteru jazyka Perl. Po nastavení PostgreSQL treba vytvoriť databázu s menom bugs, ktoré je prednastavené v konfiguračnom súbore Bugzilly. Ak použijeme iné meno, treba prepísať konfiguračný súbor. Ten musíme aj tak prepísať, lebo musíme zadať prihlasovacie meno a heslo k databáze. Potom musíme ešte nastaviť port PostgreSQL, ak nie je použitý prednastavený, a to ako v nastavení PostgreSQL tak v nastavení Bugzilly. Tesne pred záverom musíme spustiť PostgreSQL démona(v MS Windows je obdoba Service, v DOS-e je to TSR program).

Na záver musíme spustiť HTTP démona, najlepšie Apache. Tu vznikne na dobre zabezpečených systémoch ďalší problém: tieto systémy neumožňujú použiť CGI skripty v domovských adresároch užívateľov. Preto je možné buď inštalovať lokálny Apache, alebo umiestniť skripty do určeného adresára, čo ale zníži bezpečnosť. Preto sme sa rozhodli pre lokálny Apache. Ten sme nakonfigurovali tak, že je možné pristupovať k adresáru, kde sme rozbalili inštaláciu Bugzilla. Port bol nastavený na 5000 a spustili sme Apache. Potom už len stačilo, v adresári, kde bola rozbalená Bugzilla, skontrolovať nastavenie hesiel a databázy. Potom je nutné spustiť inštalačný skript Bugzilla bez parametrov. Tým sa upraví nastavenie a práva k skriptom. Od toho okamihu možno Bugzillu použiť z lokálneho počítača a na nedokonale zabezpečených systémoch aj z iných počítačov. Na test sme použili prehliadač lynx z BSD.

3 Proces riadenia projektov ako paralelné programovanie

Na server s menom Fuzzy sa nie je možné pripojiť cez porty obsluhované užívateľskými démonami. Takým démonom je aj náš Apache. Preto sme museli vytvoriť SSH tunel. Pre programátora s akýmikoľvek skúsenosťami s Unixom, nie je nič ľahšie, ako napísať tunelový klient a tunelový server, skompilovať a spustiť ich. To ale platí len za predpokladu, že chce pracovať na jednoprocesorovom počítači s jednojadrovým procesorom. Čas takých programov je ale spočítaný. Tvrdí to aj Intel, keďže v rámci konferencie Supercomputing 08, konanej v novembri 2008, poriadal diskusiu s veľavravným názvom: „There Is No More

Sequential Programming. Why Are We Still Teaching It?“ [1, 2]. V diskusii s Intelom to potvrdili aj zástupcovia spoločností AMD, Sun, IBM a nVidia. Pre efektívne paralelné programovanie však platí Amdahlov zákon [3]:

$$k = \frac{1}{1 - f + \frac{f}{n} + H(n)}, \tag{1}$$

kde k je pomer rýchlosti paralelnej aplikácie k rýchlosti sekvenčnej aplikácie,
f je pomer časti kódu, ktoré môžu bežať paralelne, a teda nie je dané poradie behu úloh k celkovej veľkosti kódu resp. pomer časov behu týchto kódov,
n je menšie z čísel N_1 a N_2 kde N_1 je počet jadier počítača, na ktorom beží kód a N_2 je počet procesov alebo vlákien aplikácie,
$H(n)$ je réžia správy procesov a vlákien pre N_2 procesov na N_1 jadrách. Pre ideálny operačný systém platí $H(n) = 0$, ak $N_1 > N_2$.

Pre ideálny aj reálny operačný systém platí (rovnica pre $N_1 = N_2$, v reálnom systéme platí aj pre $N_1 > N_2$)

$H(n) \to 0$, ak $N_1 = N_2$,

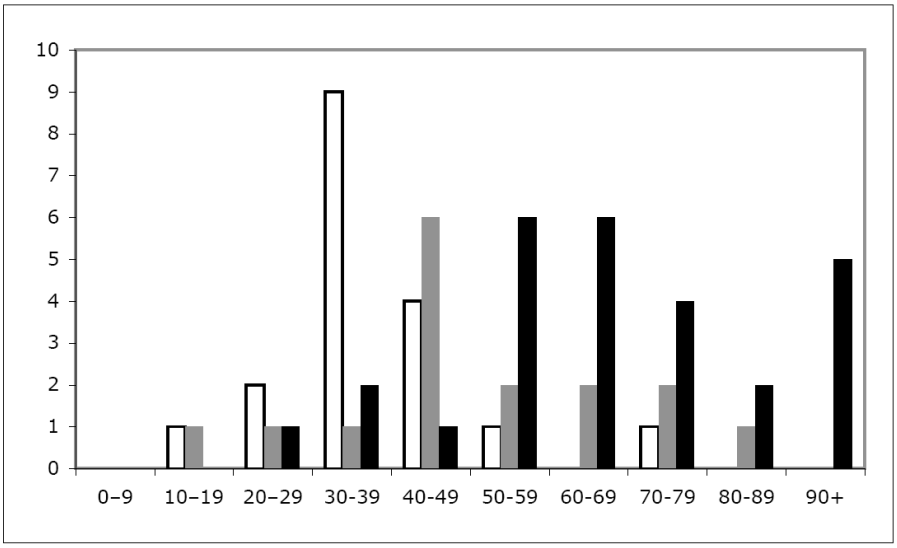
$0 > H(n) \leqq f$, ak $N_1 < N_2$.

Kedže klient bežiaci na serveri a server bežiaci u klienta majú kódy podľa prídavných súborov BugzillaKlient.c a BugzillaServer.c, ktoré majú f=1 pre $N_2 = 1$ v prípade klienta a $f = 1$ pre $N_2 = 2$ v prípade servera. V tom prípade je pre server $k = 2$ na dvojjadrovom procesore, ale $1/2 \leqq k < 1$ na jednojadrovom procesore. Preto výrazne neodporúčam použiť tento server na počítačoch s menej ako 2 jadrami, keďže aj webový prehliadač je proces a aj Apache je najmenej jeden proces. Ide o školský jednoduchý príklad. *Bežný zložitý softvér nikdy nemôže dosiahnuť $f = 1$, lebo vždy existujú časti kódu, ktoré nemôžu bežať paralelne*.

Obdobne ako s paralelným kódom je to aj s členmi programátorského tímu. Platí paralela, že členovia tímu sú procesy a súbory sú jadrá CPU. Dehnadi [4] rozdelil študentov programovania do troch skupín:

- 44% študentov použilo na riešenie všetkých, alebo skoro všetkých úloh rovnaký spôsob uvažovania *bez ohľadu na to či správny alebo nesprávny*. Nazvime ich *ľudia s konzistentným uvažovaním.*
- 39% študentov použilo na riešenie rozdielnych úloh rozdielny spôsob uvažovania. Nazvime ich *ľudia s nekonzistentným uvažovaním.*
- Zostávajucich 8% študentov odmietlo odpovedať na všetky alebo skoro všetky otázky resp. odpovedalo neviem. Nazvime ich *ľudia bez vyhraneného uvažovania.*

Potom dal rôznym skupinám odhadnúť, ktorá zo skupín bude najúspešnejšia. Programátori a iní zamestnanci v IT, takmer všetci, *predpovedali úspech skupiny bez vyhraneného*



Obr. 1: Úspešnosť typov študentov – biela = nekonzistentné myslenie na začiatku aj konci semestra, sivá = nekonzistentné alebo nevyhranené myslenie na začiatku a na konzistentné konci semestra, čierna = konzistentné myslenie na začiatku aj konci semestra, na osi *X* sú získané percentá bodov, na osi *Y* je množstvo študentov

uvažovania. Neprogramujúci humanitní vedci, matematici a historici, takmer všetci, *predpovedali úspech skupiny s nekonzistentným uvažovaním*.

Reálne výsledky sú na Obr. 1. S nimi korešpondujú aj iné výsledky. 57% profesionálov v IT sú strašní individualisti [5]. Aby toho nebolo dosť, tak len *spôsobov programovania je asi* $2,6 \cdot 10^{42}$.

Riadenie tak rozdielneho tímu, sa teda podobá plánovaniu úloh na „hybridnom multijadre“. Jediný použiteľný koncept plánovania úloh na procesore s nerovnakými jadrami t. j. „hybridnom multijadre“ je koncept koprocesorov t. j. hlavné jadro prideľuje úlohy ostatným jadrám podľa toho, ktoré sú voľné a na čo sú špecializované. Ak neexistuje voľné jadro, tak úlohu počíta hlavné jadro *(koncept Commodore C64)*. Modifikácia môže byť v tom, že koprocesor „číha“ na jemu vhodnú úlohu a ak ju nájde oznámi hlavnému jadru, že ju bude počítať. *Oba tieto koncepty podporuje, ako metodiky riadenia práve Bugzilla.*

Reálne je to tak, že tím je rozdelený na podtímy, ktorým hlavný vývojár rozdáva úlohy. Každý tím má svojho podvedúceho. Títo podvedúci vedia stanoviť aj časový odhad riešenia problému alebo pridania novej vlastnosti do svojho podprojektu. V princípe nič nebráni, aby každý programátor bol spoluvedúcim podprojektu a teda podvedúcim. Na čo sa ale zabúda, sú testeri. Tí by mali tiež byť členmi tímu a nemal by medzi nimi byť programátor, s výnimkou, ak ide o bezpečnostné aspekty t. j. podprojekt je bezpečnostnou časťou projektu. Ale aj vtedy by mal byť aspoň jeden z testerov laik až človek, čo počítač vidí prvý krát (tzv. BFU). Toto má vyriešené jedine Microsoft. Napr. pri Windows 7 sú tímy podieľajúce sa na vývoji zložené z n vývojárov, n testerov a $n/2$ programových manažérov.

Veľké OpenSource projekty nemôžu zohnať testerov z radov BFU, a teda nedbajú na použiteľnosť pre ľudí s nekonzistentným myslením. Toto sa ale dá riešiť tým, že BFU používajúci OSS, budú chyby hlásiť práve cez systém typu Bugzilla. Problém je, že OSS používajú BFU vo veľmi malej miere a to práve kvôli netestovaniu „použiteľnosti pre BFU“. To znamená jediné: OSS potrebuje podporu testerov vo veľkých firmách ťažiacich z existencie OSS, alebo ešte lepšie zvýšiť počet BFU používajúcich OSS administratívnym nariadením. O to prvé sa snažia hlavne Sun(Oracle) a IBM. O to druhé sa snaží Riaditeľstvo pre informatiku Európskej komisie a Európsky parlament. To, čo nemá Microsoft vyriešené je technická stránka veci, aby ulahodil testerom používa veľké tímy o veľkosti 100 ľudí. V českom preklade knihy *Programmers at work* sa píše doslova:

„Došel jste k nějakému konkrétnímu stylu práce, který je pro vaše účely nejproduktivnější? C. Wayne Ratliff: Pracuji buď sám, nebo s velice malou skupinou lidí. *Jakmile má tým víc než šest členů, nastane chaos*. Ted Glasser, který má řadu patentů, záznam v *Kdo je kdo* a patří mezi důležitější postavy počítačového oboru, jednou prohlásil, že *Největší tým, který ještě zvládne řídit, je tým, který může vzít na pizzu ve volkswagenu*. Od té doby změnil písničku – *teď už je to obyčejné americké auto*. Vřele s ním souhlasím“. Práve nevyhnutnosť mať mikrotímy, aby sa dal písať technicky kvalitný kód, má vyriešené vynikajúco Open source.

Preto v Microsofte nevedú tímy programátorov programátori a to dáva OSS lepšiu kompatibilitu s inými riešeniami a/alebo kvalitnejšie technické prevedenie.

4 Aplikácia v systéme Bugzilla

Ako sme uviedli, Bugzilla má aj webové rozhranie a po spustení klienta, servera z SSH tunelu, ktoré sme skompilovali, sa k bugzille dostaneme cez URL `http://localhost:5000/` ak port servera je 5000 a server má spustený Web server, SQL a verejný *kľúč* k súkromnému kľúču v súbore web_rsa je na serveri v súbore authorized_keys pre tcpcli01, pozri Listing1.

Listing 1

```
-bash-2.05b$ more .ssh/authorized_keys

command="/home/fodrek/tunel/tcpcli01"ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAmpqvwj0UDJScv2mRjELmOMvMoeZxCn1Y35Bs80se5o6
0W2I8Hxv3ZEItaIQv/yAmshwMOnpm7udNYjRy9Ba91rIauD4OkJEdPrIaTlYf9FPUne+YBa
uXMZ6EOdvTr8gIMBfayuek3Qp1uEGsbvC54KSBA LtsdjgXuhAGb8ScCe8=peto@fodrek

command="/home/fodrek/tunel/tcpcli01"ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAt4EcD6afQVzC9wOWq/5FlkJjI+M9YAPbdBOeAROAukN
dJvFvCqpQVz8GbRQMgZ+exhQUx.......usTzn9YIf8a2GZykleVvHyaFfQPDEE6Y1cuK+D
OMdIj0rb1IuapyqG7xva0lz1Glo+2z83Q==peto@fodrek
```

Ako vidieť jeden užívateľ môže mať viac a rôzne dlhých a tým rôzne bezpečných kľúčov. Riadok kódu v prílohe server

```
execl("/usr/bin/ssh","ssh","-T","-i","/home/peto/.ssh/web_rsa",
"fodrek@pid.kasr.elf.stuba.sk",ppid,NULL);
```

má hrubo vytlačenú cestu k súkromnému kľúču bez hesla a modrým užívateľa pod ktorým sa na webserver prihlásime. Obrázok 2 a Obr. 3 je web rozhranie systému Bugzilla ako aj Eclipse rozhranie pre Bugzilla v doplnku na prácu v tíme Mylyn Bugzilla Connector. Login name je e-mail, kam sa posielajú emaily pre Vás ak to povolíte.

Do elektronickej verzie článku (je dostupný na webovej stránke konferencie `http://frcatel.fri.uniza.sk/OSS09/Materialy`) sú vložené obrázky, na ktorých je vidieť používanie systému Bugzilla v prostredí Eclipse. Pre nečitateľnosť a z priestorových dôvodov tieto obrázky v tlačenej verzii článku neuvádzame.

1. Nastavenie Bugzilla Servera.
2. Vytvorenie nového riadiaceho reportu.
3. Vyplnenie reportu.
4. Odoslanie reportu.
5. Zmena stavu reportu – t. j. napr. z vytvorený na pridelený vývojárovi X, alebo na vyriešený.
6. Využitie vyhľadávania v rámci reportov napr. tie, ktoré mám riešiť.
7. Zobrazenie nájdeného reportu v druhej záložke.

Ďalšie obrázky sú snímky obrazoviek pre webové rozhranie Bugzilly:

1. Prihlasovacia obrazovka.
2. Hlavná obrazovka užívateľa.
3. Výber projektu (produktu).
4. Okno s vlastnosťami produktu.
5. Prístup k reportom produktu.

5 Záver

Ukázali sme vhodnosť použitia stratégie použitej v Bugzille na riadenie projektu, s tým, že podtímy riešia „Komponenty Produktov“ a ukázali sme ako sa používa Bugzilla v Eclipse a prehliadači.

Literatúra

[1] STEINBERG, P.: *Sequential programming is no more. Let's teach parallel programming*. [online], Santa Clara : Intel Software Network, 2008, dostupné na URL adrese: `http://software.intel.com/en-us/blogs/2008/11/12/sequential-programming-is-no-more-lets-teach-parallel-programming/`

[2] MANCHESTER, P.: *Intel rallies rivals on parallel programming education*. [online], London : Situation Publishing, Ltd., 2008, dostupné na URL adrese: `http://www.theregister.co.uk/2008/11/13/intel_parallel_programming_priority`

[3] VADKERTI, L.: Dvořák,V.: V*ývoj paralelních aplikací s Intel threading tools*. [online], Brno : diplomová práca – Vysoké učení technické v Brne, 2007, dostupné na URL adrese: `http://www.fit.vutbr.cz/study/DP/rpfile.php?id=539`

[4] DEHNADI, S. – BORNAT, R: *The camel has two humps (working title)*. [online], London : Middlesex University, 2006, dostupné na URL adrese: `http://www.cs.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf`

[5] The Inquirer: *57 percent of IT professionals are sad individuals*. [online], London : Incisive Media Limited, 2007, dostupné na URL adrese: `http://www.theinquirer.net/inquirer/news/145/1041145/57-cent-it-professionals-sad-individuals`

[6] NOSKA, M.: *Windows 7: Na jeho vývoji pracuje 2500 expertov Microsoftu*. [online], Bratislava : Digital Visions, spol. s r.o, 2008, dostupné na URL adrese: `http://www.itnews.sk/buxus_dev/generate_page.php?page_id=55956`

[7] LAMMERS, S. – ZNAMENÁČEK, T.: *C. Wayne Ratliff – 2 (Programmers at Work)*. [online], Praha : Stickfish, s. r. o, 2008, (preklad z Redmod, WA : Microsoft Press, 1986), dostupné na URL adrese: `http://www.abclinuxu.cz/clanky/rozhovory/c.-wayne-ratliff-2-programmers-at-work`

Kontaktná adresa
Peter FODREK (Ing., PhD),
Ústav riadenia a priemyselnej informatiky FEI STU v Bratislave,
Ilkovičova 3, 041 20 Bratislava
peter.fodrek@stuba.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

VYUŽITIE / ZNEUŽITIE SUBVERSION V PEDAGOGIKE NA SLEDOVANIE PRIBEŽNEJ PRÍPRAVY POČAS SEMESTRA A PRI HODNOTENÍ

FODREK, Peter, (SK)

1 Úvod

Pri výučbe programovania sa z dôvodu nedostatku kvalifikovaných pedagógov a kapacít miest pri počítačoch resp. termináloch sa pracuje často v dvojiciach. V snahe naučiť absolventov tímovej práci bol vytvorený predmet *Tímový projekt*, ktorý však neuspel z dôvodu nepripravenosti učiteľov a infraštruktúry na administráciu tímov. V prípade tímového projektu ako aj výučby programovania v dvojiciach sa hodnotí každý študent zvlášť napriek tomu, že produkt je jediný za celú skupinu. To so sebou prináša konflikty, keď študent tvrdí, že na projekte pracoval rovnako intenzívne ako ostatní členovia napriek tomu, že výsledný produkt vôbec nepozná a pri otázke, kde použil danú technológiu hľadá v podrobných komentároch riadok, kde je to napísané. Tieto problémy sa dajú ľahko odstrániť použitím programov SCM (Source Code Management).

2 Source Code Management

SCM je softvér, ktorý vývojárom/programátorom pomáha s udržovaním prehľadu o projekte, na ktorom pracujú. SCM uchováva históriu jednotlivých zmien, a preto *nie je problém zistiť, kto, kde a kedy uskutočnil nejakú úpravu* (pozn. kde, čo a ako sa v kóde zmenilo/pridalo-/ubralo). Zároveň *zjednodušuje spoluprácu viac vývojárov*, ktorí môžu na *projekte pracovať súčasne* [1].

My sme sa rozhodli použiť jeden konkrétny SCM, a to Subversion (SVN). Jeho výhodou pre nasadenie v školstve je, že ide o centralizovaný SCM s tým, že na server sa posielajú len zmeny oproti predchádzajúcemu stavu a nie celé zmenené súbory, ako to robí napr. CVS (Concurent Versions System). Zo servera k vývojárovi idú len zmeny urobené inými vývojármi tak ako u CVS. Ďalším kladom SVN je to, že má podporu prostredia Eclipse, v ktorom robíme výskum, takže sa nemusíme preškoľovať, len doškoľovať, na prídavný modul pre CVS. To doškolenie by sme robili aj tak, lebo náš výskumný projekt narástol na cca. 500 tisíc riadkov, pri 5 vývojároch, a to už nejde riadiť bez SCM.

Pri testovaní SVN sme prišli na ďalšie vlastnosti SVN vhodné na výučbu: možnosť vytvárania patch-ov medzi repozitármi. Repozitár je miesto kam si tím odkladá svoje programy a ich časti. Patch je súbor opisujúci rozdiely medzi stromovými štruktúrami adresárov a súbormi v nich. Patch opisuje zmazania, pridania, premenovania/premiestnenia súborov a dát v súboroch. Z toho vyplýva, že malý patch znamená veľmi podobný výsledok práce tímov. Toto je, podľa skúseností, veľmi častý prípad. Na internátoch sa objavovali inzeráty ponúkajúce 2–5 tisíc slovenských korún, teda až do 170 € za vypracovanie zadania z predmetu, ktorý vyučujem. Programy sa teda logicky podobali, ale bolo však ťažké dokázať, kto je autorom, hoci to cvičiaci vie rozoznať. Pri použití SCM to nie je problém na základe časov vloženia zmien do SCM. Ak niekto vkladal zmeny postupne a niekto vložil výsledok naraz, deň pred odovzdávaním, je jasné že ten druhý odpisoval. Naviac, ak sa vytvoria repozitáre pre dvojice, a tieto po skončení semestra nezmažeme, máme zabezpečenú povinnú archiváciu zadaní pre možné reklamácie a súdne spory s ohľadom na pripravovaný štatút verejného činiteľa pre učiteľov. Tým sa odstráni argument: „ja som všetko vedel a dostal som menej bodov ako druhý z dvojice“, s ktorým sa stretávame. Obdobne sa dajú porovnávať semestrálne a záverečné práce v rámci univerzity, či centrálneho registra prác.

Účinnosť kontroly však znižuje použitie proprietárnych formátov požadované väčšinou učiteľov. Naopak, ak by sa celoplošne zaviedol formát ODF, kvalita porovnania by sa výrazne zlepšila. Podmienkou je mať repozitár pre každú dvojicu a každú záverečnú prácu. Ďalšou výhodou SVN je možnosť riadiť prístup k adresárom z repozitára pre jednotlivé oprávnené osoby t. j. členov dvojice a pedagógov. Dá sa teda zistiť, s ktorou časťou kódu pomohol učiteľ. Ba čo viac, učiteľ môže konkrétnemu študentovi dodať individuálne upravené študijné materiály s dôrazom na to, čomu nerozumie, prípadne odoprieť prístup k materiálom, ktoré obsahujú pomoc k témam, ktoré už boli ohodnotené a teda ich študent ovláda a nepotrebuje materiály.

3 Koncepcia nasadenia

Predstava o nasadení SVN je nasledovná

1. Vyberie sa server resp. použije sa už existujúci.
2. Nainštaluje sa SVN s overením voči PAM (cez SASLAUTH) alebo voči LDAP, kde

už majú študenti a učitelia prístupové používateľské účty.

3. Vyberú sa dvojice a každej dvojici sa vytvorí repozitár.
4. Nastavia sa práva prístupu len pre autorizovaných užívateľov, t. j. členov dvojice a pedagógov.
5. V rámci repozitárov sa vytvoria adresáre na študijné materiály a v nich podadresáre pre každého z dvojice. K týmto podadresárom pridáme práva na zápis pre pedagógov a na čítanie len pre príslušného študenta.
6. Aktivujú sa repozitáre a

   (a) do pravidiel práce v laboratóriu sa pridá povinnosť pridať zmeny do repozitára na konci cvičenia,

   (b) zabezpečí sa automatický commit (pridanie zmien do repozitára) pri odhlásení,

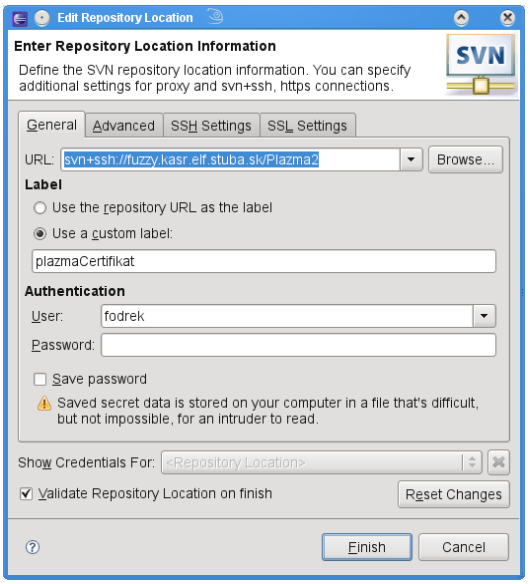
   (c) zabezpečí sa automatický commit v 10 minútových intervaloch počas cvičení.

7. Aktivujú sa prístupy k repozitárom na počítačoch učiteľov.
8. Otestuje sa účinnosť pri prvom čiastkovom zadaní z troch.
9. Pri príprave na odovzdanie druhého čiastkového zadania sa učiteľ pripraví na otázky dvojici na základe stavu repozitára a to tak, že vyberie otázky pre študenta z tej časti, ktorú nerobil.
10. Zhodnotí sa prínos.

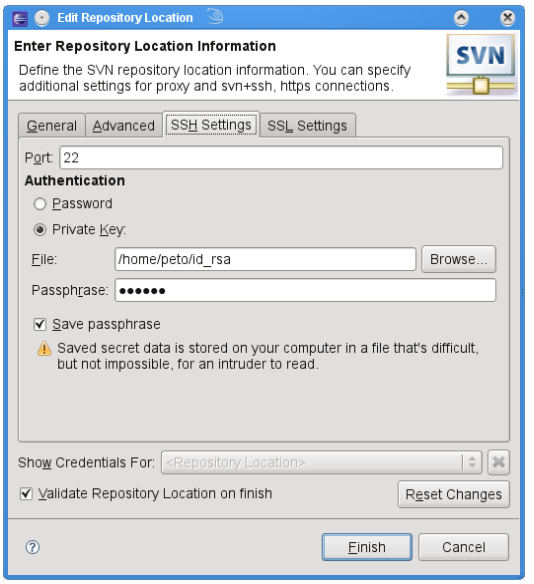
4 Realizácia

S realizáciou koncepcie sa začalo, ale z dôvodu neochoty administrátora, sa začiatok posunul a zmenil sa aj pôvodne zamýšľaný server, ako aj administrátor. Nakoniec sa realizácia ukončí až po vytvorení laboratória Unix a open source na našej fakulte. To by malo vzniknúť v roku 2010, v rámci centra excelencie s podporou KEGA. V tomto semestri majú študenti len možnosť systém využiť, nie je to však povinnosť. V rámci výskumu však máme vytvorené repozitáre napr. pomocný na URL/URI podľa Obr. 1 a Obr. 2, ktoré sú ukážkou nastavenia repozitára vo vývojárskom systéme Eclipse pri použití SVN doplnku Subversive. Tento doplnok bol použitý pre nutnosť použiť SSH tunel na server s doménovým menom `fuzzy.kasr.elf.stuba.sk`. Tento server podporuje len SSH, HTTPS a HTTP spojenia. Iný SVN doplnok pre Eclipse, s názvom Subclipse, ktorý dodáva tvorca SVN, nepodporuje protokol SVN cez SSH. Subversive potrebuje na svoju činnosť externé tzv. SVN konektory, ktoré podporujú SVN + SSH. Tým bol výber doplnku daný. Eclipse sme vybrali aj z dôvodu jeho multiplatformovej orientácie.

Na server FUZZY, bolo potrebné inštalovať SVN, a to vo verzii 1. 5. 6 zo zdrojových kódov, lebo Fuzzy beží na FreeBSD, a na ňom predinštalovaná verzia nevyužívala Berkeley



Obr. 1: Nastavenie repozitára



Obr. 2: Nastavenie SSH autentifikácie

database, tak sme urobili lokálnu inštaláciu s ňou. Následne sme museli nastaviť tunel na strane servera. Hlavne bolo potrebné nastaviť, ktorý program sa spustí na serverovej strane tunela a vložiť verejnú časť RSA kľúča, ku ktorému súkromnú časť sme zadali Eclipse. Realizácia podľa listingu 1, nám umožnila využiť jediný používateľský účet na pripojenie viacerých používateľov s tým, že ich meno, podľa ktorého identifikujeme ich príspevky, je závislé len na ich kľúči a je zadané ako *parameter tunnel-user* pri spúšťaní servera na serverovej strane tunela. Navyše študenti nemusia poznať heslo k účtu.

Listing 1

```
-bash-2.05b$ pwd
/home/fodrek/.ssh
-bash-2.05b$ more authorized_keys

command="/home/fodrek/my_progs/bin/svnserve -t --tunnel-user=petoF -r
/home/fodrek/repos/" ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA3jGrhqp+9MZPpcGFbEbpkj0gXtuVaBXOOPqlfkKUk5c
qAh44z2aTSgKEUxmkSua248AuYzjMUdSiQYo/prOGWLnSxt86CkrXSp7f6narOaS6fildja
u7Bk7kvvgydjocealY+QrO83DOcAviZyYSnzP5ouIDQ5acClj33tg4Ak5wrKIJQwrxmBsXq
MNBAYwQyDL4urXk6xLVX6k0SO5v9VxdMoGmxSyS8DU6WhFsc3YtdoxHn2PM4IwQpdHbZknt
cMEVifsG4tGgbkzODRgJyylWjeXnMvAf+pR9wfRK63v6zjr2VBCOyX1PXPkKyAW2BkRL88Y
git/k+DjYUdLO9Q==peto@fodrek

command="/home/fodrek/my_progs/bin/svnserve -t --tunnel-user=student -r
/home/fodrek/repos/" ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAABIwAAAQEAo9qpiGQoFdj1t+OJXxoUM0YoS/C/stwyt0BDxgun2U2
ELu9P0+DBFfbEAs60GU1CJ+2NAqNwGXOLUrMw0+fBkrZK28di6Di3qxbtq2Wp+zBQZOJ//k
N+/5XD0eXY6wHokNfkep7M4JdewK89a1Hf3qp84QaZsjtLjz0xAC7cOk2Sw+rUoIS4n7MI4
uCHHBtt8G2V8+/plNeqSp4dGvg71ZmR+/QBvZn6G1gSuyKrylPvCCxab7gggyxeGg9X38gh
LsoIfPzgVhseXs0tKzDGN7P59chT+E9Pc27S5aJGF+UO9Eb74PBBemoZtvsdw/bcWdjWhsc
LcGONPGcN9wIsQ==student@fodrek

command="/home/fodrek/my_progs/bin/svnserve -t --tunnel-user=vlado -r
/home/fodrek/repos/" ssh-rsa
AAAAB3NzaC1.....................................................
.....................................== vladu@zrj

command="/home/fodrek/my_progs/bin/svnserve -t --tunnel-user=plazmak
-r /home/fodrek/repos/" ssh-rsa
AAAAB3N.........................................................
.................................................==plazmak@fodrek
```

Tým je realizácia SVN pripravená na monitorovanie študentov.

5 Výsledky

Podrobnejšie výsledky budú dostupné v priebehu zimného semestra šk. roku 2009/2010, počas predmetu, na ktorom sa učí Java pod OS Windows. Je to pre to, že počas predmetu o systémovom paralelnom programopvaní v UNIXe, nakoniec pre posun termínu nezvýšil čas na nasadenie SVN a záujem o predvedenie prejavili len dve dvojice. Toto je veľká škoda. Na koniec uvedieme výstup z Eclipse, kde je ukázaný potenciál monitorovania práce študentov podľa výpisu histórie projektu na Obr. 3.

6 Záver

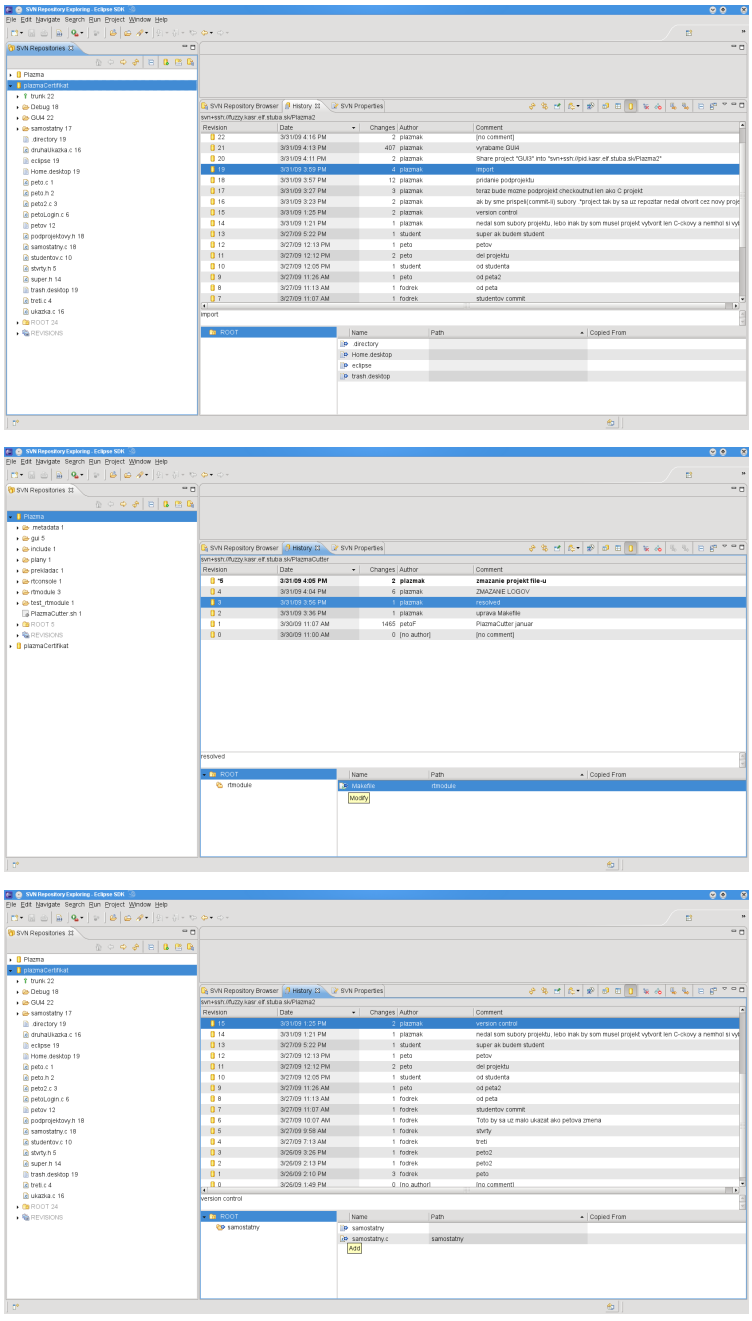
SCM Subversion je vhodný a potrebný prostriedok na nasadenie a riadenie práce v dvojiciach a tímoch ako aj porovnávanie originality prác. Túto skúsenosť však treba overiť aj počas nasledujúceho semestra.

Literatúra

[1] KŘENEK, M. – KRÁTKÝ, R. – WATZKE, D.: *SCM*. Praha : Stickfish, s. r. o., 2006–2008, dostupné na URL `http://www.abclinuxu.cz/slovnik/scm`

Kontaktná adresa

Peter FODREK (Ing., PhD),
Ústav riadenia a priemyselnej informatiky FEI STU v Bratislave,
Ilkovičova 3, 041 20 Bratislava
peter.fodrek@stuba.sk



Obr. 3: Zmeny, ich autori, čas a typ zmeny (konkrétne zmeny sa dajú zistiť tiež)

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

UML/DSM NÁSTROJ UML .FRI

JANECH, Ján, (SK)

1 Úvod

Pri analýze softvérových produktov sa v stále väčšej miere využívajú rôzne CASE nástroje. Tie pokrývajú rôzne fázy vývoja softvéru. Niektoré sú vhodné na biznis analýzu, niektoré na návrh objektovej štruktúry, či návrh databázy. Niektoré, ako UML sa snažia pokryť celý vývojový proces. Mnohé z nich sú dokonca zamerané na konkrétne typy projektov, ako simulácie alebo multimediálne informačné systémy.

V priebehu štúdia informatiky na vysokej škole príde študent do styku s mnohými z týchto typov nástrojov. Vizualizácia štruktúr a algoritmov, ktorú prinášajú CASE nástroje je dobrou pomôckou pri vyučovaní a umožňuje jednoduchšie pochopenie preberanej látky. Okrem toho sa študent stretne s podobnými nástrojmi, aké sa v danej oblasti používajú aj v praxi. Analýza problémov v CASE dokonca podporuje abstraktné myslenie a učí študentov nepozerať hneď na implementáciu v konkrétnom programovacom jazyku. Znamená to však používať niekoľko rôznych nástrojov na rôznych predmetoch, čo pre školu automaticky znamená veľké výdavky na ich zakúpenie. Okrem toho sa treba starať o pravidelnú aktualizáciu týchto nástrojov na nové verzie, čo tiež stojí peniaze.

Napriek širokému rozsahu zameraní, majú tieto nástroje veľa spoločného. Väčšinou sa jedná o zápis a prezentáciu informácií pomocou diagramov rôznych druhov a následné automatické spracovanie týchto diagramov. Líšia sa len typom podporovaných diagramov a spôsobom ich automatického spracovania. Špeciálnou výnimkou sú nástroje DSM[1]. Tie nemajú špeciálne zameranie (ako napr. E-R, BPMN, . . .) a ani sa nesnažia pokryť všetky potreby jedným univerzálnym modelovacím jazykom (ako napr. UML). Namiesto toho definujú

[1] Domain Specific Modeling

jazyk na definíciu modelov (metamodel), a výber alebo návrh konkrétneho modelovacieho jazyka zostáva na užívateľovi. Nechávajú mu teda slobodu vybrať si modelovací jazyk, ktorý sa mu najviac hodí na použitie v konkrétnej situácii.

Tento princíp sa dá s výhodou využiť pri vyučovaní. Stačí mať jeden nástroj a len zadefinovať metamodely podľa požiadaviek konkrétnych predmetov. Okrem toho tým vzniká možnosť využiť nástroj aj na predmetoch, kde žiadny nemali. Na Fakulte riadenia a informatiky Žilinskej univerzity v Žiline sa teda v roku 2005 začal vývoj vlastného DSM nástroja, ktorý by plne zodpovedal požiadavkám na nástroj používaný pri výučbe.

Nástroj je označovaný ako UML/DSM, čím je označené jeho hlavné zameranie na jazyk UML a technológie s ním spojené. Jeho názov je UML .FRI a stiahnuť sa dá pod licenciou GPL zo stránky `umlfri.org`.

2 Nástroj vyvíjaný študentmi

Nástroj UML .FRI je vyvíjaný študentmi Fakulty riadenia a informatiky v rámci projektovej výučby počas inžinierskeho štúdia. Študenti sa teda na projekte striedajú a každý rok nastúpi nová skupina. Na výmenu skúseností majú vždy dve skupiny vyhradený jeden semester. Tento spôsob vývoja má ako výhody, tak aj nevýhody.

Medzi nevýhody určite patrí pomalý vývoj. Vzhľadom na to, že sú študenti vyťažení štúdiom, práca na projekte prebieha pomaly. Okrem toho, ako už bolo spomenuté, jeden tím pracuje na projekte dva semestre, z ktorých jeden je vyhradený na výmenu skúseností a začlenenie tímu do projektu. Aby dohnali stratený čas, sú často nútení na projekte pracovať počas víkendov, sviatkov, alebo prázdnin. Napriek tomu, keby existoval vývojový tím, ktorý by sa plne venoval vývoju nástroja, určite by práce napredovali omnoho rýchlejšie.

Ďalšou nevýhodou je tiež určite nižšia kvalita kódu. Na projekt prichádza celá škála študentov, od tých najlepších až po tých najslabších. Napriek snahe vedúcich projektu, nie je možné skontrolovať zdrojové kódy vytvorené všetkými študentmi. Tým sa do projektu dostáva isté množstvo zle navrhnutého, resp. zle implementovaného kódu.

Medzi nesporné výhody určite patrí cena riešenia. Vývoj študentmi ako F/OSS riešenie minimalizuje náklady na vývoj. Zaplatiť stále treba potrebnú techniku, učebnice a softvér. Stále je to však lacnejšie, ako nechať softvér vyhotoviť externou firmou, alebo zamestnať na vývoj zamestnancov na plný úväzok.

Okrem toho, študenti sú schopní, v spolupráci s vedúcimi spomedzi učiteľov, lepšie špecifikovať požiadavky na daný softvér. Práve študenti budú tí, čo budú výsledný nástroj používať. Majú skúsenosti z rôznych nástrojov používaných na škole a dokážu povedať, aké chyby mali, ktorým sa treba vyhnúť.

Za výhodu je možné považovať aj to, že sú študenti ešte počas štúdia zaradení do reálneho projektu a sú nútení pracovať na ňom v tíme. Majú vďaka tomu možnosť zistiť, aké výhody plynú z dobrého návrhu. Vyskúšajú si rôzne nástroje na podporu práce v tíme, ako verzionovací systém, alebo nástroj na správu chybových hlásení. Tak isto sa naučia

technológie, s ktorými sa počas štúdia vôbec nestretli, alebo im bolo venované málo priestoru (Python, GTK+, XML, XML Schema, metaprogramovanie. . .), čím si zlepšujú rozhľad.

2.1 Projektová dokumentácia

Ako bolo spomenuté vyššie, projekt je vyvíjaný generačne, pričom dve po sebe idúce generácie majú iba jeden semester na výmenu skúseností. Projekt však stále naberá na zložitosti. V dnešnej dobe už neexistuje človek, ktorý by mal prehľad v celom projekte do podrobností, čo komplikuje plánovanie prác a komplikuje vývoj nových vlastností. To prispelo k tomu, že vznikla nutnosť vypracovať kvalitnú programátorskú dokumentáciu.

Tá sa začala vytvárať na dvoch úrovniach:

- *API dokumentácia* – je realizovaná ako generovaná dokumentácia vnútorného API aplikácie. Ako formát dokumentačných komentárov bol použitý pravidlami obmedzený Epytext[2]. Vďaka tomu sa dá jednoducho pomocou nástroja Epydoc vygenerovať prehľadná dokumentácia vo formáte HTML, alebo PDF. Táto dokumentácia je pravidelne generovaná skriptom každý deň o polnoci a vystavená na projektovej stránke. Vďaka tomu je dostupná aj tým vývojárom, ktorí nemajú Epydoc nainštalovaný.

- *Projektová dokumentácia* – opisuje princípy fungovania aplikácie, jej architektúru a technológie, na ktorých je založená. Dokumentácia je určená hlavne na zmiernenie problémov so zaúčaním novej generácie študentov. Písaná je preto ako ucelený dokument. Aby bola zabezpečená jednoduchosť editácie viacerými študentmi súčasne, bol za formát zvolený LaTeX, zdieľaný pomocou systému Subversion. Dokumentácia je síce verejná, ale jej editácia je momentálne prístupná len interným členom tímu UML .FRI. Dokumentácia je momentálne v štádiu riešenia.

2.2 Diplomové práce

Po skončení prác v rámci projektovej výučby majú študenti možnosť pokračovať na projekte vo forme diplomovej práce. V takom prípade je vybraná jedna konkrétna problematika, ktorú potom po teoretickej aj praktickej stránke spracuje študent ako svoju diplomovú prácu.

Ako prvé sa v rámci diplomových prác riešili transformácie M2T[3] a T2M[4]. Študenti navrhli kompletný transformačný jazyk, vďaka ktorému je možné jedným zápisom definovať obe transformácie. Okrem toho byli vytvorené transformácie na generovanie zdrojových kódov (C++, Delphi Pascal a Python) a reverzné inžinierstvo pre diagram tried a generovanie textovej dokumentácie z modelu UML vo formáte HTML. Tieto transformácie môžu uľahčiť prácu s nástrojom na predmetoch, na ktorých sa vyučuje algoritmizácia, objektové

[2] Formátované dokumentačné reťazce podobné formátu javadoc

[3] Model-to-Text – transformácia modelu do textovej podoby

[4] Text-to-Model – reverzné inžinierstvo zdrojových kódov do modelu

programovanie, údajové štruktúry a podobne, lebo z modelu umožňujú vygenerovať (po zadefinovaní transformácie) kompletný zdrojový kód.

Ďalšou diplomovou prácou je systém zásuvných modulov[5]. Ten umožňuje rozšírenie nástroja o automatické spracovávanie diagramov. Napríklad môžu vykonávať rôzne algoritmy z teórie grafov, alebo asistovať študentovi pri tvorbe modelu. Zásuvné moduly môžu byť vytvorené v takmer ľubovoľnom jazyku, vďaka čomu ich môže vytvárať každý so základnými znalosťami objektového programovania.

Zatiaľ posledná diplomová práca rieši implementáciu systému UNDO/REDO[6]. Na jeho implementáciu existuje niekoľko algoritmov, cieľom študenta bolo nájsť najvhodnejší a upraviť ho pre potreby nástroja UML .FRI. Okrem toho musel identifikovať miesta v nástroji, ktoré bránia implementovať systém UNDO/REDO a nájsť riešenie vzniknutých problémov.

V budúcnosti sa plánuje niekoľko ďalších diplomových prác (transformácie M2M[7], podpora pre elektronické tabule. . .), ale aj bakalárske práce na jednoduchšie časti nástroja (napr. import projektov uložených v iných nástrojoch). Bakalárske práce môžu slúžiť ako úvod do projektu, čím by sa mohol zjednodušiť nástup novej generácie na projekt v inžinierskom štúdiu. Všetky diplomové a bakalárske práce by mali zjednodušiť nasadenie nástroja vo vyučovacom procese, alebo priniesť do neho nové možnosti.

3 Použité technológie

Celý systém je naprogramovaný v jazyku Python. Implementácia mohla byť vďaka tomu na niektorých miestach zjednodušená dynamickým typovaním a využitím metaprogramovania. Zdrojové kódy sú vďaka tomu kratšie a prehľadnejšie, ako v iných programovacích jazykoch. Jazyk Python však umožňuje niektoré konštrukcie, ktoré odporujú praktikám zaužívaným v objektovom programovaní (hlavne neobsahuje nástroje na vynútenie zapuzdrenia), čo si vynútilo vydanie príručky „Coding style[8]“, ktorá popisuje, čo môžu študenti využívať a čo nie. Tá je súčasťou projektovej dokumentácie. Z istého pohľadu je to aj výhoda, lebo sa tým učia určitej programátorskej disciplíne.

Vzhľadom na to, že programovací jazyk Python neobsahuje žiadne východzie knižnice pre tvorbu GUI rozhrania, bolo treba vybrať externú knižnicu. Zvolená bola GTK+. Je to multiplatformová knižnica na tvorbu GUI. Vďaka kombinácii Python a GTK+ je možné portovať aplikáciu na takmer ľubovoľnú platformu, od OS GNU/Linux, MS Windows, až po mobilné telefóny, alebo rôzne zabudované systémy. Knižnica je napísaná v jazyku C, čo čiastočne obmedzuje možnosti objektového programovania. Preto bola do aplikácie pridaná podpora pre objektové zapuzdrenie prezentačnej vrstvy.

Na vykresľovanie diagramov je použitá knižnica Cairo. Tá umožňuje jednoduchý spôsob práce s vektorovou aj bitmapovou grafikou. Výsledné obrázky dokáže prezentovať priamo

[5] anglicky plugin

[6] vrátiť poslednú činnosť/zopakovať činnosť

[7] Model-to-Model – transformácia modelu na iný model

[8] Štýl programovania

užívateľovi, ale aj exportovať do rôznych vektorových a bitmapových formátov. Okrem toho prináša výhody hardvérovej akcelerácie, podporuje anti-aliasing, transformačné matice a podobne.

Na ukladanie štruktúrovaných dát bol zvolený formát XML. Jeho hlavnými výhodami sú jednoduché spracovanie pomocou technológií DOM alebo Element Tree a možnosť jednoduchej validácie voči definícii vo formáte XML Schema. Uložené modely sú navyše komprimované vo formáte ZIP. V súčastnosti je to pomerne často používaná kombinácia. Študenti sa teda zoznámia s možnosťami spracovania tohoto formátu, jeho výhodami i nevýhodami.

Z dôvodu rozsahu projektu sa študenti nezoznámia do podrobností s každou uvedenou technológiou. Na tom sa prejaví aj ich schopnosť tímovej práce. Každý študent musí pracovať na inej časti a skrývať detaily použitej technológie za vopred dohodnuté rozhrania vnútornej objektovej štruktúry.

4 Nástroj používaný študentmi

Ako bolo uvedené vyššie, nástroj UML .FRI je určený hlavne pre podporu vyučovania. Cieľovou skupinou sú teda študenti. Tomu je prispôsobený aj systém vývoja. Študenti na niektorých predmetoch sú motivovaní, aby nástroj testovali a hlásili chyby.

Plánu rozšíriť nástroj na začiatku na Fakulte riadenia a informatiky boli podriadené aj používané nástroje. Najdôležitejší spôsob spätnej väzby – hlásenie chýb je implementovaný dvoma rôznymi spôsobmi.

- *Hlásenie chýb priamo z aplikácie* – V prípade, že aplikácia spôsobí automaticky detekovateľnú chybu, užívateľovi sa zobrazí okno s chybovou správou. To priamo obsahuje možnosť automatického odoslania informácie o chybe na server. Hlásenie okrem samotnej chybovej správy voliteľne obsahuje užívateľov projekt, aby mali vývojári na čom testovať, a krátky popis chyby od užívateľa.

- *Hlásenie chýb pomocou systému na správu chybových hlásení* – Na internetovej stránke projektu sa nachádza odkaz na systém na správu chybových hlásení[9]. Užívateľ je síce nútený sa zaregistrovať, ale registrácia je voľná a nie je ani viazaná na štúdium na Fakulte riadenia a informatiky. Vzhľadom na cieľovú skupinu je celé rozhranie v slovenčine a takisto chybové hlásenia sa píšu po slovensky.

Nasadenie aplikácie na vyučovaní prebieha postupne podľa toho, ako sa do aplikácie implementujú nové vlastnosti. Už pred zaradením nástroja do výučby niektorého predmetu ho používalo niekoľko študentov na tvorbu modelov do semestrálnych prác. Bolo ich však málo a tak spätná väzba od študentov bola malá.

[9] Bug tracking system

Ako prvé prebehlo nasadenie na predmet Objektové programovanie v druhom ročníku, kde boli študenti bodmi motivovaní hlásiť chyby v aplikácii. Študenti začali nástroj používať aj na tvorbu semestrálnych prác. Spätná väzba bola už väčšia, študenti boli s nástrojom spokojní, dokázal čo potrebovali (jednoduché UML diagramy tried). Napriek tomu sa našlo dosť veľké množstvo chýb, ktoré boli členmi tímu odstraňované priebežne.

Ďalší predmet, na ktorom sa nástroj používal, bol predmet Základy informatiky 2 v prvom ročníku. Na tomto predmete sa využíval na modelovanie vývojových diagramov. Bolo to však tento semester a na spätnú väzbu od študentov v dobe písania tohoto článku sa ešte len čaká.

Budúci rok sa plánuje využitie nástroja na predmete Informatika 1, 2 v prvom ročníku. Je to nový predmet, ktorý nahrádza Základy informatiky a bude sa na ňom vyučovať metódou „Objects first“[10]. Preto bude treba nástroj na modelovanie UML diagramov tried.

Napriek tomu, v súčastnosti sa využívajú iba dva typy modelov, v pláne je implementovať v budúcnosti aj iné (grafy, DFD, BPMN), aby bolo možné nástrojom UML .FRI nahradiť iné, doteraz využívané nástroje. Študenti tým získajú jednotné prostredie, ktoré bude plne vyhovovať požiadavkám daných predmetov.

5 Metamodel

Ako už bolo uvedené, medzi najväčšie výhody nástroja patrí oddelenie metamodelu od zdrojových kódov aplikácie. Tým sa získava veľká flexibilita nástroja a možnosť jeho využitia v rôznych oblastiach. Aplikácia je schopná automaticky rozpoznať typ modelu a na základe toho zvoliť príslušný metamodel. V počítačových laboratóriách na škole teda stačí mať nainštalovaný nástroj UML .FRI jeden krát, ale s niekoľkými metamodelmi. Na predmete objektové programovanie teda študent vyberie šablónu modelu pre objektové programovanie a automaticky sa zvolí metamodel pre UML. Ak si na predmete Základy informatiky vyberie šablónu pre tento predmet, zase sa zvolí príslušný metamodel.

Väčšinu metamodelov budú definovať študenti v rámci prác na projektovej výučbe. Vzhľadom na otvorený formát metamodelu definovaný pomocou štruktúr XML si dokáže vytvoriť metamodel každý s príslušnými znalosťami daného typu modelov.

Metamodel v dobe písania tohoto článku obsahuje 4 typy objektov.

- *Objekt typu doména* – obsahuje definíciu metadát pre ostatné typy objektov (element, spojenie). Popisuje, aké parametre (atribúty) môže užívateľ pre daný objekt zadať. Ku každému atribútu definuje typ a implicitnú hodnotu, ktorá bude nastavená pri vytvorení objektu. Domény, ako jediný typ objektov môžu byť vnorované, teda každý atribút môže mať ako typ uvedenú doménu, alebo zoznam položiek s definovanou doménou.

- *Objekt typu element* – reprezentuje element diagramu. Elementy môžu byť napríklad

[10] Začína sa s objektovo orientovaným programovaním

Obr. 1: Príklad vzhľadu elementu

triedy, tabuľky v databáze, vrcholy grafu a podobne. Definícia elementu obsahuje rôzne informácie, od názvu elementu až po jeho vzhľad.

- *Objekt typu spojenie* – reprezentuje spojenie medzi elementmi. Spojenie je tvorené lomenou čiarou a voliteľne aj množinou popisných políčok. Tie sa zobrazia na určených miestach pri čiare spojenia a zobrazujú užívateľom definované atribúty spojenia.

- *Objekt typu diagram* – reprezentuje diagram v modeli. Definícia diagramu obsahuje zoznam povolených elementov a spojení pre daný typ diagramu. Plánovaná je aj podpora domén pre diagramy, aby mohol užívateľ definovať aj hodnoty atribútov diagramu.

Špeciálna pozornosť pri návrhu metamodelu bola venovaná definícii vzhľadu elementov a spojení. Toto je oblasť, ktorej mnohé konkurenčné DSM nástroje nevenujú veľkú pozornosť. Ak sa však má nástroj používať na výučbu existujúcich modelovacích nástrojov, ktoré sú často štandardizované aj čo sa týka vzhľadu, je nutné aby bol grafický výstup správne sformátovaný, aby neboli študenti zmätení, keď začnú pracovať s konkurenčným nástrojom.

Na zápis definície vzhľadu bol zvolený systém kontajnerov. V zápise existujú tri typy objektov.

- *Komplexný kontajner* v sebe môže obsahovať ľubovoľný počet iných objektov. Komplexné kontajnery existujú v systéme dva: s horizontálnym usporiadaním a s vertikálnym usporiadaním.

- *Jednoduchý kontajner* môže obsahovať len jeden objekt. Ako jednoduchý kontajner je implementovaná väčšina grafických objektov. Napríklad obdĺžnik, elipsa, tieň. . .

- *Elementárny objekt* neobsahuje ďalšie objekty. Je to text a oddeľovacia čiara.

Vždy platí, že najskôr sa vykreslí samotný kontajner a až následne na to sa vykreslia objekty vnútri v ňom. Druhý a posledný princíp je, že veľkosť v akej sa vykreslí kontajner je určená veľkosťou objektov v ňom. Celý princíp sa dá ukázať na nasledujúcom príklade. Na obrázku 1 je uvedený vzhľad elementu definovaného touto štruktúrou:

```
<Shadow padding="3" color="gray">
    <Rectangle fill="lightyellow" border="black"
                right="10 rounded" left="10 rounded">
        <Padding padding="5">
            <Align align="center center">
```

```
                <TextBox text="END" font="Arial 10" color="black" />
            </Align>
        </Padding>
    </Rectangle>
</Shadow>
```

Na najvyššej úrovni je tieň. Ten sa vykresľuje ako prvý. Tieň obsahuje obdĺžnik so zaoblenými stranami. Na úplne najnižšej úrovni je text a teda jeho veľkosti sa prispôsobuje celý vzhľad elementu.

Tento zápis umožňuje jednoducho definovať takmer ľubovoľný vzhľad elementu. Vďaka tomu je možné pokryť aj takú zložitú normu ako je UML. Okrem toho kontajnerový spôsob definície sa postará automaticky o určovanie veľkosti podľa užívateľom zadaných atribútov elementu a teda sa nestane, že by text pretekal mimo vyhradenej plochy.

6 Záver

Pri výučbe rôznych predmetov na vysokej škole je výhodné používať jednotné prostredie CASE. Keď sa k tomu priráta cena, je nástroj UML .FRI schopný konkurovať rôznym komerčným nástrojom. Jeho výhodou je veľká flexibilita, ktorá mu umožňuje pokryť kompletne niekoľko oblastí v rámci jednej inštalácie.

Flexibilita však nie je jediné čo musí nástroj podporovať, aby bol použiteľný vo výučbe. Do budúcnosti zostáva na nástroji ešte veľa práce, aby bol schopný pokryť čo najväčšiu časť vyučovacieho procesu. Je potrebné pridávať nové metamodely a rozširovať existujúce, podľa požiadaviek jednotlivých predmetov a pridávať nové možnosti samotného softvéru, podľa požiadaviek užívateľov.

Poďakovanie

Táto práca vznikla s podporou grantovej agentúry KEGA v rámci riešenia projektu 3/2158/04 „Využitie open source softvéru vo výučbe na vysokých školách“.

Literatúra

[1] BAČA, T. – JURÍČEK, M. – ODLEVÁK, P.: *UML .FRI – Case tool developement*. In: *Journal of Information, Control and Management Systems*. 2008 vol. 6, no. 1, ISSN 1336-1716

[2] *DSM Forum*. [online].
Dostupné online: <http://www.dsmforum.org>.

[3] JANECH, J. – SADLOŇ, Ľ.: CASE UML .FRI. In: *OBJEKTY 2007*. Ostrava : Fakulta elektroniky a informatiky, VŠB – Technická univerzita Ostrava, november 2007, ISBN 978-80-248-1635-7

[4] Wikipedia: *Domain-specific modeling – Wikipedia, The Free Encyclopedia*. [online]. Dostupné online: <`http://en.wikipedia.org/wiki/Domain-specific_modeling`>.

Kontaktná adresa

Ján JANECH,
Katedra softvérových technológií FRI ŽU v Žiline,
Univerzitná 1, 010 26 Žilina,
jan.janech@kst.uniza.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

WEB2PY FRAMEWORK NA TVORBU WEBOVSKÝCH APLIKÁCIÍ

KARABÁŠ, Ján (SK), SILÁČI, Jozef (SK), ŠUCH, Ondrej, (SK)

1 Úvod

Programovanie webovských stránok je najrozšírenejším druhom programovania. Zamestnávatelia očakávajú od absolventov skúsenosti v najnovších, najproduktívnejších webovských technológiach. V súčasnosti výučba týchto technológií prebieha zhruba v štyroch predmetoch/tematických okruhoch:

- *Internetové technológie*, kde sa študent oboznámi s technológiami HTML, XML, JavaScript.
- *Databázové systémy*, ktorý uvedie študenta do sveta relačných databáz a modelovania dát pomocou relácií.
- *Programovanie v OO jazyku*, výučbou moderného objektovo orientovaného programovacieho jazyka ako je C# alebo Java.
- *Programovanie webovských aplikácií*, kde sa predchádzajúce tri predmety prakticky využijú na výučbu tvorby JSP alebo ASP aplikácií.

Ako alternatívu k posledným krokom sa niekedy využíva tzv. LAMP stack, kde sa tieto predmety nahradia praktickou výučbou jazyka PHP.

Hlavným problémom súčasných prístupov je komplexita týchto technológií. Štandardným prístupom k odstráneniu komplexity vo vývoji softvéru je premiestniť bežné, no časovo náročné postupy do *softvérových knižníc* alebo komplexnejších vývojových balíkov tzv. *frameworkov*. Frameworkov na tvorbu webovských aplikácií existujú stovky, ako potvrdí krátky pohľad do Wikipedie [3]. Vybratie ideálneho frameworku pre začínajúcich študentov nie je

ľahké. Na jednej strane je trh dosť fragmentovaný, a nie je možné naučiť študentov všetky najpoužívanejšie frameworky. Na druhej strane, nie je možné vyučovať technológiu, čo ako krásnu, ktorú by študenti nemohli využiť ďalej v praxi.

My sme pri výbere frameworku zvažovali nasledujúce kritériá:

- pozvoľne stúpajúca krivka náročnosti,
- nabádanie k všeobecne odporúčaným vývojarským postupom,
- výber programovacieho jazyka,
- podpora viacjazyčnosti,
- integrácia,
- podpora viacerých platforiem,
- interoperabilita,
- expresívnosť a potenciál na rozširovanie,
- cena,
- použitie v praxi,
- materiály na výučbu.

Nie je možné v rozsahu tohto článku, a ani v schopnostiach autorov, zvážiť všetky možné frameworky a technológie použiteľné na webovské programovanie. Našim cieľom bude poukázať, že open-source framework web2py (`http://www.web2py.com`) je podľa týchto kritérií veľmi vhodným kandidátom na výučbu. Pri hodnotení sme vychádzali z našich doterajších skúseností s výučbou tradičných technológií (PHP, SQL, Java, .NET).

2 Framework web2py

Tento framework bol vyvinutý špeciálne na výučbu tvorby webovských aplikácií. Jeho hlavným autorom je Massimo Di Pierro, profesor informatiky na DePaul University v meste Chicago. Zdrojový kód frameworku je licencovaný pod GPL licenciou verzie 2, a k jeho vývoju prispel celosvetový tím vývojárov vrátane jedného z autorov [10]. Je vybudovaný na jazyku Python, a od väčšiny frameworkov sa líši poskytnutím DAL (database abstraction layer) komponentu, ktorý je používaný na udržiavanie modelu dát. Podobne ako LINQ technológia od Microsoftu, tento komponent umožňuje programátorovi uvoľniť sa z diktátu relačných databáz a pracovať s dátami bez toho, aby priamo používal SQL.

2.1 Výber jazyka

Existujú webovské frameworky pre jazyky od Erlang až po Perl. *Ak si môžete vybrať hociktorý jazyk, ktorý by ste si vybrali?* Táto rečnícka otázka má veľa odpovedí. Tu citujeme odpoveď jedného z prvých internetových milionárov. Paul Graham takto [9] vysvetľuje, prečo si vybrali jazyk Lisp pre ich firmu:

Vybrali sme Lisp. *Jedným z dôvodov bol, že rýchly vývoj bol zjavne veľmi dôležitý na tomto trhu. Všetci sme začínali od nuly, a tak firma, ktorá vyvinie funkcionalitu skôr než konkurenčné, bude mať veľkú výhodu. Vedeli sme, že* Lisp *je veľmi dobrý jazyk na rýchly vývoj softvéru, a serverovské aplikácie ešte umocňujú tento efekt, pretože môžete publikovať softvér v tú istú minútu ako ho dokončíte.*

Hoci výhody jazyka Lisp sú neodškriepiteľné, treba poznamenať, že je nerealistické očakávať od študentov bakalárskeho stupňa, aby sa naučili efektívne používať a oceniť také zložité koncepty jazyka Lisp ako je *continuation passing style programming*.

Alternatívnym pohľadom na výber jazyka je .NET stratégia firmy Microsoft. Táto nepredurčuje jeden jazyk (ako napríklad Java), ale umožňuje písať zdrojový kód s použitím množstva programovacích jazykov, ktoré sa všetky kompilujú do veľmi rýchleho .NET runtime. Samozrejme nevýhodou je, že aplikácie vyvinuté na .NET platforme bývajú s ťažkosťami prenositeľné z platformy Windows, a to napriek úsiliu firmy Novell vytvárať implementáciu .NET pre GNU/Linux a MacOS X.

Web2py je založený na jazyku Python. Hoci nie je najpoužívanejším jazykom, Python sa využíva v širokej škále aplikácií. Podľa Tiobe indexu [11] je to šiesty najpopulárnejší jazyk v júni 2009, predčiaci dokonca jazyk C# od spoločnosti Microsoft. Je to vhodný jazyk pre výučbu programovania, pretože podporuje rôzne programátorské paradigmy – imperatívnu, objektovo orientovanú ako aj funkcionálnu. Bez väčších problémov sa ho naučia aj začínajúci študenti. Je ľahko portovateľný a vďaka veľkému množstvu štandardných knižníc umožňuje tvorbu komplexných, interoperujúcich aplikácií na všetkých platformách. Používa slabšie typovanie než Java alebo C++ (táto téma a porovnanie s Javou sú bližšie rozobraté v [12]).

2.2 Internacionalizácia

V európskych podmienkach je podpora mnohojazyčnosti nevyhnutnou funkcionalitou vo webovskom frameworku. Napriek tomu, podľa našich skúseností, správne fungovanie jazykov iných ako anglických je pre začiatočníkov tým najfrustrujúcejším aspektom vývoja aplikácií. Stratégia je zdanlivo veľmi jednoduchá – vstupné dáta, väčšinou v znakovej sade UTF-8, by sa mali bez straty mäkčeňov a dĺžnov dostať do a späť z databázy. Napriek tomu, snáď kvôli veľkému množstvu kombinácií softwarových komponentov, cez ktoré sa reťazce presúvajú, je toto veľmi častým problémom. Web2py úspešne rieši tento problém, vrátane správneho prekódovania reťazcov do UTF-16 pre SQL Server.

Druhá časť internacionalizácie je podpora viacjazyčných rozhraní. V tejto oblasti web2py integruje komponent na tvorbu prekladov rozhraní do viacerých jazykov. Nevyhnutná zmena v kóde samotnej aplikácie je minimálna, stačí pridať volanie na `T()` konštruktor. Napríklad slovenskú aplikáciu reprezentovanú kontrolerom

```
def pozdrav():
    return "Pekny den"
```

možno zmeniť na viacjazyčnú jednoduchou zmenou

```
def pozdrav():
    return T("Pekny den")
```

a pridaním prekladov pre jednotlivé jazyky.

2.3 MVC

Tým, že web2py bol vytvorený viac než 10 rokov po nástupe svetového Internetu, jeho autor mal možnosť poučiť sa z problémov skorších frameworkov. A túto možnosť aj využil. Používa MVC (model-view-controller) postup popularizovaný frameworkom Ruby on Rails. Z výučbového hľadiska tento postup umožňuje študentom rozdeliť webovskú aplikáciu do troch samostatných častí:

- *model*, ktorý reprezentuje štruktúru dát,
- *kontroler*, v ktorom aplikačná logika manipuluje dáta,
- *pohľad (angl. view)*, ktorý formátuje dáta do konečnej XHTML prezentácie.

Tu je príklad rozdelenia aplikácie na nahrávanie obrázkov na model, kontroler a pohľad.

```
# model
db=SQLDB('sqlite://storage.db')
db.define_table('image',
                db.Field('name'),
                db.Field('file','upload'))

# kontroler
def index():
    form = SQLFORM(db.image)
    if form.accepts(request.vars, session):
        response.flash = 'obrazok bol nahraty'
    return dict(form = form)

# pohlad
{{extend 'layout.html'}}
<h1>Upload obrazka</h1>
{{= form}}
```

Prevzatie kontroly dát a štruktúry dát webovskou aplikáciou z veľkej časti odstraňuje problém nazývaný ‘object-relational impedance mismatch’ [7]. Samozrejme, toto riešenie má svoju cenu, a tou je horšia integrácia s existujúcimi relačnými databázami. Napríklad, častou sťažnosťou na diskusných fórach je to, že web2py vyžaduje, aby tabuľky mali primárny kľúč nazvaný `id`. Na jednej strane takýto umelý kľúč môže byť považovaný za dobrú programátorsku praktiku. Na druhej strane je nerealistické očakávať od administrátorov existujúcich relačných databáz, aby takúto požiadavku splnili.

2.4 Riešenie problémov s SQL

Nedostatky jazyka SQL sú všeobecne známe, praktické aj teoretické (napr. [13]). Snáď najväčším praktickým je balkanizácia implementácií tohto jazyka, ktorá pod rúškom uvádzania nových funkcionalít prakticky znemožňuje zákazníkom prenos aplikácie z jedného systému riadenia bázy dát na druhý. Odtiaľ pochádza popularita zjednodušujúcich unifikujúcich technológií ako sú ODBC a Perl DBI, a odmietnutie komplexných technológií ako sú OLE DB.

PHP má snáď najväčší problém s SQL, pretože priam nabáda programátorov na vytváranie priestoru pre útoky pomocou SQL injection. Java je na tom lepšie, často využíva ORM frameworky ako je Hibernate na odbremenenie programátora od SQL jazyka. Takisto Microsoft s technológiou LINQ atakuje problém s SQL v aplikáciach, poskytujúc programátorom unifikovaný jazyk na manipuláciu dát z relačnej databázy, XML alebo pamätových štruktúr. Trend zakrývať variabilitu SQL je zjavný, a implementáciu DAL komponentu vo web2py možno považovať za prirodzené riešenie. Tento komponent umožňuje nielen zredukovať výučbu nevyhnutných relačných technológií z jedného semestra na jednu hodinu, ale takisto umožňuje použiť web2py na Google App Engine [1], ktorý nepoužíva relačnú technológiu.

2.5 Krivka náročnosti

Ako je napísané na wiki stránke web2py [5], tento framework sa líši od ostatných tým, že bol navrhnutý ako výučbový nástroj na DePaul University a preto prirodzene jeho krivka náročnosti je veľmi mierna. Je triviálne začať s ním pracovať (dokonca nepotrebuje ani inštaláciu, alebo administrátorské práva), a poskytuje webovské rozhrania na vývoj, debugovanie, testovanie, správu a administráciu bez použitia ďalších nástrojov. Pritom jeho celková veľkosť je približne 10 MB.

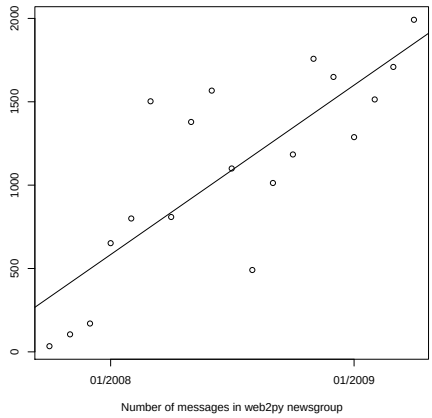
Jedným z hlavných problémov súčasných prístupov k výučbe je množstvo vedomostí, ktoré študent musí absorbovať, než je schopný ich skombinovať na tvorbu interaktívnej webovskej aplikácie. A pritom nie je principiálny dôvod, aby to tak bolo. Toto jasne demonštruje web2py na stránke [4] s 50 jednoduchými príkladmi, kde s minimálnym množstvom kódu (pozri Tabuľku 1) sa implementujú bežné úlohy. Táto jednoduchosť je podľa nášho názoru kriticky dôležitá pre výučbu.

2.6 Použitie v praxi

Na rozdiel od komerčného softvéru nie je možné vyhodnotiť použitie web2py v praxi v peňažnom meradle. Existujú však nepriame meradlá, ako je napríklad aktivita na diskusnom fóre web2py, ktorá je solídne rastúca. Obrázok 1 ukazuje nárast počtu príspevkov na fóre doplnený o priamku vypočítanú metódou lineárnej regresie. Poznamenávame, že nárast koncom roku 2008 je pravdepodobne následok slashdot efektu, keď web2py bolo priaznivo hodnotené na `slashdot.com`. Hoci tento efekt pominul, nárast príspevkov neprestáva.

Tab. 1: Počet riadkov v modeli (M), kontroleri (C) a pohľade (V) pre vzorové príklady zo stránky `http://www.web2py.com/examples/default/examples`

Príklad	M	C	V	Príklad	M	C	V
Hello world		2		Hello world (viacjazyčný)		2	
Hello world + pohľad		2	2	Zmena pohľadu		2	
HTML elementy		2		Hello world + oznam		3	
Zobrazenie request, session		2		Redirekcia		2	
HTTP výnimka		2		Python výnimka		3	
Zmena typu response		4		Použitie JSON		2	
Konvertovanie na RTF		9		Spracovanie RSS prúdu		20	
Konvertovanie WIKI syntaxe		9		Počítadlo pomocou session		4	6
Zobrazenie premenných		1	4	Cyklus v šablóne		1	6
if-then v šablóne		1	12	Výnimka v šablóne		1	8
Funkcia v šablóne		1	7	Automatický escape		1	5
Potlačenie escape		2	5	Automatické formátovanie		1	5
Rozšírenie šablóny		6	3	Rozšírenie šablóny		6	3
Základná šablóna		6	3	Formulár		14	
Model	34			Vloženie užívateľa		14	6
Vloženie psa		7	6	Registrácia produktu		6	6
Nákupný formulár		30	7	Odstránenie kúpy		3	
Zmena dát		3		Download obrázku		2	
Cache 1		4		Cache 2		4	
Cache 3		5		Cache 4		5	
Cache 5		5		Cache 6		6	
Cache 7		6		AJAX 1		8	12
AJAX 2		3		AJAX 3		2	8
Testovanie		8		Príklad prúdu		2	
XML-RPC		15					



Obr. 1: Počet príspevok diskusného fóra web2py

2.7 AJAX

Pod termínom *Web 2.0* sa chápe nová generácia interaktívnych webovských aplikácií, ktoré sami iniciujú komunikáciu s webovským serverom. Typickým príkladom takejto aplikácie je Gmail od spoločnosti Google. Vývoj takýchto aplikácií je veľmi náročný, pretože vyžaduje kombináciu serverovských technológií spolu s programovaním v Javascripte na klientovi. Jedným, dobre známym riešením je využitie Google Web Toolkitu (GWT), ktoré je založené na programovaní aplikácie v jazyku Java. Menej známe je to, že existuje Pythonovská implementácia GWT nazvaná Pyjamas, a táto sa dá použiť na implementáciu AJAX vo web2py [6].

2.8 Integrácia

Programátori preferujú, ak všetky vertikálne komponenty platformy pochádzajú od jedného dodávateľa, či už je to Microsoft (Visual Studio + IIS + .NET framework) alebo autor Linuxovskej distribúcie (Eclipse + MySQL + PHP). Integračné testovanie vykonané týmto dodávateľom totiž ušetrí najmä laickejších užívateľov od ťažko riešiteľných problémov s interoperabilitou komponentov. V tejto oblasti web2py poskytuje bohatý balík, ktorý obsahuje webovský server cherrypy, systém riadenia bázy dát sqlite, konfiguračné riadenie, administratívne rozhranie vrátane lokalizačného rozhrania, ako i logovanie a zobrazovanie výnimok.

2.9 Dokumentácia

Na domovskej stránke `www.web2py.com` sa nachádza množstvo príkladov, tutoriálov a návodov ako pracovať s web2py. Aktívne je takisto diskusné fórum `groups.google.com/group/web2py`. Autor frameworku, Massimo DiPierro, je zároveň autorom knihy [8] o web2py, ktorú možno považovať za autoritatívnu referenciu. V papierovom vydaní sa dá kúpiť za 50 €, v elektronickej forme za 10 €. Tu podávame stručný prehľad obsahu.

Kniha má 256 strán a je členená do 10 kapitol. Po všeobecnej úvodnej kapitole nasleduje 20 stranová kapitola o jazyku Python. Najdôležitejšie na prácu s web2py sú kapitoly 3 až 7.

V tretej kapitole (40 strán) autor krok za krokom buduje aplikácie, aby prakticky ukázal používanie frameworku. Aplikácie sú rastúcej komplexity, od jednoduchej aplikácie na výpis reťazca, cez blogovaciu aplikáciu s obrázkami až po WIKI aplikáciu. Štvrtá kapitola (40 strán) vysvetľuje celkovú štruktúru web2py, ako sa mapujú URL, a základné objekty: request, response, session, cache, internacionalizáciu a všeobecný workflow. V piatej kapitole sú vysvetlené šablóny na integráciu HTML s Pythonom. Šiesta kapitola sa zaoberá DAL, komponentom na prácu s databázami a siedma kapitola preberá webovské formuláre.

V ôsmej kapitole sa autor venuje implementácii AJAX, v deviatej ukazuje ako uvádzať web2py aplikácie do prevádzky vrátane Google App Engine, a desiata adresuje rôzne webovské technológie ako sú XML-RPC, JSON, RSS/ATOM, RTF, Flash, Email apod.

3 Porovnanie s PHP

Je všeobecne známe, že programovanie s PHP, napriek jeho rozšírenosti, je náročné (napr. [2]). V tejto časti článku ukážeme priame porovnanie web2py so Zend frameworkom, ktorý je tiež navrhnutý ako MVC framework. Na porovnanie sme implementovali kompletnú blogovaciu aplikáciu vo web2py aj Zende.

3.1 Porovnanie modelu

V Zend frameworku sa používa externá databáza a preto programátor potrebuje na rozdiel od web2py znalosti o vytváraní relačných databáz. Kód vo web2py bol približne o 50% kratší.

3.2 Porovnanie kontroleru

Každý kontroler v Zende musí byť definovaný ako trieda odvodená od Zend_Controller_Action. Na vytvorenie formulára na editáciu sa používa metóda addElement(). Spracovanie tohto formuláru vyžaduje manuálne kódovanie, zatiaľ čo vo web2py sa formulár vytvorí aj spracuje v niekoľkých riadkoch kódu.

3.3 Porovnanie pohľadu

Vo frameworku Zend sa vytvára HTML stránka skombinovaním statického obsahu s výsledkom PHP fragmentov. V porovnaní s týmto prístupom, web2py vedie študenta k vytváraniu správne formovaných XHTML stránok tým, že preddefinuje triedy pre HTML elementy. Toto má za nepriamy následok prevenciu cross-site útokov na webové aplikácie, čo je rizikom pre menej skúsených programátorov.

4 Záver

V tomto článku sme sa snažili prezentovať argumenty, ktoré nás viedli k rozhodnutiu používať web2py na výučbu študentov na našej univerzite. Pre nás najdôležitejším faktorom bol fakt, že sa jednoducho implementujú jednoduché webovské stránky, no popritom študent nie je obmedzený v možnosti rozširovať webovskú aplikáciu. Hoci sa web2py odlišuje v niektorých aspektoch od momentálne najpoužívanejších frameworkov, na ktoré študent narazí v praxi, veríme, že je veľmi dobre použiteľný pri výučbe webovského programovania, a to najmä v nasledujúcich prípadoch

- pri výučbe na stredných školách, kde je obmedzený hodinový rozsah
- na vysokých školách ako publikačnú technológiu pre neinformatikov
- ako open source alternatíva k tradičným Java/.NET frameworkom
- ako aplikáciu v rámci výučby jazyka Python

Za hlavný nedostatkok z hľadiska výučby možno zaradiť nedostatok študijných materiálov o web2py v slovenskom jazyku.

Literatúra

[1] Google App Engine `http://code.google.com/intl/sk/appengine/`

[2] Blurring of MVC lines: Programming the Web Browser `http://advogato.org/article/993.html`

[3] Comparison of web application frameworks, `http://en.wikipedia.org/wiki/List_of_web_application_frameworks`

[4] web2py Examples, `http://www.web2py.com/examples/default/examples`

[5] web2py, `http://en.wikipedia.org/wiki/Web2py`

[6] Using pyjamas with web2py, http://mdp.cti.depaul.edu/AlterEgo/default/show/203

[7] Object-relational impedance mismatch, http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

[8] DI PIERRO, M.: Web2Py Manual, Wiley 2008; http://www.lulu.com/content/4968879

[9] Want to start a startup? http://www.paulgraham.com/avg.html

[10] web2py Developers, http://www.web2py.com/examples/default/who

[11] TIOBE Programming Community Index, http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[12] Strong versus Weak Typing, A Conversation with Guido van Rossum, http://www.artima.com/intv/strongweakP.html

[13] HUGH, D. – DATE, C. J. *The third manifesto*. ACM SIGMOD Record (New York, NY, USA: ACM Press) 24 (1): 39–49, (March 1995).

Kontaktná adresa

Ján KARABÁŠ, (Mgr., PhD.),
Univerzita Mateja Bela, Ďumbierska 1,
974 11 Banská Bystrica,
karabas@savbb.sk

Jozef SILÁČI, (Mgr.),
Katedra informatiky, Univerzita Mateja Bela, Tajovského 40,
974 11 Banská Bystrica,
silaci@fpv.umb.sk

Ondrej ŠUCH, (PhD.),
Katedra informatiky, Univerzita Mateja Bela, Tajovského 40,
974 11 Banská Bystrica,
such@fpv.umb.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

LMS MOODLE JAKO NÁSTROJ PRO VZDĚLÁVÁNÍ UČITELŮ V MORAVSKOSLEZSKÉM KRAJI

KOZÁKOVÁ, Blanka, (ČR)

1 Úvod

Ve vzdělávací instituci Krajské zařízení pro další vzdělávání pedagogických pracovníků a informační centrum, Nový Jičín, příspěvková organizace realizujeme od podzimu 2008 projekt Perspektiva 2010 na podporu rozvoje digitální gramotnosti učitelů základních a středních škol Moravskoslezského kraje (dále jen MSK). Většina kurzů v tomto projektu má také e-learningovou část (v rozsahu od 4 do 10 h). Hledali jsme tedy vhodný nástroj, pomocí kterého bychom poskytli učitelům Learning Management System (dále jen LMS) [1] a zároveň je motivovali k jeho nasazení ve své vlastní škole do výuky. Vzhledem k dobré zkušenosti s Moodle jsme zvolili tento LMS a to jak pro řízení vzdělávání učitelů, tak jako datové úložiště studijních materiálů a komunikační prostředí lektorského týmu. Po půl roce používání můžeme objektivně posoudit výhody resp. omezení tohoto nástroje a obhájit jeho volbu z množiny informačních a komunikačních technologií (dále jen ICT).

2 Proč nasazení LMS Moodle

V průběhu tří let prochází vzděláváním na podporu digitální gramotnosti 2,5 tis. účastníků, vzdělávání jim poskytuje šedesátičlenný lektorský tým a organizačně je zajišťují tři pracovníci – to vše na území Moravskoslezského kraje tj. na ploše 5,5 tis. km^2. Z těchto údajů je zřejmé, že velkou část organizace vzdělávání je nutné řešit pomocí informačních a komunikačních technologií.

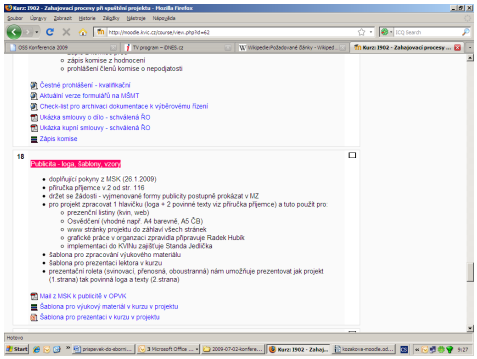
Z technického hlediska je zvolené řešení velmi úsporné, využili jsme server, který dosloužil na původní pozici a má následující parametry:

1. IBM xSeries 235, Intel Xeon 2 GHz, 1,5 GB RAM ECC, HDD 3x SCSI 36 GB in IBM ServRaid 5i,
2. operační systém na serveru RedHat Linux ES4,
3. 2 Mbps synchronní přístup serveru do internetu,
4. datum nákupu serveru – 30. 11. 2002 (do roku 2007 sloužil k jinému účelu).

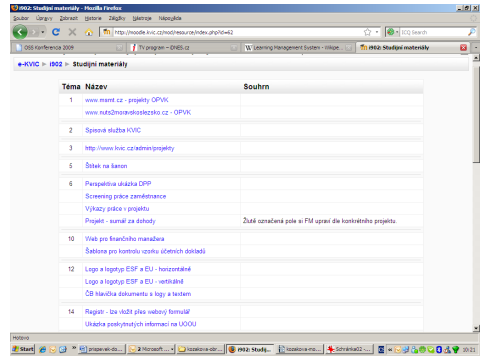
Z personálního hlediska je pro správu tohoto serveru nutný pracovník se znalostí operačního systému Linux v příslušné distribuci a pracovník, který má v Moodle pozici administrátora. Obě pozice řešíme externími pracovníky.

3 LMS Moodle jako datové úložiště

Nejjednodušším modelem pro nasazení tohoto řešení je Content Management System (dále jen CMS) [2]. Velké množství výukových materiálů, prezentace lektorů, šablony pro odevzdávání prací účastníků – tohle všechno je obsah, který potřebujeme přehledným a dohledatelným způsobem organizovat pro více než šedesát kurzů. A tuto službu Moodle v přiměřené formě poskytuje. Můžeme se podívat, jak vypadá např. kurz *Řízení projektů s ICT*. Řešení na prvním z obrázků zobrazuje třídění dokumentů do jednotlivých článků kde je popis dokumentů a v závěru jsou vloženy šablony, texty, prezentace a další zdroje.



Obr. 1: Moodle jako CMS – roztřídění dokumentů dle témat v kurzu



Obr. 2: Moodle jako CMS – studijní materiály ke kurzu tvoří datový sklad

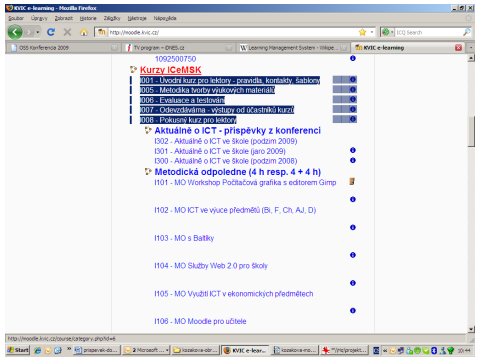
Na obrázku 2 vidíme, jak vypadá přehled všech vložených elektronických materiálů v jednom datovém skladu u stejného kurzu. Pokud je dostatečně jednoznačný *Název* materiálu, není třeba vyplnit informaci v poli *Souhrn*.

V závěru rozsáhlejších kurzů vkládají lektoři do datového skladu výstupy účastníků, jedná se o elektronické materiály, které využijí také další účastníci kurzů. Soubory jsou lektorem umístěny dle tématu kurzu, velikost vkládaných souborů je omezena na 20 MB/soubor,

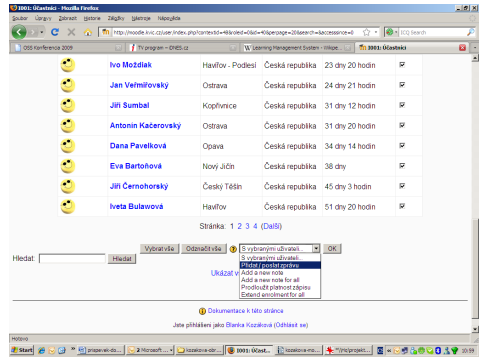
což až na výjimky jako jsou např. výstupy z kurzů na téma digitální fotografie, digitální film, je velikost přiměřená.

4 Kdy lektor řekne ano

Podmínkou nutnou pro realizaci takového řešení je zejména podpora ze strany lektorského týmu (účastníci dostávají přidanou hodnotu, zatímco lektoři mají dojem, že jde o práci navíc). Abychom motivovali lektory pro aktivní využívání Moodle, byla zřízena sekce určená pro administrativu lektora a také pomocné kurzy pro jejich práci.



Obr. 3: Kurzy v Moodle pro podporu práce lektora



Obr. 4: E-mail o změně dokumentu v kurzu

V kurzu *Úvodní kurz pro lektory – pravidla, kontakty, šablony* je k dispozici povinná dokumentace, které jsou lektoři pro realizaci kurzů povinni používat. Výhoda jednoho umístění je zřejmá, jakoukoliv změnu v povinné dokumentaci provádíme na jednom místě. Lektorům tuto změnu avizujeme hromadným mailem.

Pro přístup lektorů do jednotlivých kurzů využíváme tři úrovně oprávnění – role *Student, Lektor (jen čtení)* a *Lektor*. V kurzu, ze kterého pouze čerpají informace, šablony a pokyny mají lektoři přístup v roli *Student*. Editaci těchto kurzů zajišťují organizační pracovníci projektu, vkládají zde dokumenty a zprávy pro lektorský tým, spouštějí diskusní fóra, chaty na společná témata. Tato část Moodle je jedním z hlavních nástrojů pro řízení lektorského týmu.

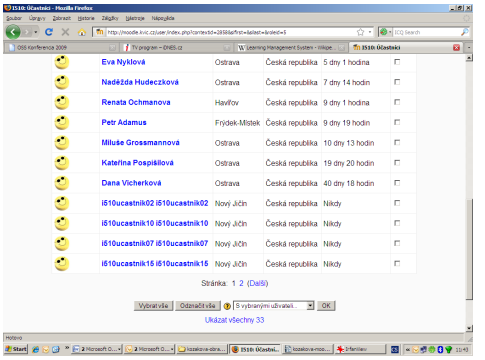
V kurzu, kde mají lektoři možnost vše si vyzkoušet, změnit nastavení, ověřit služby, mají přístup v roli *Lektor*. Např. se jedná o kurz *Pokusný kurz pro lektory*, kde se lektoři učí s prostředím Moodle efektivně pracovat. Dále je to kurz *Odevzdávárna – výstupy od účastníků kurzů*, do kterého odevzdávají výstupy účastníků kurzů, zde rovněž potřebují právo zápisu atd. Do ostatních kurzů mají přístup *Lektor (jen čtení)*.

Obsah každého z kurzů je dán akreditací schválenou na Ministerstvu školství, mládeže a tělovýchovy. Toto vymezení kurzu dodržuje jeho garant a lektorský tým, který k tomuto kurzu přísluší (týmy jsou od jednoho až po více než 20 lektorů). Garant má v kurzu roli

Lektor s právem zápisu a udržuje prostředí kurzu v aktuálním stavu např. z podkladových materiálů zaslaných členy lektorského týmu. Lektoři mají přístup *Lektor (jen čtení),* který je pro řízení výuky dostačující.

Přístup účastníků do svého kurzu si lektor volí ze dvou možností, v obou případech jde o přístup v roli *Student*. První z nich jsou účty obecné, které zůstávají v kurzu přiřazeny po celou dobu trvání projektu. Každý kurz má 16 takových obecných účtů – v průběhu prezenční části kurzu tedy účastníci mohou využívat chat, není aktivní e-mail. Pokud se lektor rozhodne využít celé portfolio služeb Moodle, jsou účastníkům zřízeny osobní účty a cca měsíc po skončení kurzu jsou ze skupiny *Student* odebráni (v Moodle zůstávají). Při nastavení skupiny účastníků do kurzu spolupracuje s lektorem organizátor. Organizační pracovníci projektu mají ve všech kurzech roli *Lektor* a mohou tak kdykoliv a každému z lektorů pomoci.

Obr. 5: Moodle – kurz pro administrativu lektora



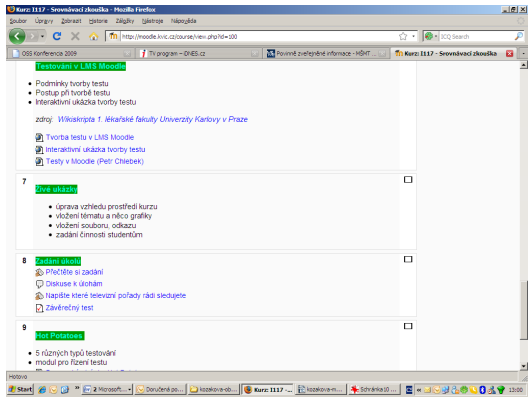
Obr. 6: Účty účastníků kurzu

5 Účastník kurzů v prostředí LMS

V první části kurzu obdrží každý z účastníků přístup v roli *Student*. Od této chvíle lektor s účastníky průběžně aktivně využívá prostředí Moodle ke stahování souborů, odkazování na webové informační zdroje, k plnění úkolů ať jednotlivce nebo skupiny. Přiměřeným tempem a intuitivním způsobem organizace prostředí kurzu se účastníci sžívají s prostředím e-learningu.

Přístup k LMS využívá maximálně 100 účastníků současně. Toto číslo není explicitně nastaveno na serveru, ale je odvozeno z počtu paralelně realizovaných kurzů v jednom dni a z naší zkušenosti s využíváním výukových materiálů v průběhu víkendů.

Pro případ, že by se účastník chtěl v předstihu nebo následně a hlouběji seznámit s Moodle a jeho využitím ve vyučovacím procesu, má možnost navštívit samostatný kurz *Moodle pro učitele.*



Obr. 7: Kurz je ukončen ověřením kvality na výstupu

V průběhu vzdělávání využívá účastník výhody e-learningu jako např. možnost komunikace s lektorem i mimo dobu prezenční části kurzu, má stále k dispozici výukové materiály s jistou mírou interaktivity (např. průběžné testování), má možnost srovnávat si své výstupy s ostatními účastníky v průběhu jejich tvorby. Lektory vedeme k ukládání materiálů buď ve formátech `*.pdf` resp. jako www stránky, také účastníci jsou vedeni ke zpracování výstupů ve formátu `*.pdf`.

Asi největší výhodou tohoto nástroje při vzdělávání dospělých je podpora individualizované výuky. Do kurzů vstupují učitelé s rozdílnou úrovní znalostí a dovedností v oblasti ICT – hovoříme zde o libovolném učiteli ze základních a středních škol v Moravskoslezském kraji, který do vzdělávání může vstupovat. V průběhu výuky je snahou lektora pomoci zdatnějším účastníkům dále se rozvíjet, účastníkům s menšími zkušenostmi posunout se znalostně a dovednostně na minimální požadovanou úroveň na výstupu z kurzu.

6 Závěr

Po půlročním provozu LMS Moodle v projektu Perspektiva 2010 pro podporu rozvoje digitální gramotnosti učitelů je možné uzavřít naši zkušenost s ověřenými závěry. Pozitivní zjištění jsou následující:

1. neplatíme za software, úhrada je za správu a konektivitu,
2. jedno prostředí pro lektory i účastníky,
3. rozšíření tohoto LMS ve školách v MSK,
4. intuitivní orientace a použití kurzu účastníky,
5. přehledný, organizovaný a garantovaný obsah kurzů,
6. dostupná a stejně kvalitní podpora všem, účastníkům z MSK 7*24 h.

Omezení vnímáme v těchto oblastech:

1. počítač s operačním systémem Linux (lze řešit outsourcingem),
2. nutnost promyslet a vhodně nastavit pravidla správy prostředí,
3. potřeba stáleho vedení garanta kurzu a lektorů k údržbě obsahu a volbě vhodných formátů souborů,
4. sběr výstupů větších než 20 MB pomocí DVD resp. veřejným datovým úložištěm.

Pokud zvážíme přínosy a omezení LMS Moodle, lze konstatovat, že se jednalo o vhodnou volbu pro nasazení v uváděném projektu. Dotazováním mezi účastníky jsme ověřili, že více než 60 % z účastníků se po absolvování kurzů aktivně zapojilo do využívání LMS ve své škole (tam, kde je LMS instalován). Využití Moodle při vzdělávání učitelů má v tomto případě jako přidanou hodnotu aplikaci vlastní zkušenosti s LMS do pedagogické praxe na základních a středních školách v MSK.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ
Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

Literatúra

[1] *Learning Management System*, Wikipedia, staženo 2009-06-07 na URL adrese: `http://cs.wikipedia.org/wiki/Learning_Management_System`

[2] *Content Management System,* Wikipedia, staženo 2009-06-07 na URL adrese: `http://cs.wikipedia.org/wiki/Syst%C3%A9m_pro_spr%C3%A1vu_obsahu`

Kontaktná adresa
Blanka KOZÁKOVÁ (Mgr.),
Krajské zařízení pro DVPP a informační centrum, Nový Jičín, příspěvková organizace
Štefánikova 7, 741 11 Nový Jičín, ČR
blanka.kozakova@kvic.cz

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

TERMINÁLOVÉ ŘEŠENÍ V UNIVERZITNÍM PROSTŘEDÍ

KŘÍŽ, Pavel (CZ); SEBERA, Václav (CZ)

1 Úvod

Terminálové řešení (dále TŘ) v současnosti patří mezi často diskutované pojmy v souvislosti s optimalizací infrastruktury výpočetních systémů. Klasické řešení budované na základě individuálních počítačů se s rostoucím počtem klientů dostává do pozadí a dává tak prostor pro nízkonákladové terminálové řešení. Ekonomické důvody tak udávají pro řešení server – tenký klient dynamický vzestup. Mezi další výhody patří přínosy ekologické, ergonomické a bezpečnostní.

Lesnická a dřevařská fakulta Mendelovy zemědělské a lesnické univerzity v Brně spustila projekt terminálového řešení do provozu v roce 2007 [1]. Projekt byl původně zaměřen na uživatele, kteří vyžadují vysoký výpočetní výkon, nicméně plány do budoucna počítaly i s rozšířením řešení pro celou fakultu mimo Ústav nauky o dřevě, kde původně vzniklo. Technologicky odzkoušené a kompaktní řešení bylo v roce 2008 nasazeno i na ostatních pracovištích, kde v mnohých případech nahradilo běžné stolní počítače. V současné době terminály využívá široké spektrum zaměstnanců i studentů v počítačových laboratořích v celkovém počtu 50 kusů. Počet současně připojených uživatelů je velmi orientační a je závislý na aktuální situaci a prioritách jednotlivých výzkumných a pedagogických skupin.

2 Základní koncepce a technické vybavení

Koncepce je založena na infrastruktuře, která poskytuje velmi šetrné a zároveň ekonomicky výhodné řešení. Zvolený přístup umožňuje snížit celkové množství administrativní práce a nabízí zároveň stabilní prostředí, které je schopno zabezpečit standardní uživatelské požadavky.

Technické vybavení je tvořeno hardwarem společnosti Sun Microsystems a je zabezpečováno (zajišťováno) fakultou. Základní kámen tvoří server SunFire X2200 M2, osazený dvěma procesory Dual Core AMD Opteron 2222 a operační pamětí 12GB, nazývaný jako gateway. Server s tímto označením vyřizuje veškerou komunikaci s klientskými terminály, stará se o uživatelské relace a předává požadavky na aplikační server, který technicky zabezpečuje server SunFire X4600 M2, osazený čtyřmi procesory Dual Core AMD Opteron 8218 a operační pamětí 16 GB. Uvedená komponenta tvoří první terminálovou větev, která vyřizuje požadavky zaměstnancům fakulty. Pro zabezpečení nepřetržitého provozu je k dispozici druhá paralelní terminálová větev stejných parametrů. Tím je zajištěn chod terminálů v případě poruchy či výpadku některé ze služeb.

Uvedená koncepce nabízí jednoduchý způsob řešení problematických otázek bezpečnosti a zálohování. Součástí terminálového řešení jsou jak jednotky bezpečnostních serverů, tak diskové pole StorageTek 2540 o kapacitě 2 TB, které tak poskytuje dostatečný prostor pro práci s velkými objemy dat i jejich archivaci.

Koncept po softwarové stránce využívá operačního systému GNU/Linux v distribuci CentOS. Již zmiňované gateway servery využívají speciální aplikace Sun Ray Software, která obstarává základní komunikaci s terminály, Sun Ray klienty.

3 Učebna numerických simulací

Učebna náročných numerických simulací se stala pilotním projektem pro zařazení terminálů do výuky. Byla vybudována v roce 2008 za podpory grantové agentury Fondu rozvoje vysokých škol v rámci projektu FRVŠ 398/2008 – Počítačová učebna pro výuku matematického modelování fyzikálních jevů a biomechanických procesů.

Cílem projektu bylo vybudování počítačové učebny pro řešení komplikovaných numerických úloh, objemných databázových operací, zpracování rozsáhlých statistických úloh a symbolických výpočtů. Učebna je sestavena z výkonného zařízení optimalizovaného na distribuované výpočty v prostředí stávajícího i plánovaného vybavení MZLU. Projekt navazuje na již popisovanou základní koncepci terminálového řešení na LDF MZLU [1] v procesu výuky modelování a formulování matematicko-fyzikálních úloh.

Kapacitně učebna disponuje dvaceti pracovními místy, pro učitele je k dispozici jak terminál, tak běžná pracovní stanice s operačním systémem Windows XP 64 bit, která mimo jiné slouží k ovládání interaktivní tabule. Po technické stránce je běh učebny nezávislý na základní koncepci fakultního terminálového řešení, běží na vlastní terminálové větvi. Celý provoz řídí jediný server osazený osmi procesory Dual Core ADM Opteron 8220 a operační pamětí 36 GB. Stejným výpočetním výkonem disponuje i server běžící s operačním systémem Windows Server 2003 Cluster Edition. Tento server umožňuje provozovat aplikace, jejichž podpora není pod platformou GNU/Linux dořešena či podporována. Softwarově je SunRay server vybaven utilitou pro vzdálený přístup, díky které může uživatel terminálu plnohodnotně přistupovat jak k celému serveru, tak k jednotlivým aplikacím.

Obr. 1: Terminálová učebna

Následující výčet předmětů vyučovaných v terminálové učebně jen podtrhuje fakt univerzálnosti daného výukového prostředí:

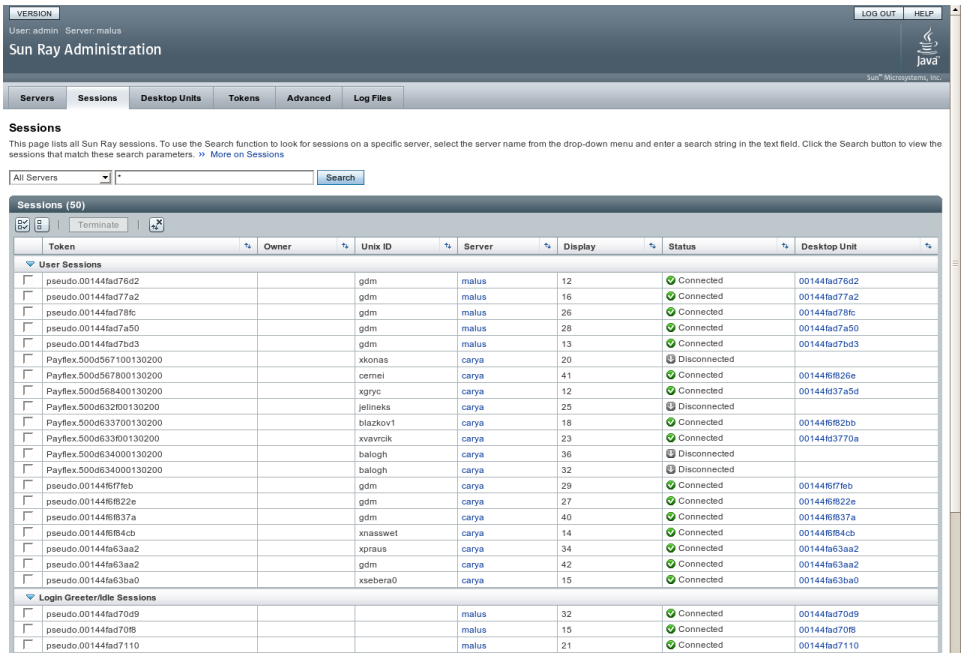
- **CAD vizualizace** – architektonické návrhy a vizualizace interiérů a nábytku v systémech Autodesk VIZ 4 a VariCAD
- **Wood Anatomy**
- **Introduction to Engineering Computing** – Ansys
- **Využití FEM** – Ansys, CFX, Comsol, Matlab
- **Dendrometrie** – Statistica, R, Microsoft Excel
- **Geoinformatika** – kartografické a geodetické úlohy v systémech ArcGIS, Topol, Kokeš, MiSYS, Atlas
- **Matematika** – Maxima, Sage
- **Geometrie** – Cabri , Rhinoceros

4 Pohled správce systému

Správa terminálových systémů je obecně velmi komfortní a přímočará. Díky centralizované administrativě a kvalitně navržené síťové topologii získává správce dostatečný přehled a maximální kontrolu nad celým systémem. Na druhou stranu z pohledu uživatele je správa téměř nulová.

Instalace nových programů se provádí globálně a je plně v kompetenci systémových administrátorů jednotlivých serverů. Touto strategií je zaručen odborný přístup a zejména zamezení instalace nelegálního nebo nevhodného softwaru.

Terminálové řešení Sun Ray disponuje speciálním systémem autentizace pro přenos uživatelských relací mezi klientskými stanicemi. V praxi to znamená nezávislost uživatelské



Obr. 2: Sun Ray Administration

relace na klientském terminálu, kde je uživatel schopen díky své osobní čipové kartě přenést svoji započatou relaci na jakýkoliv terminál v systému.

Součástí software Sun Ray Serveru je Sun Ray Administration – nástroj pro správce systému. Webová aplikace zprostředkovává přehled připojených Sun Ray klientů, uživatelských relací a celkový stav aplikace Sun Ray Server, viz Obr. 2.

Z pohledu správce systému je administrace terminálových systémů velmi pohodlnou záležitostí, ovšem vyžaduje perfektní zacházení a přesnost. Chybným nedopatřením lze způsobit nevratné škody či odstavit celé prostředí mimo provoz.

5 Uživatelský pohled

Pohled uživatele na TŘ je samozřejmě vymezen úrovní jeho znalostí v oblasti IT (SW i HW). Ohlasy jednotlivých uživatelů tedy můžeme rozdělit dle jejich schopností zhruba do tří rozličných kategorií: 1. skupina – uživatelé s minimální znalostí IT (MS Word. . .); 2. skupina – středně pokročilí (samosprávci PC. . .); 3. skupina – velmi pokročilí (administrace GNU/Linuxu. . .). Zvážíme-li ohlasy ze všech těchto skupin za celou dobu, po kterou je TŘ na LDF v provozu, můžeme konstatovat následující:

1. nejobtížnější přechod z PC (Windows) na TŘ je pro 1. a 2. skupinu, avšak zhruba

po 2–4 měsících tyto skupiny pochopily základní strukturu GNU/Linuxu a obsáhly open-source alternativy ke svým předešlým aplikacím a nyní pracují bez větších problémů,

2. uživatelé všech skupin si pochvalují bezproblémovost systému a snadnou doinstalaci potřebné aplikace („zvednutí telefonu“),
3. uživatelé 3. skupiny využívají všech výhod GNU/Linuxu a oceňují výkon serverů v grafických i výpočetních aplikacích (VIZ 4, ANSYS. . .),
4. pedagogové oceňují autentizační karty a snadný přenos své relace do učeben,
5. nevýhody TŘ vidí uživatelé všech skupin podobně a to zejména v oblasti výměny dat pomocí médií – omezené připojení CD/DVD mechanik a problémový chod některých druhů flashdisků, dále omezené akcelerace grafiky (pouze 2D) a pomalejší grafické odezvy.

Všichni uživatelé tedy po určité době nabyli schopnost se orientovat a pracovat s operačním systémem GNU/Linux na takové úrovni, kterou pro svou práci potřebují. Jinými slovy to dokazuje ten fakt, že je-li profesionálně postaráno o administraci operačního systému, nečiní základní práce v GNU/Linuxu potíže ani fakultním administrativním pracovníkům/cím, kteří k práci v TŘ přistupovali právě s největšími obavami a rezistencí.

6 Závěr

Na LDF MZLU v Brně byla vytvořena progresivně rostoucí výpočetní infrastruktura založená na terminálovém řešení, které uživatelům nabízí unikátní a také unifikované pracovní prostředí jak pro každodenní administrativní a pedagogickou práci, tak i pro komplexní práci vědeckou, resp. programátorskou. Do rámce TŘ spadá i vytvořená terminálová učebna. Zhruba po roce provozu/výuky lze danou koncepci učebny doporučit. Roli nehrají pouze aspekty zvýšení kvality pracovního prostředí (snížení hluku aj.), ale zejména snadná administrace učebny a zamezení instalace nevhodného nebo nelegálního SW. Specifika TŘ samozřejmě doprovází i již zmíněné nevýhody a to zejména v oblasti výměny dat. Z celkového pohledu však TŘ plní všechna očekávání a to jak z pohledu provozu a údržby, tak z pohledu stávající a budoucí koncepce informačních technologií na celé fakultě.

Literatúra

[1] KOŇAS, P. – SEBERA, V.: *Terminálové řešení, LinFUN a Open Source na LDF*. In Proceedings of 7th International Conference APLIMAT 2008 – Part V Open source Software in Research and Education. Bratislava: Department of Mathematics, Faculty of Mechanical Engineering, Slovak University of Technology, 2008, s. 35–45, ISBN 978-80-89313-04-4

[2] Webová stránka Ústavu nauky o dřevě: `http://wood.mendelu.cz/cz/pcucebna/`

Kontaktní adresa

Pavel KŘÍŽ (RNDr.),
Lesnická a dřevařská fakulta MZLU v Brně,
Zemědělská 3,
613 00 Brno,
kriz@mendelu.cz

Václav SEBERA (Ing.),
Ústav nauky o dřevě, LDF MZLU v Brně,
Zemědělská 3,
613 00 Brno,
xsebera0@mendelu.cz

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

MATHEMATICAL ASSISTANT ON WEB
SKLÁDANKA ZE SVOBODNÝCH MATEMATICKÝCH PROGRAMŮ

MAŘÍK, Robert, (CZ)

1 Úvod

Cílem tohoto příspěvku je představit základní myšlenky, na nichž je vybudována online aplikace Mathematical Assistant on Web (MAW, [4]). Tato aplikace automatizuje a sama provádí řešení těch úloh základních kurzů matematické analýzy, které jsou do jisté míry jen mechanickým sledem předem daných kroků. Vstupem jsou vždy data pro zadání jisté úlohy a výstupem řešení úlohy, včetně jednotlivých mezikroků. Například při úloze na výpočet derivace je zadáním funkce jedné nebo dvou proměnných a proměnná, podle které se má derivovat. Výstupem je PDF soubor, ve kterém jsou naznačeny jednotlivé kroky potřebné pro výpočet derivace zadané funkce a mezivýsledky po aplikaci těchto jednotlivých kroků.

Aplikace je vybudována se snahou využít v maximální možné míře již hotové Open-Source nástroje pro řešení jednotlivých dílčích úloh. Zpracování těchto úloh je realizováno vždy voláním externího programu, který dílčí úlohu umí vyřešit. Tento přístup je zcela v duchu unixové filozofie používání jednoúčelových nástrojů, které sice řeší třeba jenom jeden problém, řeší ho však efektivně a dobře.

Aplikace je volně přístupná na adrese `http://www.mendelu.cz/user/marik/maw` a její zdrojové kódy jsou k dispozici na `http://mathassistant.sourceforge.net`. Je v provozu od listopadu 2007 a postupně vylepšována, doplňována a opravována. Ve zkouškovém období registrujeme cca 3000 přístupů denně.

Problematice výpočtů v prostředí webového prohlížeče jsou věnovány i internetové verze komerčních programů Mathematica a Maple. V naprosté většině výukových aplikací je však nasazení těchto produktů střelba dělovou koulí proti mouše. Proto v závěru uvedeme jednoduchý praktický příklad, který umožňuje vzít z parametru PHP skriptu zadání, zpracovat

toto zadání matematickým softwarem a výsledek zobrazit ve webovém prohlížeči. To vše s minimální znalostí programování. (Dobrá znalost systému počítačové algebry, který tyto výpočty provádí, je naopak nezbytná.)

2 Opičí dráha jménem MAW

Skutečnost, že výpočet některých typů úloh vysokoškolské matematiky je zcela mechanický, je jistě nepřehlédnutelná. Při řešení některých úloh pomocí systémů počítačové algebry a zejména při vhodně navržených zápisnících často stačí pouze měnit vstupní údaje na začátku zápisníku a po přepočítání všech příkazů dostáváme odpovídající výstup. MAW je založen na podobné myšlence, přichází však ještě s jednou další a podstatnou vlastností: je možné jej používat pouze prostřednictvím webového prohlížeče, bez instalace jakéhokoliv programu na lokální počítač a bez nutnosti učit se program ovládat nebo v něm zapisovat příkazy. Tím je systém zpřístupněn mnohem většímu počtu zájemců, než by tomu bylo v případě offline zápisníku pro některý ze systémů počítačové algebry. Díky tomu je systém otestován na velice rozsáhlém vzorku matematických problémů. Jistou nevýhodou je nutnost udržování serveru který provádí požadované výpočty a nutnost věnovat pozornost otázkám bezpečnosti a výkonu, aby jeden uživatel neomezil (ať již vědomě či nechtěně) v používání systému ostatní uživatele. V následujících odstavcích se zaměřím na popis procesu, kterým vstupní data procházejí při zpracování systémem MAW.

Cestu vstupních údajů MAWem je možno přirovnat k opičí dráze, kterou musí vstupní data proběhnout, aby se z nich na konci dráhy stal vzorově vyřešený příklad. Hlídač na vstupu provádí základní selekci a jeho úkolem je nepustit na dráhu zraněné, nemocné nebo celkově slabé opičky, u nichž je na první pohled zřejmé, že dráhu neproběhnou bez úhony. Jestliže se opička zdá být v pořádku, je vpuštěna na dráhu tvořenou řetězcem překážek – programů, které se podílejí na zpracování úlohy, které způsobují nejdelší prodlevy ve zpracování a u kterých je největší riziko neúspěchu. Tyto překážky (OpenSource programy) byly získány zdarma a v některých případech navíc výrobcem přímo upraveny na míru pro naši dráhu[1]. Protože opička je po proběhnutí dráhy zadýchaná a rozcuchaná, je vhodné ji pro závěrečné defilé úspěšných běžců učesat a celkově poupravit (TEXem) případné nedostatky ve vzhledu.

2.1 Vstup jen pro zvané

Systém MAW je určen zejména pro slabší studenty matematiky. Je proto nutno očekávat, že uživatelé budou značně neohrabaní při zápisu matematických výrazů. Aby se co největší procento chyb odhalilo ještě před vypuštěním opičky na dráhu, jsou vstupní data testována pomocí regulárních výrazů a seznamu nezvaných hostů, aby byly odchyceny například

- nebezpečné příkazy, zneužitelné pro útok na server;

[1] Například díky ochotě autorů programu `formconv` je MAW v současnosti jediný program, který umí nakreslit graf funkce $y = \sqrt[3]{1-x^2}$ na množině všech reálných čísel.

- zapomenuté závorky vyznačující rozsah působení funkcí (např. `sin x`);
- zapomenutý znak pro násobení (např. `2x+8x^3`) nebo nepárová závorka;
- použití jiných než explicitně povolených funkcí (`erf`, hyperbolické funkce), proměnných a parametrů.

Při tvorbě těchto vstupních filtrů bylo mnoho myšlenek převzato z projektu MaximaPHP [7]. Pokud vstupní data sítem těchto filtrů neprojdou, MAW věnuje úsilí k zajištění nápravy. Toto je realizováno jednak opět pomocí regulárních výrazů (například konverze `2x+8x^3` na `2*x+8*x^3`), jednak pomocí programu `formconv` (například konverze výrazu `sin^3 8x` na `(sin((8*x))^3)`).

2.2 Typy překážek

Opičí dráha je sestavena z překážek typu Maxima ([6], systém počítačové algebry, zajišťuje veškeré matematické výpočty), GNUplot ([3], kresba většiny grafů a obrázků), formconv ([1], převod matematických výrazů mezi formáty pro program Maxima, GNUplot a TEX) a mfpic (kreslení některých obrázků). Jedná se o kvalitní a velkým počtem uživatelů prověřené programy, které patří k nejkvalitnějším nástrojům svého druhu i ve srovnání se svými komerčními alternativami.

Překážka Maxima je jedna z nejtěžších překážek. Protože její zdolání je časově náročné[2], je většina opičích drah postavena tak, aby se na nich těchto překážek vyskytovalo co nejméně, pokud možno pouze jedna (Lagrangeův polynom, Taylorův polynom, výpočet derivací, ...). V případě obzvláště obtížných překážek je hlavní překážce předřazena ještě jedna nebo více lehčích překážek Maxima, které provedou podrobnější selekci než byla provedena vstupními filtry. Například při výpočtu dvojného integrálu $\int_a^b \int_{f(x)}^{g(x)} \phi(x,y)\,\mathrm{d}y\,\mathrm{d}x$ je nutno zkontrolovat, že platí nerovnost $a < b$ a na intervalu (a,b) platí $f(x) \leqq g(x)$. Vstupní data, která touto kontrolou neprojdou, nejsou vpuštěna na další překážku, protože úloha není korektně zadána a případné další výpočty by postrádaly smysluplnou interpretaci.

Další záludnost překážky Maxima spočívá v tom, že opička v překážce někdy zabloudí a ptá se na cestu. Tato situace může nastat například při výpočtu integrálu $\int \frac{1}{x^2+a}\,\mathrm{d}x$. Pro další pokračování výpočtu program Maxima potřebuje informaci, jestli je a kladné, záporné nebo nula. Protože bez další interakce s uživatelem MAWu není možno tuto informaci zjistit, opička by zůstala na překážce viset do nekonečna. Aby k této situaci nedocházelo, má každá překážka typu Maxima svého hlídače[3], který hlídá aby opička při pokusu o zdolání nepřekročila jistý předem daný časový limit. Pokud se po uplynutí limitu opička z překážky nedostane, hlídač opičku sesadí a oznámí neúspěch jejímu majiteli.

[2] Například pouhé spuštění programu Maxima 5.13 (GNU Common Lisp) trvá více než 0.1 s. Spuštění poslední verze programu Maxima 5.18 (CMU Common Lisp) dokonce 0.7 s.

[3] OpenSource skript `timeout`, `http://www.shelldorado.com/scripts/cmds/timeout`

Další překážkou je překážka GNUplot. Protože program GNUplot má poněkud neobvyklou formu zápisu matematických výrazů[4], je této překážce vždy předřazen program pro konverzi matematických výrazů formconv. V některých případech je tato překážka dokonce trojitá a po dvojici formconv+GNUplot ještě následuje jednoduchý filtr v Perlu, jehož úkolem je odstranit z grafů nespojitých funkcí nežádoucí svislé čáry, které se zde někdy objevují.

Při postupném zdolávání překážek opička sbírá certifikáty o jejich zdolání – TEXovské vzorce u překážek Maxima a obrázky u překážek GNUplot.

2.3 Příprava na závěrečné defilé

Po zdolání všech překážek čeká na běžce na konci opičí dráhy vizážista LATEX. Jeho úkolem je upravit pocuchanou a udýchanou opičku do podoby vhodné pro prezentaci na veřejnosti. Jedná se opět o program s otevřeným zdrojovým kódem, který je velice podrobně prověřen obrovskou uživatelskou základnou, sahající od jednotlivců píšících seminární práce až po velká nakladatelství zabývající se tiskem matematických monografií. Pokud při postupném zdolávání překážek opička získala všechny certifikáty (všechny kroky proběhly bez problémů), je tento poslední úkol poměrně snadný a rychle vyřešený. Výsledek může vypadat jako na obrázku 3.

2.4 Zkušenosti provozovatele opičí dráhy

Záznamy o jednotlivých běžcích, snažících se opičí dráhu zdolat, jsou pečlivě evidovány a pokud to čas dovolí, jsou záznamy o jednotlivých výkonech a zejména selháních procházeny a analyzovány. Protože je aplikace volně dostupná na internetu, je v záznamech vždy dostatek materiálu pro testování a pro inspiraci k dalším vylepšením a opravám případných chyb. Vzhledem k velkému počtu zpracování většinou triviálních a stále se opakujících úloh však vzala za své představa autorů, že provozem získají mimo jiné zdroj zajímavých úloh, které budou poté moci použít ve vlastní pedagogické praxi.

Zajímavou a překvapivou zkušeností je, že někteří uživatelé opakovaně vypouštějí na dráhu opičky značně nemocné (například `odmocnina*(x+y/2)`) a ignorují jakékoliv instrukce týkající se zápisu matematických výrazů. Tímto dáváme zcela za pravdu jiné studii o volně dostupném rozhraní pro online výpočty [2]. V této studii autor mimo jiné dospěl k závěru, že od většiny uživatelů nelze očekávat, že si prostudují jakkoliv krátké instrukce nebo návod k použití programu.

3 Praktická ukázka

Jako praktickou ukázku si uveďme program na řešení kvadratické rovnice pomocí diskriminantu. PHP skript [5] převezme z URL adresy kvadratický výraz nebo kvadratickou rovnici

[4] Například lomítko pro celočíselné dělení.

a zobrazí postupně převod rovnice na základní tvar $ax^2+bx+c=0$, výpočet diskriminantu a kořenů rovnice a převod levé strany rovnice na čtverec. Vstup je zpracován programem `formconv`, což uživateli umožní například vynechávat znak `*` pro násobení. Veškeré výpočty jsou provedeny programem Maxima, jehož výstup je zobrazen v prohlížeči. Pro snazší čitelnost jsou v tomto výstupu všechny matematické výrazy zapsány v TEXu a při zobrazení jsou automaticky nahrazeny obrázky generovanými programem mathTEX. Skript je možno vyzkoušet na adrese `http://wood.mendelu.cz/math/zilina/kvadr.php`. Rovnici je nutno předat jako parametr, například `.../kvadr.php?x^2+5x=15`.

Potřebné matematické výpočty jsou prováděny programem Maxima při dávkovém zpracování souboru `kvadr.mac` [5]. Protože je program Maxima „příliš aktivní“ při zpracování výrazu na vstupu, je v některých místech nutno nastavit proměnnou `simp` na hodnotu `false`. Toto nastavení zajistí například, že výraz `2*4*5` nebude automaticky upraven na číslo `40`. Skript není příliš sofistikovaný, protože pomocí diskriminantu řeší například i rovnice jako $x^2=3$ nebo $x(x-1)=0$, což by v praxi použil jenom naprostý matematický zoufalec. Je proto nutné ho chápat pouze jako ukázku. Tyto speciální případy kvadratické rovnice se ostatně dají ošetřit samostatně a případné pokusy tímto směrem jsou ponechány čtenáři jako cvičení.

4 Závěr

V příspěvku byly představeny myšlenky, na nichž je vybudována online aplikace Mathematical Assistant on Web. Tato aplikace automaticky provádí výpočty derivací, určitých a dvojných integrálů, Lagrangeova a Taylorova polynomu, lineárního modelu metodou nejmenších čtverců, definičních oborů funkcí jedné a dvou proměnných, umí řešit základní diferenciální rovnice prvního a druhého řádu a je silnou pomůckou při výpočtu neurčitých integrálů a vyšetřování průběhu funkce. Všechna řešení jsou prezentována včetně mezivýpočtů. Ve většině případů je možno s trochou nadsázky říct, že řešení jsou prezentována tak, jak by je pravděpodobně napsal šikovný student.

Protože přes svou jednoduchost má MAW příznivý ohlas mezi vysokoškolskými studenty, je v závěrečné části uveden praktický příklad sestavení podobné aplikace, která v jednotlivých krocích zobrazuje postup řešení kvadratické rovnice a doplňování výrazu na čtverec.

Aplikace MAW byla sestavena částečně jako pomůcka pro kombinované studium (pro kontrolu výsledků), částečně ze zvědavosti jak se současná výpočetní technika vypořádá s úlohami počítanými na technických školách a univerzitách v základních kurzech matematiky, částečně jako pomůcka pro matematiky kteří věnují svůj čas odpovídání matematických dotazů na internetových fórech, částečně jako nástroj pro pohodlné vymýšlení příkladů do zkouškových písemek, částečně i jako občasný únik od vědecké práce a zpestření života vysokoškolského učitele. Vždy však byla sestavována s myšlenkou, že role živého učitele ve výuce matematiky je zcela nezastupitelná.

Lineární diferenciální rovnice druhého řádu

http://user.mendelu.cz/marik/maw

Řešíme počáteční úlohu $y'' + 2\,y' + y = 0; \qquad y(0) = 1, \quad y'(0) = -1$

Nejprve nalezneme obecné řešení rovnice a potom použijeme počáteční podmínky pro nalezení řešení partikulárního.

Řešíme homogenní rovnici

$$y'' + 2y' + y = 0$$

Charakteristická rovnice je $\lambda^2 + 2\lambda + 1 = 0$

Kořeny charakteristické rovnice: $\lambda_{1,2} = \dfrac{-\,(2) \pm \sqrt{(2)^2 - 4\,(1)}}{2} = \dfrac{-2 \pm \sqrt{0}}{2} = -1$

Charakteristická rovnice má jeden dvojnásobný kořen.
Dvě lineárně nezávislá řešení jsou $y_1 = e^{-x}$ a $y_2 = xe^{-x}$.
Obecné řešení je $y = C_1 e^{-x} + C_2 x e^{-x}$.

Nyní řešíme počáteční úlohu $y'' + 2\,y' + y = 0; \qquad y(0) = 1, \quad y'(0) = -1$

Obecné řešení je $y(x) = (C_2\,x + C_1)\;e^{-x}$
Derivace obecného řešení je $y'(x) = C_2\,e^{-x} - (C_2\,x + C_1)\;e^{-x}$

Dosazením počátečních podmínek dostáváme soustavu lineárních rovnic

$$\begin{cases} C_1 = 1 \\ C_2 - C_1 = -1 \end{cases}$$

Řešení soustavy lineárních rovnic

$$[C_1 = 1, C_2 = 0]$$

Dosazením vypočtených hodnot do obecného řešení získáme partikulární řešení
$y(x) = e^{-x}$

Obr. 1: Ukázka řešení počáteční úlohy pro lineární homogenní LDR s konstatnními koeficienty

Poděkování

Tato práce vznikla s podporou FRVŠ v rámci řešení projektu 99/2008.

Literatúra

[1] BAKOS, G. – KOVACS, Z.: formconv, http://particio.com/english/LightNEasy.php?page=formconv

[2] FATEMAN, R.: Analysis of a web user interface for mathematics: experiences with integral queries for TILU (Table of Integrals Look Up), http://www.eecs.berkeley.edu/~fateman/papers/tjames.pdf

[3] GNUplot, http://www.gnuplot.info/

[4] MAŘÍK, R. – TIHLAŘÍKOVÁ, M.: Mathematical Assistant on Web. http://user.mendelu.cz/marik/maw/

[5] MAŘÍK, R.: http://wood.mendelu.cz/math/zilina/kvadr.zip

[6] Maxima, http://maxima.sourceforge.net/

[7] Maxima PHP, http://maximaphp.sourceforge.net/, http://www.my-tool.com/mathematics/maximaphp/

Kontaktní adresa

Robert MAŘÍK (doc., Mgr., PhD.),
Ústav matematiky MZLU v Brně, Zemědělská 3,
613 00 Brno,
marik@mendelu.cz

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

MULTIPLATFORMOVÝ VÝVOJ INTERAKTÍVNEJ APLIKÁCIE – HRY PYTHON A PYGAME

MORAVČÍK, Milan, (SK)

1 Úvod

Vývoj interaktívnych aplikácií je jedným z najťažších, najtvorivejších a najkomplexnejšcích programátorských problémov. Veľmi atraktívnou aplikáciou tohto druhu sú počítačové hry. Tešia sa veľkej obľube mladších i starších hráčov, no málokto sa zamyslí, čo všetko sa za vytvorením hry skrýva a ešte menej používateľov sa pokúsi jednoduchú hru vytvoriť.

Tvorba počítačovej hry začína vymyslením scenára, pokračuje tvorbou grafických prvkov, animácií, ozvučenia a končí programovaním, ktoré poskladá jednotlivé „prísady“ do jedného celku. Teda vývoj hry je komplexná tvorivá práca, ktorá prinajmenšom vyžaduje programátorské, režisérske, grafické a estetické zručnosti. Nie je výnimkou, skôr pravidlom, že za vývojom počítačovej hry je tím niekoľkých ľudí. *Pohľad na tvorbu hry bude v našom príspevku veľmi jednoduchý, laický – chceme zaujať začínajúceho programátora, nie profesionálny vývojársky tím.*

Vývoj počítačovej hry je úžasnou motiváciou pre začínajúceho programátora. Rieši pritom rôzne programátorské problémy. Prirodzene sa dostáva k otázkam použitia údajových štruktúr (organizovanie scény, udržiavanie množín objektov), časovačom (práca s animáciami), udalostiam (interaktívnosť s užívateľom) a pod. Zoznamuje sa s vývojovým prostredím, možnosťami programovacieho jazyka a knižníc, získava základné zručnosti pre prácu s grafikou, zvukmi a organizáciou svojej vlastnej práce.

Prečo je vývoj počítačovej hry atraktívny aj pre začínajúceho programátora:

1. výsledkom je nový jedinečný produkt, s ktorým sa môže pochváliť a prezentovať tak svoje programátorské zručnosti (ale aj iné),

2. tvorí program ktorý je interaktívny, zábavný, môže byť aj poučný (edukačný softvér),
3. celý proces vývoja je komplexný, tvorivý a nekladie medze fantázii autora.

V súčasnej dobe má programátor k dispozícii množstvo vývojových technológií. Každá technológia, v užšom kontexte jazyk, prostredie, knižnice a pod., veľakrát prináša odlišné spôsoby riešenia rovnakého problému. Ostáva na pleciach skúsenejších programátorov, pre ktorú alternatívu sa rozhodnú. V našom príspevku sa zameriame na technológie a postupy, ktoré sú zaujímavejšie pre jednotlivcov – začínajúcich, resp. mierne pokročilých programátorov. Na konkrétnom príklade predstavíme programovací jazyk Python a knižnicu Pygame, ako jednu z možných alternatív vývoja jednoduchej interaktívnej aplikácie – ktorou môže byť hra, ale aj edukačný softvér pre mladších používateľov.

2 Python

Programovací jazyk Python je obľúbeným a pomerne rozšíreným jazykom o ktorom možno nájsť množstvo kníh a elektronických materiálov [5], preto len v krátkosti pripomenieme, že jazyk Python vyvinul v roku 1990 Holanďan Guido van Rossum. *Ide o moderný, dynamický, interpretovaný programovací jazyk, ktorý umožňuje objektovo orientované, štruktúrované aj funkcionálne programovanie*. Patrí medzi jazyky z najjednoduchšou a najčitateľnejšou syntaxou, ktorá je taktiež rýchla na zápis zdrojového kódu programu. Zároveň je stručný na vyjadrovanie sa, čo znamená, že rovnaký program napísaný v inom jazyku presiahne dvoj až trojnásobný počet riadkov, ako program napísaný v Pythone.

Ukážka krátkeho programu, ktorý spočíta výskyt jednotlivých slov v textovom súbore `slova.txt` (výsledok program zapíše do údajovej štruktúry – Slovník):

```
f = open('slova.txt', 'r')
slovnik = {}
for riadok in f.readlines():
    for slovo in string.split(riadok):
        if slovnik.has_key(slovo):
            slovnik[slovo] = slovnik[slovo] + 1
        else:
            slovnik[slovo] = 1
```

Mnohí menej skúsení programátori považujú Python za zastaralý jazyk, opak je však pravdou. Python patrí medzí moderné skriptovacie programovacie jazyky, poskytuje plnoautomatické riadenie pamäte, operácie na vyššej úrovni abstrakcie, jednoduchú prácu s údajovými štruktúrami (zoznamy, slovníky, *n*-tice a pod.). Počas vývoja tak programátor nemusí riešiť technické detaily, sústreďuje sa viac na riešenie koncepčných problémov tvorby aplikácie.

Platformová nezávislosť [1], licenčná politika[2] a jednoduchosť syntaxe jazyka, motivovali viaceré univerzity, kde sa Python stal jazykom na úvodných kurzoch programovania [6–8].

Použitie jazyka v rôznych oblastiach umožňujú knižnice (v kontexte s jazykom Python sa nazývajú moduly), ktorých je v prípade jazyka Python obrovské množstvo. Softvéroví vývojári sa s nimi bežne stretávajú pri:

1. vytváraní užívateľského rozhrania (PyQT, PyGTK, wxPython, dynWin, Tix),
2. spravovaní údajov a databáz (PyXpath, MySql-python, PySQLite, PyGreSQL, psycopg),
3. práci s grafikou (PIL, PyChart, gdModule),
4. vytváraní hier (Pygame, Pyglet, PySoy, PyOpenGL) a pod.

3 Pygame

Pygame je multiplatformový modul jazyka Python. Je navrhnutý pre jednoduchý a rýchly vývoj multimediálneho interaktívneho softvéru. Pygame rozširuje funkčnosť jazyka Python využívajúc knižnicu SDL. Knižnica SDL je sčasti napísaná v jazyku C pre nízkoúrovňový prístup k audiu, externým ovládacím perifériám a k hardvéru. Je tiež multiplatformová a svojím výkonom a možnosťami použitia je porovnateľná napr. s DirectX. Pygame sa skladá z niekoľkých častí, pričom každá z nich pomáha riešiť často sa objavujúce programátorské problémy, ako napr.

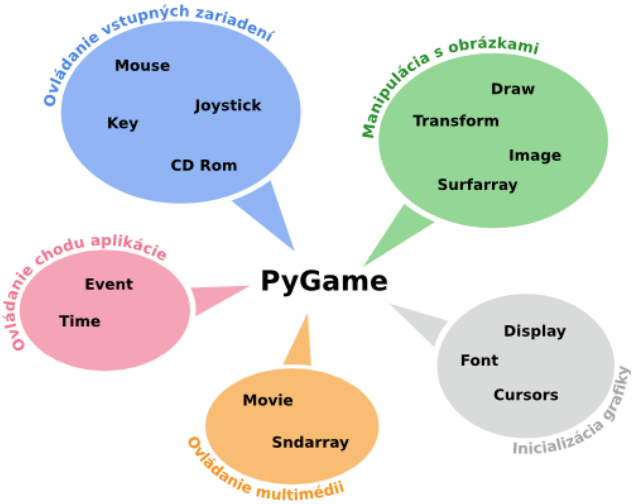
1. ovládanie vstupných zariadení (myš, klávesnica, joystick, . . .),
2. práca s udalosťami, časovačom,
3. manipulácia s grafikou hry (inicializácia obrazovky, práca s textom, . . .),
4. práca s rôznymi obrázkovými formátmi (načítanie, zobrazenie, uloženie, upravovanie aj na nízkej prístupovej úrovni),
5. prehrávanie multimediálnych formátov (audio, video).

4 Vytvorenie jednoduchej neinteraktívnej hry

Na začiatok ukážeme, ako možno naprogramovať jednoduchú hru v jazyku Python a v module Pygame. Hlavnou postavou je Ufón, ktorý sa po obrazovke pohybuje podľa stlačených

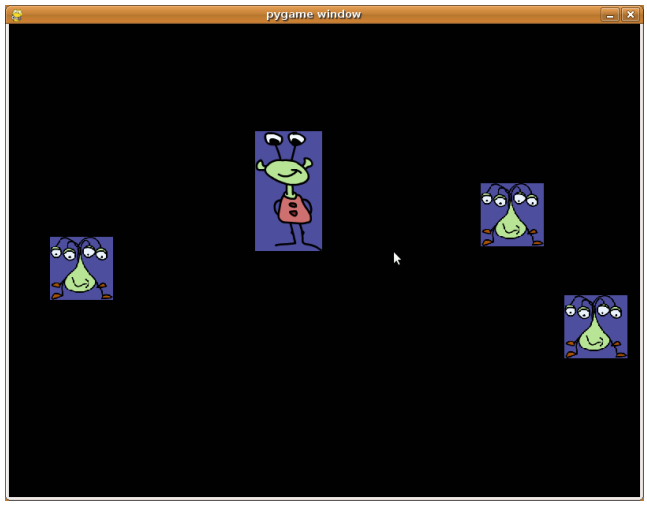
[1] Interpreter Pythonu je implementovaný pre viaceré platformy, netreba však zabúdať že mnohé z často používaných modulov existujú len v prípade OS GNU/Linux, BSD, Mac OS a MS Windows.

[2] Licencia je na pár odlišností totožná s open source licenciou, je voľne distribuovateľný, použiteľný aj pre komerčné učely. O presné vymedzenie licencie sa stará organizácia Python Software Foundation (PSF) [online cit. 29. 5. 2009] `http://python.org/psf`.



Obr. 1: Moduly knižnice Pygame. Pohľad na moduly rozdelené do piatich skupín podľa ich významu v použití pri tvorbe interaktívnej aplikácie: ovládanie vstupných zariadení, ovládanie chodu aplikácie, ovládanie multimédií, inicializácia grafiky a manipulácia s obrázkami

klávesov hore, dole, vpravo, vľavo. Na scéne sa vyskytne aj niekoľko príšer. Každá z príšer sa pohybuje náhodným smerom a náhodnou konštantnou rýchlosťou.



Obr. 2: Ukážka obrazovky prvej verzie hry

Zatiaľ si ukážeme naprogramovanie len takejto jednoduchej funkcionality, neskôr budeme hru zdokonaľovať, čím bude pre používateľa zaujímavejšia, ale aj programátorsky náročnejšia. Vytvorenie hry rozdelíme na štyri časti:

1. inicializácia hry,
2. vytvorenie grafických prvkov,
3. vykreslenie grafických prvkov,
4. rozhýbanie grafických prvkov.

Inicializácia hry spočíva vo vložení potrebných modulov a inicializácii obrazovky. Keďže budeme používať modul Pygame, musíme ho na začiatku vložiť (import pygame). Vložili sme aj ďalšie dva moduly, ich opodstatnenie si vysvetlíme neskôr.

```
import sys, pygame, random
pygame.init()
size = 800, 600
# premenná pomocou ktorej budeme pristupovať ku grafickej ploche hry
screen = pygame.display.set_mode(size)
```

Vytvorenie grafických prvkov s využitím modulu Pygame je veľmi jednoduché. Najprv vytvoríme grafický prvok hry *Ufón*.

Ufón	`ufon = pygame.image.load("ufon.gif")` `ufonrect = ufon.get_rect()`

Pre každý grafický objekt si budeme pamätať tzv. *Surface*, ktorý predstavuje „obrázkovú premennú“. Takáto premenná sa bude vykresľovať na obrazovku (v našom prípade je ňou premenná *ufon*). Taktiež si budeme pamätať rozmery a umiestnenie obrázkovej premennej. Využijeme na to premennú *ufonrect*.

Podobným spôsobom vytvoríme ďalší grafický prvok *Príšerka*. Navyše si budeme pamätať vektor jej pohybu. Pri jeho generovaní sme využili funkciu *randint* z modulu *random* (modul *random* sme vložili na začiatku pri inicializácii hry).

Príšerka	`priserka1 = pygame.image.load("priserka.gif")`
	`priserkarect1 = priserka.get_rect()`
	`a = random.randint(-10, 10)`
	`b = random.randint(-10, 10)`
	`vektor1 = (a, b)`

Vykreslenie grafických prvkov do grafickej plochy. Pripomíname že grafická plocha, ktorej veľkosť je 800×600, je uložená v premennej *screen*. Pomocou metódy *blit* vykreslíme ľubovoľný grafický objekt do plochy *screen*. Po vykreslení všetkých grafických objektov musíme zavolať metódu *pygame.display.flip()*, ktorá prekreslí obrazovku našou grafickou plochou.

```
screen.blit(ufon, ufonrect)
screen.blit(priserka1, priserkarect1)
pygame.display.flip()
```

Rozhýbanie grafických prvkov zatiaľ v nekonečnom cykle. Aby sa zmeny nevykonávali príliš rýchlo musíme vykonávanie nášho programu spomaliť. V takomto prípade sa v hrách využíva „zamrznutie“ scény hry na krátku dobu. Takúto situáciu môžeme vytvoriť pomocou „časovača“, ktorý pozastaví program na niekoľko milisekúnd/sekúnd.

```
clock = pygame.time.Clock()
while True:
    screen.fill(0)
    screen.blit(ufon, ufonrect)
    screen.blit(priserka1, priserkarect1)
    pygame.display.flip()
    # časovač ''tikne'' 25 krát za sekudnu = pozastaví program na
40 miliseknúnd
    clock.tick(25)
    priserkarect1.top = priserkarect1.top + vektor1[0]
    priserkarect1.left = priserkarect1.left + vektor1[1]
```

5 Vylepšenie hry, zavedenie interaktívnych prvkov

Doposiaľ naprogramovaná hra je neinteraktívna a má veľa nedostatkov. Napr. príšerka sa neustále pohybuje aj za okrajmi obrazovky, kde ju nie je vidieť. Prvok hry ufón je zatiaľ bez pohybu – budeme ho ovládať klávesovými šípkami a zároveň zabezpečíme aby sa nedostal mimo grafickej plochy. Taktiež náš program bude reagovať na kolízie grafických prvkov – ak má príšerka s ufónom prienik, zmizne z obrazovky. Ak zmiznú všetky príšerky hra sa ukončí výpisom: Vyhral si! Vylepšenie hry rozdelíme do piatich celkov:

1. vytvorenie tried pre grafické objekty (ufón, príšerka),
2. rozhýbanie ufóna,
3. riešenie kolízie ufóna a príšerky,
4. ukončenie hry,
5. hlavné telo programu.

Hru vylepšíme aj z pohľadu programátora. Z dôvodu lepšej organizácie grafických prvkov hry *vytvoríme dve triedy*, jednu z nich pre prácu s *ufónom* a druhú pre prácu s *príšerkou*. Pri vytvorení objektu (inštancie) triedy *Ufon* sa volá konštruktor, ktorý je v Pythone deklarovaný ako *def __Init__()*. Kedže Python je interpretovaný jazyk, pred každú premennú, ktorú chceme používať v rámci celej triedy píšeme slovíčko *self* (to isté slovíčko je prvým argumentom každej metódy).

```
class Ufon:
    def __init__(self):
        self.obr = pygame.image.load("ufon.gif")
        self.rect = self.obr.get_rect()
```

Ufóna rozhýbeme pomocou metódy *pohni()*. Jej úlohou je zistiť stav stlačeného klávesu a následný pohyb ufóna príslušným smerom. Ak bol napr. stlačený kláves hore (*if pygame.key.get_pressed()[pygame.K_UP]:*) zmenšíme pozíciu ľavého horného rohu obdĺžnika ufóna o 6 bodov. Zároveň testujeme (*if rect.top < 0:*), či nová pozícia nie je mimo okrajov obrazovky.

```
class Ufon:
    def __init__(self):
        ...
    def pohni(self):
        rect = self.rect
        if pygame.key.get_pressed()[pygame.K_UP]:
            rect.top = rect.top  - 6
            if rect.top < 0:
                rect.top = 0
        elif pygame.key.get_pressed()[pygame.K_DOWN]:
            rect.top = rect.top  + 6
            if rect.bottom > size[1]:
                rect. bottom = size[1]
        elif pygame.key.get_pressed()[pygame.K_LEFT]:
            rect.left = rect.left  - 6
            if rect.left < 0:
                rect.left = 0
        elif pygame.key.get_pressed()[pygame.K_RIGHT]:
            rect.left = rect.left  + 6
            if rect.right > size[0]:
                rect.right = size[0]
```

Grafický prvok *Príšerka* uložíme do triedy *Priserka*. Konštruktor je podobný ako konštruktor *Ufona*, no naviac obsahuje presun objektu do približného stredu obrazovky a ďalšiu premennú – vektor pohybu o náhodnej veľkosti.

```
class Priserka:
    def __init__(self):
        self.obr = pygame.image.load("priserka.gif")
        self.rect = self.obr.get_rect()
        self.rect.topleft = 400, 300
        a = random.randint(-10, 10)
        b = random.randint(-10, 10)
        self.vektor = [a, b]
```

Príšerku rozhýbeme metódou *pohni()*, presunieme ju o veľkosť vektora. Ak *príšerka* prekročí hranice grafickej plochy zmení sa jej vektor podľa pravidla: *uhol odrazu = uhlu dopadu*. V zásade sa vektor pohybu rovnako zmení keď sa objekt dostane za hornú a dolnú hranicu (horizontálny odraz), alebo taktiež za pravú a ľavú hranicu plochy (vertikálny odraz).

```
class Priserka:
    def __init__(self):
        ...
    def pohni(self):
        rect = self.rect
        rect.left = rect.left + self.vektor[0]
        rect.top = rect.top + self.vektor[1]
        # vertikálny odraz
        if (rect.left < 0) or (rect.right > size[0]):
            self.vektor[0] = -1 * self.vektor[0]
        # horizontálny odraz
        if (rect.top < 0) or (rect.bottom > size[1]):
            self.vektor[1] = -1 * self.vektor[1]
```

Do triedy *Ufon* doprogramujeme metódu *kolizia()*, ktorá testuje prienik obrázku *ufona* s obrázkom *príšerky*. V prípade ich prieniku je *príšerka* odstránená zo zoznamu príšer.

```
class Ufon:
    def __init__(self):
        ...
    def pohni(self):
        ...
    def kolizia(self, priserka):
        rect = self.rect
        if rect.colliderect(priserka.rect):
            priserky.remove(priserka)
```

Ak v scéne nie sú žiadne príšerky (*if not priserky:*), hru ukončíme výpisom *Vyhral si*!

```
if not priserky:
    font = pygame.font.Font(None, 36)
    text = font.render(Vyhral si!, 1, (255, 255, 255))
    textpos = pygame.Rect(400, 300, 0, 0)
    screen.blit(text, textpos)
```

Hlavný telo programu našej aplikácie môže vyzerať napr. takto:

1. vytvoríme inštanciu triedy *Ufon*,
2. inštancie triedy *Priserka* ukladáme priamo do premennej *priserky* (premenná je typu *zoznam*, čo je jedna zo vstavaných údajových štruktúr jazyka Python),
3. vytvoríme objekt *clock* pre ovládanie časovača hry,
4. v hlavnom cykle (*while True:*) aplikácie:

   (a) testujeme systémové ukončenie aplikácie,
   (b) prekresľujeme plochu (čiernou farbou),

(c) zabezpečujeme pohyb ufóna a príšeriek,
(d) testujeme kolízie a koniec hry.

```
# vytvorenie objektu ufon = inštancia triedy Ufon
ufon = Ufon()
# inicializácia zoznamu priserky
priserky = []
# vytvorenie a pridanie 3 príšeriek do zoznamu
priserky.append(Priserka())
priserky.append(Priserka())
priserky.append(Priserka())
# vytvorenie objektu časovača
clock = pygame.time.Clock()
# hlavný cyklus programu
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    # prekreslenie obrazovky čiernou farbou
    screen.fill(0)
    screen.blit(ufon.obr, ufon.rect)
    for item in priserky:
        screen.blit(item.obr, item.rect)
    # ''zamrznutie'' programu na 40 miliseknúnd
    clock.tick(25)
    pygame.display.flip()
    # pohyb ufóna
    ufon.pohni()
    # pohyb príšeriek s riešením kolízie = prienik s ufónom
    for item in priserky:
        item.pohni()
        ufon.kolizia(item)
    # testovanie konca hry
    if not priserky:
        font = pygame.font.Font(None, 36)
        text = font.render(Vyhral si!, 1, (255, 255, 255))
        textpos = pygame.Rect(400, 300, 0, 0)
        screen.blit(text, textpos)
```

6 Ďalšie tipy, triky a drobné vylepšenia

Konceptuálne vylepšenia

1. **Iný spôsob reagovania na udalosti** – každých 40 milisekúnd zisťujeme stav stlačenia kláves, ak užívateľ stlačil a pustil kláves tesne pred testovaním, nestihneme na túto

zmenu zareagovať. V praxi sa často tento problém rieši pomocou zásobníka udalostí, napr.:

```
for event in pygame.event.get():
   if event.type == pygame.KEYDOWN:
       print event.key
```

1. **Čo sme pre jednoduchosť zanedbali**

   (a) *zápis cesty k súborom* je odlišný pri unixových a windowsových systémov. Na odstránenie nejednotnosti môžeme použiť príkaz `os.path.join(adresar, subor)`

   (b) *ošetrenie vstupných súborov* – pre zvýšenie bezpečnosti programu by sme mali testovať existenciu a neporušenosť vstupných súborov. Programátori tento problém riešia tzv. *výnimkami*. Pre bezpečné načítanie obrázkov možno použiť konštrukciu:

   ```
   try:
      obr = pygame.image.load("obr")
   except pygame.error, message:
      print 'Nemôžem načítať obrázok:', "obr"
      raise SystemExit, message
   ```

2. **Komentáre a dokumentácia:**

   ```
   # existujú len jednoriadkové komentáre
   """dokumentácia je tiež súčasťou jazyka, môžeme ju vyvolať
   pomocou príkazu: help(nazov_triedy)"""
   ```

Drobné tipy, triky

1. **Vykresľovanie bitmapy do pozadia** – rovnakým spôsobom ako vykresľovanie objektov:

   - ```
     pozadie = pygame.image.load("obr")
     screen.blit(pozadie) # namiesto screen.fill(0)
     ```

1. **Nastavenie grafickej plochy na celú obrazovku**, tzv. *fullscreen*, môžeme urobiť aj počas behu programu príkazom `pygame.display.set_mode(size, pygame.FULLSCREEN)`.
2. **Prekeslenie časti obrazovky** – doteraz sme pomocou príkazu `pygame.display.flip()` prekresľovalicelú obrazovku. Niekedy sa nám hodí prekresliť len malú časť obrazovky. Príkazom `pygame.display.update(rect_list)`, kde *rect_list* je zoznam jedného alebo viacerých obdĺžnikov určíme časti obrazovky ktoré sa prekreslia.
     ```

3. **Skryť kurzor myši** – `pygame.mouse.set_visible(False)`, ak ho potrebujeme opätovne zobraziť `...set_visible(True)`. Systémový kurzor môžeme skryť, ak ho chceme zameniť za obárzok. Obrázok potom presúvame pri udalosti zmeny polohy myši.
4. **Priesvitné pozadie obrázkov** – príkazom `obr=pygame.image.load("obr")` vytvoríme objekt typu *pygame.Surface*, ktorého metóda *set_colorkey()* umožňuje nastavenie (aj percentuálnej) priesvitnosti. V našom prípade nastavenie priesvitnosti pre objekt *ufon*:

```
obr.set_colorkey(obr.get_at((0,0)), pygame.RLEACCEL)
```

7 Záver

Programovanie interaktívnych programov, predovšetkým hier, patrí medzi prvé projekty začínajúcich programátorov. V článku sme predstavili jednu z možných alternatív, prezentovali sme vývoj pomocou skriptovacieho jazyka Python a modulu Pygame. Zvolený jazyk a modul má svoje silné a slabé stránky. Nás zaujal pohodlný a rýchly vývoj, a to vďaka jednoduchej syntaxi moderného jazyka vysokej úrovne a jednoduchému spracovaniu grafiky aplikácie pomocou rozširujúceho modulu.

Modul Pygame sa najčastejšie používa pri tvorbe jednoduchých edukačných hier, ako sú napr. *Gcompris* alebo *Schoolsplay*. Jeho použitie v iných typoch programov je zriedkavé, väčšinou sa kombinuje s použitím ďalších modulov.

Plusy	**Mínusy**
(1) Jednoduchá konfigurácia a bezproblémová inicializácia na viacerých platformách,	(1) Nerieši otázku GUI, treba použiť ďalší modul resp. knižnicu ako je napr. GTK,
(2) licencia LGPL,	(2) chýba podpora viacerých audio a video formátov,
(3) interakcia s rôznymi vstupnými zariadeniami (nie len myš, klávesnica ale aj pákové zariadenie tzv. joystick alebo CD–ROM mechaniky),	(3) chýba nahrávanie zvuku z mikrofónu (tento nedostatok rieši napr. modul *PyAudio*, resp. *PyMedia*),
(4) široká podpora grafických obrázkových formátov (`jpg`, `png`, `gif`, `bmp`, `pcx`, `tif`, ...),	(4) v niektorých prípadoch chýba vizuálne vývojové prostredie,
(5) zobrazovanie a generovanie `TrueType` fontov,	(5) samotný beh programu je pomalší ako pri kompilovaných jazykoch.
(6) predprogramovaná trieda Sprite (organizácia práce s grafickými objektami aplikácie ako napr. vykresľovanie do vrstiev, riešenie kolízií medzi skupinami objektov, ...).	

Príspevok vznikol v rámci grantu číslo UK/370/2009 Univerzity Komenského v Bratislave.

Literatúra

[1] McGUGAN, W.: *Beginning Game Development with Python and Pygame (From Novice to Professional)*, Berkeley: 2007, 316 s., ISBN: 978-1-59059-872-6

[2] MORAVČÍK, M.: *Edukačný softvér pre deti predškolského veku* (písomná práca k dizertačnej skúške), Bratislava: 2009, FMFI UK v Bratislave

[3] HARMS, D., McDONALD, K.: *Začíname programovat v jazyce Python.* Brno: Computer press, 2003, 456 s., ISBN 978-80-251-2161-0

[4] `http://www.root.cz/clanky/letajici-cirkus-20` [online cit. 29. 5. 2009]

[5] Python [online cit. 29. 5. 2009] `http://www.python.org/`

[6] `http://mcsp.wartburg.edu/zelle/python/python-first.html` [online cit. 20. 5. 2009]

[7] KAUKIČ, M.: *Python ako prvý programovací jazyk na VŠ*, 7th International conference, APLIMAT 2008, Slovak University of Technology in Bratislava

[8] `http://www.ece.uci.edu/~chou/py02/python.html` [online cit. 29. 5. 2009]

[9] Pygame – An Open Source Community Project, [online cit. 23. 5. 2009] `http://www.pygame.org/`

[10] Simple Directmedia Layer (SDL) [online cit. 23. 5. 2009] `http://www.libsdl.org/`

[11] SchoolsPlay [online cit. 28. 5. 2009] `http://www.schoolsplay.org/`

[12] GComrpis – edukačný softvér pre deti od 2 do 12 rokov [online cit. 28. 5. 2009] `http://gcompris.net/`

Kontaktní adresa

Milan MORAVČÍK,
Katedra základov a vyučovania informatiky,
FMFI UK v Bratislave,
Mlysnská dolina 1, 842 48 Bratislava,
moravcik@fmph.uniba.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

SOFTWARE WAS JUST THE BEGINNING

2046 (CZ)

1 Virtuální problematika

1.1 Uzavřenost, otevřenost

Začneme na příkladu dnes používaných systémů. Unixové systémy jsou dosud nejkvalitnějším obecně známým systémem, jenž se díky uzavřenosti a akademické hře na vlastním písečku globálně neprosadil. Jednoho dne se objevil pan Bill Gates, který vstoupil do dění s poměrně agresivní taktikou a prosadil „jeho“ operační systém na globálním trhu. Tím otevřel cestu potencionálním kutilům, kterým se, když už nic jiného, dostalo zařízení s operačním systémem alespoň do ruky a mohli tak přemýšlet o potencionálním využití.

Díky jisté míře otevřenosti se přirozeně objevili vynalézaví lidé, kteří se snažili využít příležitosti, stavěli počítače a psali další a další programy. Tato komunita, přestože se nepovažuje za komunitu napomohla Windowsu k jeho úspěšnosti, nikoli Windows jako takový, ale jeho částečná modifikovatelnost, otevřenost.

Problém je, že se nikomu nelíbí „sloužit“ konkrétní osobě, kterou navíc zná jménem. A protože otevřenost systémů Windows nikdy nebyla zdaleka dostačující pro plné využití skrytého potenciálu, zrodila se potřeba po plně otevřené systémové architektuře.

V roce 1991 začal Linus Torvalds pracovat na otevřeném systému později nazvaným Linux. Ať už to byla krádež kódu Unixu nebo ne, nebylo dost síly na to tento projekt zastavit. Linux se totiž ukázal být zpětně výhodný jak pro akademickou tak komerční komunitu používající a vyvíjející Unix.

Výsadou Linuxu byla otevřenost jenž z akademické komunity vývojářů vytvořila komunitu vývojářů celosvětovou. Linux byl od začátku potencionálně schopný reagovat na jakékoli potřeby. Jeho otevřenost měla ale za následek nekoncentrovaný vývoj. Vzniklo

a stále vzniká mnoho distribuci duplikující již funkční části jejich protějšků, čímž se jeho vývoj „brzdí“.

1.2 Open Source

Vstřícnost a otevřenost jsou vlastnosti, kterých si na Open Source vážíme nejvíce. Open Source je právě takový, je otevřený se vstřícnou komunitou. Lidé z Open Source komunity rádi pomohou a co víc, v mnoha případech jsou ochotni věnovat svůj čas a najít řešení pro problém, jimž by se jinak vůbec nezaobírali. Všem je jasné, že není v našich silách znát všechno a ještě navíc tomu rozumět, tak i profesionálové naslouchají začátečníkům a navzájem budují otevřený a vlídný svět sdílené informace.

Funguje zde jakýsi komunizmus kde „vše“ je vlastnictvím všech a společně budují to po čem touží. Po čem touží dělník Open Source? Mít možnost vytvářet své sny a dále je šířit ve víře většího fungujícího celku, jenž mu zpětně vrací svobodu výběru, násobících se možností jeho vlastní cesty.

1.3 Problémy a zneužití

Otevřenost Open Source je zároveň jeho Achillovou patou. Otevřenost není ani tak problémem pro filosofii Open Source jako pro jeho komunitu. Díky nutnosti regulérní návaznosti na stávající jurisdikci a marketing se Open Source stává ideálním nástrojem pro výdělek třetí strany, jenž aniž by porušovala jakékoli pravidla, těží z Open Source ve svůj prospěch, bez potřeby vracet cokoli zpět. Komunita jedinců věřící v ideu svobody s potenciálem realizace pro všechny bez rozdílu ráda zapomíná na stále přítomnou hrozbu zneužití vychytralým jedincem či skupinou. Bohužel s anarchií přichází i možnost zrodu vládnoucího, jádra které se etabluje díky své finanční základně a uvědomění si nepřehlednosti situace a krátkozrakosti participujících jedinců.

Stejně tak, jak všechny druhy náboženství nabízeli a stále s úspěchem nabízejí pomocnou ruku těm kdo ji „potřebují“ v podobě nemocnic, minimálního či lepšího sociálního zabezpečení, starostlivosti o dobro svých budoucích oveček, tak pracují i třetí strany se stádem věřících v Open Source. Pomocná ruka stabilního subjektu není ztrátová, či pouhou dobročinností, jak se na první pohled může zdát, nýbrž promyšleným tahem za účelem většího zisku a moci.

1.4 Open Source, vývoj zadarmo

Google je firma, která vznikla přesně podle amerického ideálu z příležitosti a geniality pár jedinců, která se v následujících letech stala světovým fenoménem a je dnes jednou z nejsilnějších komerčních subjektů dneška. Nejenže jeho jméno má příznačnou podobnost se slovem bůh, anglicky God, ale i se slovem „good“ (dobře, výborně, ..) a dalším významem vyjadřující postoj naši doby „cool“ (super, perfektní, souhlas,..), ale navíc používá i stejné strategie cesty náboženství „dobrého otce“. Lide pracující na poli medií a masmédií by

mohli vyprávět o podprahovém vnímání významů slov a tvarů pro konečného uživatele, konzumenta. . .

Google nabízí bezplatnou možnost hledat a nacházet informace, esenciální to prvek naší doby. Nabízí nástroje jak nalézt, učit se a dokonce opět bezplatně poskytuje své další služby. Google především neustále rozšiřuje své pole působnosti. Například kupuje firmu Keyhole Inc. vyvíjející produkt Earth Viewer a přetváří ho na vlastní produkt Google Earth. Google Earth je pokus o implementaci dlouho očekávané virtuální reality k běžnému použití a mimo jiné pokus o možný posun v technologii vyhledávačů a samozřejmě rozšíření marketingového pole. Google Earth je částečně otevřen pro běžné uživatele, kteří si mohou vymodelovat, čí otexturovat (nanést fotky na 3D model) vlastní dům a tak napomoci Googlu v jeho rozvoji, aniž by musel přispívající jednotlivce platit. Google nabízí uživatelům Gmail (emailovou schránku), vytváří Google calendar (kalendář) umožnující synchronizaci Gmailem. Google přichází s projektem Google code, vytvořeném pro vývojáře pracující v souvislosti s produkty Google.

Projekt One Laptop Per Child (jeden laptop pro každé dítě) hledá vhodný systém pro svůj hardware. Podíváme-li se na projekt OLPC zblízka je zřejmý jeho komerční dopad na budoucnost pro dodavatele systému, který se tímto způsobem dostane do rukou milionu klientů (dětí). Přestože se o OLPC pokoušeli všechny majoritní firmy vyvíjející operační systémy, byl pro OLPC vybrán Red Heat (Linux) s nadstavbou grafického rozhraní Sugar. Když se ale podíváme na rozhraní se kterým se děti dostanou do styku najdeme skvěle propracovaný Google systém. Představte si malé dítě které si omylem smaže obrázek své maminky, to přece nemůžeme dopustit. A tak by mělo být všechno ve výsledku synchronizované s Googlem a veškerá data by měla ležet na serverech Google. Takže, to s čím přijdou uživatele, v tomto případě miliony dětí, do styku, je Google, nikoli systém Red Heat (Linux).

Google se snaží dostat do hry v nejsilnější oblasti, v oblasti mobilních zařízení a přišel na trh s Androidem. Android je opět postaven na Open Source, přesněji Linuxovém jádru s nadstavbou Android. Tento projekt na rozdíl od samotného Linuxu má jedno velké plus. Android je konkrétně rozvíjen na základě politicko-ekonomických (ekonomicko - politických) záměrů Google.

Proč se Google rozhodl pro Open Source? Ze stejného důvodu proč se IBM rozhodlo podporovat Apache, PHP, a MySQL. V porovnání s klasickým vývojem uvnitř firmy, téměř nulové náklady na vývoj a testování aplikací.

1.5 Centralizace

Dnešní technologie je čím dál víc nerozdělitelně spojená s Internetem. Počínaje prezentací firem, obchodovaní, poštovních sdělení, osobních stránek a jiných důležitých či nedůležitých osobních dat každého z nás, až po první pokusy o online systémy, které by měli nahradit naše osobní počítače a lokální aplikace. Lokální instalace aplikací dosud vyhovovala a to díky nemožnosti přenášet informace dostatečnou rychlostí. Vývoj tedy začal od separovaných jedinců s jejich výpočetní stanicí jenž sloužila sama sobě. Takovýto systém dovoloval do

určité míry svobodu a ochranu osobních dat. Z toho maximálně vytěžila firma Microsoft v tu dobu ovládající trh s osobními počítači. Problémem byla nezávislost a naprostá nekontrolovatelnost obsahu těchto počítačů, a to především programů, ve valné většině nelegálních a obsahu dat uživatele, které se ovšem v té době ani zdaleka nepovažovali za důležité, poněvadž pro ně nebylo komerčního využití. Čím více se firmy snažily získat přehled nad legalitou používaných programů tím více napomáhali k pádu svého jádra Microsoftu. Se sílou přenosu dat se vrátila jedna z původních myšlenek centralizace výpočetní sítě z dob počátků výpočetní technologie. V tu chvíli ovšem přichází na trh rozrůstající se Google s čerstvými myšlenkami a vědom si aktuálních potřeb uživatelů a především vizí budoucích, uživateli dnes netušených potřeb zítřka, začíná využívat Open Source.

Zdá se, že se stane to, o čem většina Linuxářů sní a to že Linux se stane majoritním systémem a vytlačí Windows. Vše bude otevřené, vše bude modifikovatelné a nebudeme sloužit Billovi . . . ale Google.

2 Reálné dopady

2.1 Osobní, neosobní

Zneužitelnost obsahu uvolněného pod jednou z „volných“ licenci (GNU GPL, CC, . . .) je pouze jedním článkem řetězce zneužívání naší sleposti. Open Source nastolil do dnešní doby nevídanou otevřenost informací. A to především osobních informací týkajících se konkrétních jedinců. Slepá honba za otevřeností všeho nás pomalu zbavuje jednoho ze základních kamenů svobody. Práva na soukromí.

Otevřenost v sobě zahrnuje nebezpečný prvek, se kterým se naše společnost do této chvíle nikdy nesetkala a neví si s ním přirozeně rady. Wikipedie transformuje informace v multi-veřejné data přístupné komukoli odkudkoli. Myspace prezentuje jednotlivce veřejnosti a FaceBook odkrývá osobní život mimo rámec našich „důvěryhodných“ přátel.

Publikace a předávání osobních informací ve velkém měřítku zvyšuje křehkost narušení osobního prostoru na maximum. Už dnes o nás Google ví téměř vše co se týče našich zálib na internetu a pokud máme už uživatelský účet na Googlu, tak je monitorována historie prohlížených stránek, stejně jako naše schůzky (Google calendar), náš bankovní účet (Google checkout) a naši přátelé (Gmail, Youtube) apod. (vsuvka: Nemluvě o záznamech našich telefonních rozhovorů a SMS v našem „reálném“ světě.)

To na co všichni poukazují v případě Hackerů, přesněji Crackerů dělá Google legálně. Shromažďuje osobní informace a využívá je pro své převážně komerční účely. Problém je, že Google nelze odsoudit za to, že nám „čte“ emaily, či prohlíží jiné soukromé informace. Stejně jako nelze odsoudit textový editor, že zobrazí text, či najde vybrané slovo. Nebo snad ano? Problém není jen to, že jedna firma bude ve výsledku schraňovat, vlastnit všechna naše osobní data, ale při cíleném, nebo „náhodném“ poškození či zneužití tohoto centrálního mozku lidstva. Dobrým příkladem takovéhoto problému, kdy centrální napojení závažně zasáhlo do našeho života bylo přerušení dodávky plynu z Ukrajiny. Pokud by existovalo

necentralizované připojení nikdy bychom se do takovýchto problémů nedostali.

Touto oklikou se dostáváme k zajímavému srovnání funkčnosti otevřeného (decentralizovaného) a uzavřeného (centralizovaného) systému. K jejich výhodám a nevýhodám. Síťová, decentralizovaná struktura se zdá být funkčnější z hlediska dostupnosti, ale nebezpečnější z hlediska narušení osobních dat. I když je otázkou, jestli je bezpečnější uchovávat peníze pod jedním polštářem, nebo rozprostřené po celém bytě.

2.2 Otevřený tok uzavřených dat

Protože vidím centralizaci jako hrozbu dosavadním úspěchům zmíním zde hrubou vizi řešení problému osobních dat, tak aby nedocházelo k zneužívání informací a k jejich následné centralizaci a zneužití jedním subjektem. Možnou variantou webu (internet, intranet) je decentralizované neveřejné uchovávání dat a jejich přenos. A to jak uchovávání tak přenos dat. Je třeba se zamyslet co je veřejné a co je osobní.

Ukazuje se, že je na doteď veřejném poli jako je internet stále víc potřeba intimity! Ideální by bylo, pokud by byl Internet v budoucnu pouze přenosovým kanálem bez možnosti záznamu citlivých dat jakoukoli třetí stranou (člověkem, programem). Web by měl mít možnost prostoru, přenosu bez historie, kde nikdo jiný jindy než v danou chvíli je nemůže číst. Je třeba vytvořit jakýsi uzavřený kanál jen pro úzký specifikovaný okruh korespondentů.

Pokud bychom se chtěli maximálně zbavit centralizace, možným řešením by mohla být naprostá decentralizace uzavřených dat, kde by nemohlo dojít k nabourání stability v důsledku odříznutí zdroje, při zachování maximální bezpečnosti a soukromí přenášených dat. Řešení decentralizace by mohlo být následující. Jednotlivé počítače uživatelů by tvořili neurony jednoho mozku, kterými by putovali kopie rozprostřených informací. Data by se tak nikdy neztratili, protože by „vždy“ existovala někde jejich kopie na některých z dalších neuronů sítě. Tím by se částečně mohl řešit i problém s ekologickou náročností serverů, protože by se využíval výkon strojů, které už tak jako tak běží. Příkladem řešení zde může být např. Skype. Řešením by mohl být systém založený na idee Onion Cat (tor). Onion Cat je jakási uzavřená síť uvnitř sítě, kde jsou v ideálním případě potřebné data předávány mezi jednotlivci aniž by někdo mohl blíže specifikovat zdroj.

Jak zabezpečit data globálně rozprostřené v prostoru je otázkou.

3 Reálná problematika

3.1 Virtualizace

Poprvé v historii lidstva máme možnost testovat složité struktury sociálního chování, před aplikací v reálném životě. A tak ideály anarchie jsou najednou aplikované komerčními subjekty. V anarchii objevují silný finanční a vývojový potenciál. Tato virtuální anarchie objevuje a postupně hledá a nachází řešení pro problémy dříve ve fyzickém světě neřešitelné.

Ba co víc, virtuálně aplikované řešení se transformuje do té míry, že se začleňuje do právního systému ovlivňující svět reálný.

Nedávno byla například do českého právního systému zařazenená licence Creative Commons. Tato licence vznikla primárně pro potřeby „virtuálního prostředí“ načež se stala součástí našeho „reálného“ světa. Technologie nám umožnila testovat jinak reálně neaplikovatelné vize a následně je zařadit do „reálného“ světa. Například anarchistické vize systému nebylo v naši společnosti dodnes téměř možné realizovat. Dnes, díky „virtuální“ realitě tady máme Open Source filozofii. Systém uvědomění si výjimečnosti a užitečnosti jedince pro celek, a to i v případě kdy narušuje stabilitu „funkčně“ zaběhnutého prostředí. Obecné sdílené prostředí je neustále modifikováno a narušováno jedinci participujícími na celku. A to vše při zachování jejich identity s možností kdykoli systém opustit a zvolit si jinou alternativu aniž by tím ohrozili svoji společenskou existenci.

3.2 Software was just the beginning

Protože v následující části se budu věnovat tématu, ve kterém panuje největší dezorientace a mnozí se tudíž k těmto závěrům a hlavně k termínu Anarchie budou apriori odvracet zády, použiji pro upřesnění pár citací.

Sen o svobodě jedince kde je „společnost kulturním socializmem bez státu, bez centralizmu a hierarchie, kde se individualita a společenskost vzájemně propojují“ [3, str. 368] se začíná reálně zhmotňovat až s příchodem GNU licence, která nám jako první dala možnost otevřeně sdílet naše výsledky a zapojit je do funkčního celku. Konečně tu byla možnost aplikovat základní požadavky Anarchie. A to projevovat se v rámci společnosti autonomně.

Anarchie je mylně spojována s agresivními útoky některých anarchistických skupin, které dnes díky virtuální přípravě hnutí Open Source není zapotřebí. Samotná historie anarchizmu je oproti obecnému mínění plná mírumilovných vizí. Jako například v Landauerovým myšlenkách kdy „kladl podmínku, aby se revoluce prosazovala pokojným způsobem, tedy aby se duch revoluce stal součástí mírového a humánního společenského pořádku“ [3, str. 368]. Je zajímavé jak moc se ideály „Open Source“ ztotožňují s ideály anarchie. Například postřehy typu „Žádný vývoj v životě národů se neodehrává jinak nežli na základě individuálního úsilí.“ přesně vystihuje praxi s jakou se Open Source denně rozvíjí [3, str. 170].

Družnost a pospolitost i potřeba vzájemné podpory jsou tak vlastní lidské povaze, že nikde v dějinách nenalézáme lidi, kteří by žili jen v osamocených malých rodinách a zápasili jedni s druhým o prostředky k životu. Novodobé bádání naopak ukazuje, že od prvopočátku předhistorického života se lidi sdružovali v rody, klany, či kmeny, které propojila idea společného původu i uctívání společných předků. Po celá tisíciletí udržovali tyto organizace lid pospolu, ač tu vůbec nebylo nějaké autority, která by je k tomu nutila [3, str. 218].

Proces spolupráce je naprosto totožný s Wardovými předpoklady: *Ward argumentoval, že člověk není svou přirozeností líný, ani se nestaví proti práci, naopak se realizuje pracovními aktivitami, i když je unavený po pracovní době. Lidé jsou ovšem proti takové práci, ve které musí soustavně poslouchat někoho jiného a pracovat na něčem co z jejich pohledu*

třeba nemá smysl. Proto také třeba mnoho lidí samostatně podniká, i když je to někdy vede k osobním rizikům a osobnímu vyčerpání [3, str. 538]. Tyto úvahy jsou o to důležitější dnes, kdy se opět schyluje k centralizaci (Google). Hrozí nám nebezpečí vzdoru, což by způsobilo návrat do doby před GNU a veškerý výdobytek otevřených myšlenek by propadl vniveč.

Naše společnost se pomalu stává schopnou fungovat na otevřených pravidlech což se ukazuje i na testech povolením pravidel rychlosti jízdy na silnicích. Dochází ke všeobecnému přirozenému zvýšení ostražitosti a obezřetnosti. Předpoklad jediného funkčního systému plného pravidel a restrikcí se ukazuje mylným. Naopak se ukazuje, že chaotická struktůra vykazuje potřebnou dávku funkčního uspořádání.

Domnívám se že tato schopnost reagovat v rámci neuspořádanosti je do jisté míry podpořena trendem sdílení na internetu. Chaotickému uspořádání informací, které jsme se naučili vyhledávat a číst. To vše opět díky otevřeným chaotickým technologiím. Na základě vývoje a dopadu hnutí „Open Source“ se dá říci, že obvyklá námitka, že anarchistický systém by mohl existovat pouze někde mezi přírodními národy, a ne v pokročilých moderních sociálních celcích, se ukazuje jako scestná. Tato idea je naopak realizovatelná až s vyspělou civilizací již technologie umožnila aplikovat otevřené přístupy myšlení a otevřené struktůry. Open source hnutí se tedy dá přirovnat k politickému směru, kterým se podle mě ve výsledku stane.

4 Závěr

Domnívám se, že Open source je předzvěstí nového sociálního uspořádání, založeného na kapitalizmu s možností otevřené kolaborace a následného zisku při zachování autonomity jedince a udržení funkčního celku.

Literatúra

[1] `http://Google.com`

[2] `http://en.wikipedia.org`

[3] TOMÁNEK, V. – SLAČÁNEK, O.: *Anarchizmus svoboda proti moci*. 2006, ISBN 80-70021-781-2

[4] TAPSCOTT, Don – WILLIAMS, A. D.: *Wikinomiks*. 2008, ISBN 978-1-84354-637

[5] ZITRAN, J.: *The future of the Internet*. 2008, ISBN 978-1-8461-4014-3

[6] MARTIN, J.: *The Meaning of the* 21th *century*. 2006, ISBN 9781903919866

[7] TZU, Lao: *Tao Te Ching*. `http://www.taopage.org/resources.html`

Kontaktná adresa
2046, 2046@o—-o.info, `http://o----o.info/`

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

OpenStreetMap – SLOBODNÁ WIKI MAPA SVETA

PÁLENÍK, Michal, (SK)

1 Úvod

V celosvetovom ako i Slovenskom priestore existuje niekoľko mapových systémov. Väčšina z nich má nedostatky, ktoré sťažujú alebo znemožňujú ich použitie v akademickom prostredí:

1. dáta aj softvér majú proprietárnu licenciu,
2. je problematické zanášať zmeny do dát,
3. rozsah typov možných objektov v dátach je veľmi úzky,
4. je problematické vytvárať prispôsobené vizualizácie a exporty.

Ako odpoveď na tieto nedostatky vznikol v roku 2004 projekt OpenStreetMap, ktorý si kladie za cieľ vytvoriť slobodnú wiki mapu sveta.

2 Podobné projekty v minulosti

Podobný problém bol v minulosti s encyklopédiami. Zabehnuté encyklopédie sa neprispôsobili trendu internetizácie a podcenili vedomosti bežných používateľov Internetu. Ako alternatíva vznikla Wikipédia, ktorá posunula hranice a možnosti encyklopédií. Známou je malá možnosť kontrolných mechanizmov, nerovnomerné pokrytie kľúčových slov a vojny okolo diskutabilných kľúčových slov (napr. interupcie alebo vojna v Iraku). Aj napriek týmto nedostatkom má Wikipédia veľký objem. Slovenská verzia má po zozipovaní 80 MB (bez revízií, diskusií a obrázkov), anglická verzia ťažšie zvládnuteľných 4,9 GB.

Projekt OpenStreetMap je podľa oficiálnej stránky www.OpenStreetMap.org „The Free Wiki World Map“ [1], čiže slobodná wiki mapa sveta.

Mapa je slobodná v zmysle licencie Creative commons Attribution-Share Alike 2.0 Generic. Licencia umožňuje použitie dát (kopírovať a upravovať dáta) pri dodržaní dvoch podmienok: označiť zdroj dát a licencovať upravené alebo kopírované dáta pod rovnakou alebo podobnou licenciou [2]. Rovnaká licencia je používaná aj na projekt wikipédia.

Mapa je wiki, v zmysle, že každý (kto koná v zmysle licencií a súhlasí so zverejnením svojej práce) môže upravovať mapu. Licenčné ujednania sú uvádzané pri registrácií nového prispievateľa.

Mapa je mapa sveta, keďže obsahuje dáta z celého sveta. Pokrytie údajmi nie je rovnomerné (ako dôsledok wiki filozofie). Tradične je najlepšie pokrytie v Nemecku a Anglicku.

3 História OpenStreetMap a Freemap

História projektu OpenStreetMap začala v roku 2004, keď bola zaregistrovaná doména `http://www.OpenStreetMap.org/` [1]. Na vianoce roku 2005 bol zaregistrovaný tisíci používateľ (aktívny používateľ, teda prispievateľ). V roku 2006 bol vydaný editor Josm, podarilo sa naimportovať dáta do prístroja Garmin, a bola dostupná online mapa s vizualizáciou Mapink. V roku 2007 pribudol používateľ číslo 10 000, bol vydaný editor Potlatch, v databáze už bolo 5 miliónov ciest. V roku 2008 bolo už 25 000 užívateľov a 20 miliónov ciest. V roku 2009 bola prekročená hranica 100 000 používateľov.

V priebehu času sa postupne menilo ukladanie dát v databáze (API 0.6 bolo dané do používania v apríli 2009). Taktiež sa rozširoval počet možných objektov v mape. V súčasnosti je napríklad 38 druhov obchodov [4].

Projekt OpenStreetMap zastrešuje OpenStreetMap Foundation so sídlom v Spojenom kráľovstve, ktorá vznikla v roku 2006 [5]. Organizácia zabezpečuje prevádzku serverov, organizáciu konferencií a rozširovanie mapy do rozvojových regiónov.

Celosvetový projekt používa 15 serverov. Tieto sú pomenované podľa drakov nejak spájaných s frázou „tu žijú draci“ [9]. Niekoľko ďalších organizácií prevádzkuje mirrory databázy, takže aj v prípade fyzického zničenia hlavných serverov fungovanie projektu nebude ohrozené.

OpenStreetMap sa na Slovensku začal rozvíjať v roku 2006. Portál `http://www.freemap.sk/` bol založený v roku 2007. Komunita okolo projektu prevádzkuje webový portál, vytvára exporty pre rôzne zariadenia, propaguje mapu v regiónoch a získava dáta pre hromadné importy. WWW stránka [14] ukazuje vývoj dĺžok jednotlivých ciest na území SR.

4 Štruktúra dát

Každý objekt v dátach má niekoľko atribútov (tags). Tieto atribúty sú z veľkej časti štandardizované [4]. Proces štandardizácie je zdĺhavý a musí vyhovieť rôznorodým požiadavkám z rôznych častí sveta. Napríklad v Rumunsku sú turistické chodníky značené nielen prúžkami, ale aj trojuholníkmi, krížikmi a krúžkami, existuje niekoľko rôznych druhov národných parkov a v časti sveta sa jazdí vľavo.

4.1 Body

Bod (node, POI) je jednotlivý bod na mape. Môže stáť buďto sám (napríklad telefónna búdka) alebo byť časťou cesty (napríklad semafor na ceste). Niektoré body môžu stáť aj samostatne aj ako časť cesty (napríklad hotel môže byť samostatne alebo v tej časti budovy, kde je vchod do hotela).

4.2 Cesty

Cesta (way) v mape spája niekoľko bodov. Cesta môže byť napríklad diaľnica, hranica okresu, lanovka alebo chodník. Špeciálnou cestou je uzavretá cesta, oblasť. Príkladom je budova, jazero alebo hranica lesa [6].

4.3 Relácie

Relácia (relation) spája niekoľko ciest do jedného celku. Príkladom sú linky MHD alebo čísla ciest. Keďže relácie boli doplnené iba nedávno (október 2007), ich používanie nie je ešte úplne vžité [7].

5 Zdroje dát

5.1 Verejne dostupné dáta

Najrýchlejším spôsobom doplnenia mapy je hromadný import. Toto však skrýva niekoľko problémov. Prvým sú licenčné problémy, importované dáta musia byť poskytnuté pod licenciou CC-BY-SA [2]. Druhým problémom je dôveryhodnosť a presnosť importovaných dát. Tretím problémom je prekrývanie sa s existujúcimi dátami (nie je vhodné mať tú istú cestu dvakrát). Štvrtým problémom je automatické doplnenie dát. Taktiež importované dáta zväčša nemajú všetky potrebné tagy a treba ich upraviť a doplniť. Ako vidno, aj pri hromadnom importe je veľa manuálnej roboty, ale verejne dostupné dáta dokážu veľmi uľahčiť rýchle dopĺňanie mapových dát.

V USA je niekoľko dostupných datasetov použitých v projekte OpenStreetMap. Najväčším je dataset Tiger [10]. Obsahuje ulice a cesty. Import tohto datasetu trval viac ako rok.

Na Slovensku je niekoľko datasetov, ktoré sú naimportované do databázy. Najväčšie sú údaje o obciach zo Štatistického úradu SR, vrstva Wikipédie a údaje zo Štátnej ochrany prírody [11]. Ostatné zdroje sú v rokovaní.

5.2 Individuálny zber

Oveľa dôležitejší ako hromadný import je individuálny zber informácií. Najjednoduchšie prispievanie do mapy je dopĺňanie bodov do už existujúcej mapy. Používateľ vie, kde sa nachádza reštaurácia (napr na rohu ulíc), nájde toto miesto v niektorom editore mapy (napr Potlatch alebo JOSM), zakreslí bod, priradí mu tagy amenity=restaurant a názov. K tomuto nepotrebuje ani GPS prístroj ani veľa času. Podobne sa dá prispievať napríklad pomenovávaním ulíc.

Iná možnosť bez použitia GPS prístrojov je obkresľovanie ciest z ortofotomáp Yahoo. Yahoo poskytlo svoje fotky na použitie v projekte OpenStreetMap [12]. Naneštastie, Slovensko je kvalitne pokryté iba na západe republiky. Používanie iných ortofoto podkladov (napr Google alebo Zoznam) nie je z licenčných dôvodov možné.

Ak prispievateľ vlastní GPS prístroj, môže vytvárať mapu aj na „miestach, kde žijú draci“. Stačí zaznamenať trasu po rôznych uliciach, vizualizovať ich na (prázdnej) mape, obkresliť cesty a zaznačiť ich kvalitu a názvy [13].

Licenčné ujednania komerčných poskytovateľov máp (napr. Google maps, Zoznam mapy, katastrálne mapy, ...) neumožňujú ich používanie pri vytváraní mapy OpenStreetMap. Taktiež, tieto mapy sú často nepresné (napríklad [8]) a obsahujú zámerné chyby, podľa ktorých sa dá ľahko identifikovať kópia. Preto je najlepšie vytvárať mapy podľa reality v oblasti, ktorú užívateľ pozná.

6 Nadstavby nad dáta

6.1 Webové rozhrania

V rámci definície je popísané iba akými tagmi sú definované jednotlivé objekty [4]. Spôsob ich vizualizácie sa môže líšiť.

Základné rozhranie je dostupné na www stránke [1]. Táto zobrazuje štandardné objekty, napr. `http://www.OpenCycleMap.org/` je prispôsobené pre cyklistov (obsahuje vrstevnice), `http://www.informationfreeway.org/` obsahuje inak zobrazované cesty, a podobne.

Na `http://www.FreeMap.sk/` [6] je vizualizácia prispôsobená zvyklostiam slovenských používateľov. Okrem dát z OpenStreetMap obsahuje vrstevnice, turistické a cyklistické chodníky, hrady, objekty z wikipédie, a podobne.

6.2 Exporty do zariadení

Licencia dátových podkladov umožňuje vytváranie offline kópie. Toto otvára možnosti pre použitie v rôznych špecializovaných zariadeniach. Existuje niekoľko aplikácií do mobilných telefónov. Napríklad rastrový TrekBuddy alebo vektorový GpsMid. Existuje aj export do Garminu.

Okrem elektronických verzií je niekoľko nástrojov na generovanie tlačených máp. Najjednoduchšia verzia, určená najmä pre zakresľovanie do mapy, je dostupná na `http://walking-papers.org/`.

6.3 Vyhľadávanie ciest

Dátové podklady sú vo forme orientovaného grafu, takže vytvorenie vyhľadávania cesty je relatívne jednoduché. Najznámejšia online aplikácia je na `http://www.yournavigation.org/` iná je na `http://www.freemap.sk/`. Ďalšie sú súčasťou špecializovaných programov (napríklad GpsMid).

6.4 Iné

Dostupnosť dát umožňuje vytvorenie mnohých iných aplikácií. Prvým je zobrazenie mapy z OpenStreetMap na vlastnej stránke, napríklad pomocou služby `embedded.freemap.sk`. Jedná sa o zobrazenie pozície bodu alebo cesty.

Ďalšou možnosťou je zobrazenie cesty (vizualizácia tracklogu). Služba umožňuje prehľadné zobrazenie cesty, napríklad prejdenej túry alebo zobrazenie cesty k miestu konania konferencie.

7 Použitie v školách

Dátové podklady v projekte OpenStreetMap sú pod slobodnou licenciou a väčšina nástrojov pre prácu s dátami je pod licenciou GNU/GPL. Toto umožňuje jednoduché využitie v školskom a akademickom prostredí.

Najjednoduchšie spôsoby použitia sú pasívne. Na portáli `http://www.freemap.sk/` sú vrstvy wikipédia a vrstva hradov. Wikipédia obsahuje všetky údaje zo slovenskej wikipédie, ktoré majú koordináty. Žiakom sa dá jednoducho ukázať, kde sú ktoré významné mestá a iné objekty. K dispozícií je tiež vrstva reliéfu, ktorá názorne ukazuje významné pohoria a rieky.

Ďalšie možnosti sú aktívne, žiaci môžu editovať dátové podklady. V prvom rade získajú počítačové zručnosti a zistia vnútorné usporiadanie mapových podkladov. Veľkou výhodou je, že editujú reálne dáta, ktoré sa zobrazia na mape v priebehu niekoľkých dní. Taktiež pri kreslení mapy spoznajú svoje okolie, nielen objekty, ktoré bežne poznajú, ale aj objekty zaujímavé pre iné cieľové skupiny.

Iné možnosti využitia projektu OpenStreetMap je upravovanie vizualizácie a kreslenie ikoniek. Na mape je veľa ikoniek reprezentujúcich rôzne objekty, veľa ikoniek chýba (napríklad ohnisko alebo fontána). Podobne, vykresľovanie niektorých objektov (napr tunel alebo vlek) nie je riešené najšťastnejšie. Skúsenejší žiaci alebo študenti môžu tieto vizualizácie upravovať a nakoniec aj uviesť do života.

Dátové podklady OpenStreetMap sú vlastne sieť alebo orientovaný graf. Toto umožňuje implementáciu a testovanie rôznych algoritmov na hľadanie ciest alebo optimálnych tokov. Podobne, dopĺňanie ďalších foriem vizualizácie dát je možné a relatívne ľahké (napríklad špeciálna vizualizácia pre cyklistov, rôzne špecializované Java aplikácie a podobne).

8 Záver

V príspevku sme stručne predstavili projekt OpenStreetMap, ako naozaj voľne dostupnú a slobodnú wiki mapu sveta. Načrtli sme dôvody jeho vzniku, históriu, základy fungovania a použitie v praxi. Predstavili sme mapu ako sieť, ktorá je vizualizovaná niekoľkými spôsobmi. V článku sme tiež v krátkosti ukázali možné nadstavby nad základnú mapu a poukázali sme na jej možnosti využitia vo vyučovacom procese. V neposlednom rade, sme v článku predstavili projekt Freemap Slovakia.

Literatúra

[1] http://www.openstreetmap.org/

[2] http://creativecommons.org/licenses/by-sa/2.0/

[3] http://wiki.openstreetmap.org/wiki/History

[4] http://wiki.openstreetmap.org/wiki/Map_Features

[5] http://www.osmfoundation.org/

[6] http://www.freemap.sk/

[7] DURSA, E.: Ako sa robí mapa, http://hysteria.sk/xmas2008/ako_sa_robi_mapa-ventYl.pdf

[8] PÁLENÍK, M.: Freemap, http://linuxfest.sk/prednasky/linuxfest05/Mapy_a_Linux_freemap.pdf

[9] http://wiki.openstreetmap.org/wiki/Server

[10] http://wiki.openstreetmap.org/wiki/Tiger

[11] http://wiki.freemap.sk/AkviziciaDat

[12] http://wiki.openstreetmap.org/wiki/Yahoo

[13] http://wiki.openstreetmap.org/wiki/Sk:Beginners_Guide

[14] http://wiki.freemap.sk/FreemapStats

Kontaktná adresa

Michal PÁLENÍK,
združenie Freemap Slovakia, Matičná 8/A5,
900 28 Ivanka pri Dunaji,
michal.palenik@freemap.sk
http://www.FreeMap.sk/

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

NAJLACNEJŠIE ÚPLNÉ PÁRENIE VO VŠEOBECNÝCH GRAFOCH S POUŽITÍM OSS NÁSTROJA GLPK

PEŠKO, Štefan, (SK)

1 Úvod

V teórii grafov [3] pod *úplným párením* v grafe $G = (V,H)$ rozumieme takú podmnožinu P množiny hrán H, že každý vrchol z množiny vrcholov V inciduje práve s jednou hranou z P. Budeme sa zaoberať problémom najlacnejšieho úplného párenia v grafe (MWPM – Minimum Weight Perfect Matching), ktorý možno formulovať takto:

V danom hranovo ohodnotenom grafe $G = (V,H,c)$, kde každá hrana $h \in H$ má cenu c_h (reálne číslo) treba nájsť úplné párenie P s minimálnou sumárnou cenou $c(P) = \sum_{h \in P} c_h$.

Okrem úloh MWPM s *bipartitnými grafmi* (napr. modelujúcich priradenie pracovníkov k strojom), ktoré možno ľahko formulovať ako úlohy LP (lineárneho programovania) o najlacnejšom prípustnom toku v sieti, nie je nám vo *všeobecných grafoch* známa takáto transformácia. Medzi základné výsledky kombinatorickej optimalizácie patrí Edmondsovo riešenie problému MWPM časovo polynomiálnym *kvetovým* algoritmom, ktorý je mimoriadne náročný na implementáciu [1]. S potrebou riešiť takéto úlohy sme sa stretli v dopravnej logistike [4] pri heuristickom riešení problémov tvorby rozvrhov vozidiel a osádok alebo kapacitných okružných dopravných úloh, ale aj pri tvorbe študijných skupín v univerzitnom rozvrhu.

V tomto príspevku sa chceme podeliť nielen o alternatívne riešenie problém MWPM stredných rozmerov pomocou OSS knižníc pre zmiešané celočíselné lineárne programovanie (MILP Mixed Integer Linear Programming) – konkrétne nástrojov GLPK [2] s možnosťou využitia pohodlného modelovacieho jazyka GMPL (GNU Mathematical Programming Language), ale aj o cestu, ktorá nás priviedla k úvahám nad celočíselným jadrom MILP.

2 Edmondsov lineárny model

Edmondsov algoritmus z roku 1965 je založený na LP formulácii úlohy MWPM. Nech $G = (V,H,c)$ je hranovo ohodnotený graf s párnym počtom vrcholov a nech $\mathscr{C}$ označuje množinu všetkých aspoň trojprvkových podmnožín množiny vrcholov V s nepárnym počtom prvkov. Pre všetky podmožiny $S \subseteq V$ budeme značiť $\delta(S)$ množinu hrán, ktoré majú práve jeden vrchol v S. Pre vektor $(x_h : h \in H)$ a množinu $E \subseteq H$ nech $x(E) = \sum_{e \in E} x_e$. Teraz už môžeme formulovať Edmonsovu primárnu úlohu lineárneho programovania [1] (ELP):

$$\sum_{h \in H} c_h x_h \ \rightarrow \min, \tag{1}$$

$$x(\delta(\{v\})) = 1 \quad \forall v \in V, \tag{2}$$

$$x(\delta(S)) \ \geqq 1 \quad \forall S \in \mathscr{C}, \tag{3}$$

$$x_h \geqq 0 \qquad \forall h \in H. \tag{4}$$

Cieľová funkcia (1) minimalizuje cenu párenia. Incidenčná podmienka (2) zabezpečuje aby každý vrchol incidoval s práve jednou hranou grafu. Podmienka (3) zaručuje bivalentnosť premenných x_h od ktorých sa požaduje len obligátorná podmienka nezápornosti (4).

Významným úspechom Edmondsovej formulácie je, že sa autorovi podarilo ostrániť nutnosť požiadavky bivalentnosti premenných x_h. Jej slabým miestom je, že početnosť množiny $\mathscr{C}$ rastie exponenciálne s početnosťou množiny vrcholov V. Tento nedostatok je ale vyriešený pomocou duálnej úlohy k úlohe ELP a podmienok komplementarity týchto duálne združených úloh. Od publikovania Edmondsovho algoritmu, ktorý má zložitosť $O(|V|^2 \cdot |H|)$ uvádza Cook a kol. [1] do roku 1999 až 19 rôznych počítačových implementácií, ktoré sa líšia použitými dátovými štruktúrami a zložitosťou algoritmov. V citovanej práci boli úspešne vyriešené inštancie až s 5 000 000 vrcholmi pričom 100 000 - vrcholové geometrické inštancie boli vyriešené do troch minút na počítači 200 MHz Pentium–Pro.

Náš cieľ je oveľa skromnejší, čo do rozmeru riešených inštancií úloh i doby výpočtu. Začneme s formuláciou úlohy, ktorá je vhodná pre riešiče MILP.

3 MILP formulácie problému

Najjednoduchšou bivalentnou formuláciou úlohy MWPM, ktorá má zhodný počet premenných aj ich interpretáciu ako v úlohe ELP, je nasledujúca úloha (BLP):

$$\sum_{h \in H} c_h x_h \ \rightarrow \min, \tag{5}$$

$$x(\delta(\{v\})) = 1 \quad \forall v \in V, \tag{6}$$

$$x_h \in \{0,1\} \qquad \forall h \in H. \tag{7}$$

V tejto úlohe sme len nahradili podmienky (3) a (4) úlohy ELP podmienkou (7). Výhodou tejto formulácie je len jej jednoduchosť. Jej nevýhoda, neprijateľne dlhá doba výpočtu, sa prejaví už pri riešení úloh stredných rozmerov $|V| \approx 60$.

Pozorovali sme, že jednou z príčin dlhého výpočtu je veľký počet celočíselných premenných, ktorý spôsobuje vetvenie hlboko v strome riešení aj v prípade, keď je optimálne riešenie nájdené, ale ešte nie je overené. Preto je potrebné nájsť takú MILP formuláciu úlohy, v ktorej bude čo najmenej celočíselných premenných.

Prvý model, ktorý spĺňal naše požiadavky, vychádzal z formulácie klasického priraďovacieho problému v úplnom bipartitnom grafe K_{nn} s párnym počtom n vrcholov v oboch častiach $S = \{1,2,\ldots,n\}$ a $T = \{1,2,\ldots,n\}$ grafu, ktorého dvojice (i,j) sú ohodnotené symetrickou maticou $C = (c_{ij})$ pre $(i,j) \in S \times T$, kde $c_{ii} = \infty$. Nech (x_{ij}) je hľadaná bivalentná matica neznámych, ktorá priradí hrane (i,j) hodnotu 1 ak je $\{i,j\}$ hranou hľadaného párenia P v grafe G. Dostávame tak nasledujúcu úlohu zmiešaného celočíselného lineárneho programovania (AMLP):

$$\sum_{i\in S}\sum_{j\in T} c_{ij}x_{ij} \quad \to \min, \tag{8}$$

$$\sum_{j\in S} x_{ij} \quad = 1 \qquad \forall i \in S, \tag{9}$$

$$\sum_{i\in T} x_{ij} \quad = 1 \qquad \forall j \in T, \tag{10}$$

$$x_{ij} \geqq 0 \qquad \forall (i,j) \in S \times T, \tag{11}$$

$$x_{ij} - x_{ji} \quad = 0 \qquad \forall (i,j) \in S \times T : i < j, \tag{12}$$

$$y_i - \sum_{j\in T: j>i} j x_{ij} = 0 \qquad \forall i \in S - \{n\}, \tag{13}$$

$$y_i \geqq 0, \text{ celé} \qquad \forall i \in S - \{n\}. \tag{14}$$

Klasický priraďovací problém je tu formulovaný LP úlohou (8)–(11). Symetrizačná podmienka (12) zabezpečí, že hrana párenia $\{i,j\}$ je reprezentovaná dvojicou $(i,j),(j,i) \in S \times T$, pre ktoré je $x_{ij} = x_{ji}$. Dôkaz, že jednotkové hodnoty páriacich premených $x_{ij} = x_{ji} = 1$ vynucujú podmienky (13) a (14) bude podrobne vyložený v pripravovanom článku Peško [5]. Do tých čias sa uspokojíme s experimentálnym overením tohoto poznatku.

Model AMLP sme formulovali v úplnom bipartitnom grafe K_{nn}, aby sa ukázalo, či počítačové experimenty na náhodných inštanciách úplného grafu G nenájdu neceločíselné optimálne riešenie. Ukázalo sa, že celočíselné premenné y tvoria *celočíselné jadro* [5] modelu AMLP, t. j. *z celoločíselnosti premenných jadra vyplýva celočíselnosť všetkých bázických riešení uvažovanej úlohy MILP*.

Počítačové experimenty ukázali podstatné zrýchlenie výpočtu v modeli AMLP oproti modelu BLP. Naviac pre reálnu inštanciu so 100 vrcholmi, na ktorej riešič *glpsolve* modelu BLP nedopočítal riešenie ani po cca 16 hodinách výpočtu, bolo nájdené riešenie v modeli

AMLP po cca 50 minútach. To nás priviedlo k finálnej modifikácii modelu BLP, v ktorom sa naviac požaduje celočíselnosť len dodatočného vektoru $n-1$ premených $(y_i : i \in V - \{n\})$ čo vedie k úlohe (YLP):

$$\sum_{\{i,j\}\in H} c_{\{i,j\}} x_{\{i,j\}} \quad \rightarrow \min, \tag{15}$$

$$\sum_{\{i,j\}\in \delta(\{i\})} x_{\{i,j\}} \quad = 1 \quad \forall i \in V, \tag{16}$$

$$x_{\{i,j\}} \geqq 0 \quad \forall \{i,j\} \in H. \tag{17}$$

$$\sum_{\{i,j\}\in \delta(\{i\}):j>i} j \cdot x_{\{i,j\}} - y_i = 0, \quad \forall i \in V - \{n\}, \tag{18}$$

$$y_i \geqq 0, \text{ celé} \quad \forall i \ \in V - \{n\} \tag{19}$$

Relaxácia modelu BLP je tu formulovaná v tvare LP úlohy (15)–(17). Podmienky celočíselného jadra (18) a (19) sú len prepisom podmienok (13) a (14) v reči hrán grafu $G = (V,H,c)$. Podarilo sa nám tak nahradiť exponenciálny počet obmedzení v Edmondsovej podmienke (3) alternatívnymi podmienkami celočíselného jadra.

Pre implementáciu modelov BLP a YLP sa ukazuje výhodným reprezentovať hrany $\{i,j\} \in H$ takými usporiadanými dvojicami (i,j), pre ktoré platí $i < j$. Potom môžeme hrany okolia vrcholu $k \in V$ definovať vzťahom

$$\delta(k) = \{(i,j) \in H : i = k \; or \; j = k\}.$$

4 Riešič glpsolve nástroja GLPK

Riešič *glpsolve* nástroja GLPK pre úlohy LP resp. MILP môže používať viacero modelovacích jazykov. My sme zvolili modelovací jazyk *GNU MathProg*, ktorého syntax je intuitívna, ako vidieť z nasledujúceho textového súboru YLP.mod. Súbor obsahuje modelové špecifikácie optimalizačnej úlohy YLP a datové špecifikácie konkrétnej inštancie úlohy.

Modelová špecifikácia definuje potrebné parametre a samotný model pomocou príkazov jazyka, pričom obmedzeniu (17) zodpovedá príkaz *s.t. match* a obmedzeniu (18) príkaz *s.t. kernel*. Datová špecifikácia obsahuje príkaz na určenie počtu vrcholov a zoznamu ocenených hrán uvažovanej inštancie – v našom prípade ilustračného grafu so 6-timi vrcholmi. Poznamenajme, že ak by sme z modelu odstránili premenné y a podmienku *s.t. kernel* dostaneme relaxovanú úlohu BLP, ktorej riešenie je neceločíselné.

```
/* MWPM Minimum Weight Perfect Matching */
# glpsol --math YLP.mod
```

```
param n, integer, >= 2;
set V := {1..n};
set H, within V cross V;
param c{(i,j) in H};

var x{(i,j) in H}, >=0;
var y{i in V: i<n}, >=0, integer;
s.t. match{k in V}: sum{(i,j) in H: i=k or j=k} x[i,j]  = 1;
s.t. kernel{i in V: i<n}: sum{(i,j) in H} j*x[i,j]  - y[i] = 0;
minimize obj: sum{(i,j) in H} c[i,j]*x[i,j];
solve;

/*=====================================*/
printf "\nCena  ---> %d\n",sum{(i,j) in H} c[i,j]*x[i,j] ;
printf("  { i    j }  c[i,j] \n====================\n");
for {(i,j) in H: x[i,j] >0} printf " %4d %4d %6d\n",i,j,c[i,j];
printf "\n";

/*=====================================*/
data;
param n := 6;
param : H : c :=
1   2   1
1   3   1
1   5   2
2   3   1
2   4   2
2   6   2
3   5   2
4   5   1
4   6   1
5   6   1
;
end;
```

Teraz už len stačí spustiť v prikazovom riadku riešič prikazom *glpsol –math YLP.mod* a dostaneme nasledujúci výpis:

```
Reading model section from YLP.mod...
Reading data section from YLP.mod...
49 lines were read
Generating match...
Generating kernel...
Generating obj...
Model has been successfully generated
ipp_basic_tech:  1 row(s) and 0 column(s) removed
ipp_reduce_bnds: 2 pass(es) made, 15 bound(s) reduced
```

```
ipp_basic_tech:  0 row(s) and 0 column(s) removed
ipp_reduce_coef: 1 pass(es) made, 0 coefficient(s) reduced
glp_intopt: presolved MIP has 11 rows, 15 columns, 35 non-zeros
glp_intopt: 5 integer columns, none of which are binary
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  6.000e+00  ratio =  6.000e+00
Problem data seem to be well scaled
Crashing...
Size of triangular part = 10
Solving LP relaxation...
      0: obj =   6.000000000e+00  infeas =  1.000e+00 (1)
*     2: obj =   5.000000000e+00  infeas =  0.000e+00 (0)
*     9: obj =   3.000000000e+00  infeas =  0.000e+00 (0)
OPTIMAL SOLUTION FOUND
Integer optimization begins...
+     9: mip =     not found yet >=              -inf        (1; 0)
+    21: >>>>>   4.000000000e+00 >=   3.333333333e+00  16.7% (9; 0)
+    43: mip =   4.000000000e+00 >=     tree is empty   0.0% (0; 27)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (179625 bytes)

Cena  ---> 4
  { i    j }  c[i,j]
====================
    1    3      1
    2    4      2
    5    6      1

Model has been successfully processed
```

Ak chceme experimentovať s rôznymi inštanciami úlohy, potom stačí vytvoriť textový súbor MWPM.dat, ktorý obsahuje len datovú špecifikáciu t. j. *data . . . end;* a spustiť program *glpsol –math YLP.mod –data MWPM.dat*. Riešič má k dispozícii množstvo ďalších prepínačov, pomocou ktorých môžeme meniť voľbu stratégií vetvenia, rezných nadrovín atď.

5 Záver

Naše experimenty s nástrojom GLPK riešiča pre úlohy LP/MILP nás utvrdili v dobrej skúsenosti s OSS. Študovaný problém MWPM bol pomerne jednoduchý na formuláciu, ale dostatočne obtiažny na riešenie. Pri experimentálnom hľadaní vhodného modelu sa nám podarilo, vďaka dobre zdokumentovanému a stabilnému sofvéru, získať novú formuláciu optimalizačnej úlohy, ktorá využíva pri riešení ideu celočíselného jadra úlohy MILP.

Poďakovanie

Táto práca vznikla s podporou grantovej agentúry VEGA v rámci riešenia projektu 1/0135/08 „Optimalizačné problémy v logistických a dopravných systémoch“.

Literatúra

[1] COOK, W. – ROHE, A.: 1999. *Computing Minimum–Weight Perfect Matching*. INFORMS Journal on Computing, Vol. 11, No. 2, Spring, 1999

[2] MARKHORIN, A.: 2008. *GNU Linear Programming Kit*, Moscow : Reference Manual, Verson 4.29, pp. 154, 2008

[3] PLESNÍK, J.: 1983. *Grafové algoritmy*, Bratislava : VEDA, 1983

[4] PALÚCH, S. – PEŠKO, Š.: 2007. *Kvantitatívne metódy v logistike*, Žilina : EDIS, 2007, ISBN 80-8070-636-0,

[5] PEŠKO, Š.: 2009. *Minimum Integer Kernel for Integer Linear Programming*, Mathematical Programming, in preparation

Kontaktná adresa

Štefan PEŠKO (doc., RNDr., CSc.),
Katedra matematických metód, FRI ŽU v Žiline, Univerzitná 8215/1,
010 26 Žilina,
stefan.pesko@fri.uniza.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

VTK – VIZUALIZAČNÝ NÁSTROJ S OTVORENÝM KÓDOM

ŠRÁMEK, Miloš, (SK)

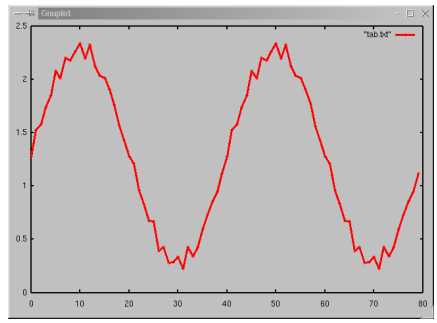
1 Úvod

Jedným z najvýraznejších trendov posledných rokov je enormný nárast objemu údajov, ktoré sú získavané simuláciou, meraním, alebo len jednoduchým zaznamenávaním udalostí. Príkladom môžu byť dáta, ktoré sú simulované na predpovedanie počasia a ktorých objem dosahuje až 1 PB (P=peta, teda 10^{15} bytov) ročne. Obdobný rozsah dát produkujú senzory snímačov v satelitnej astronómii alebo v časticovej fyzike.

Veľké objemy dát však nie sú doménou len veľkých projektov. Výsledkom dnes už bežne dostupného tomografického vyšetrenia bývajú desiatky až stovky miliónov číselných údajov. Samozrejme, že analýza takýchto objemov dát v ich pôvodnej numerickej forme nie je mysliteľná. Preto, v súlade so starou pravdou, že obrázok je hoden tisícky slov, sa často obraciame na možnosť prezentácie takýchto dát vo forme obrázkov. Tie na jednej strane skrývajú presnosť každého konkrétneho vstupného čísla, ktorú nahradzujú farbou, jasom či vhodným geometrickým tvarom. Na druhej strane však takáto abstraktná reprezentácia dokáže lepšie vypovedať o dátach ako celku, o ich charakteristických trendoch a črtách, čo používateľovi môže poskytnúť viac ako v prípade presných numerických údajov. Príkladom môže byť Obr. 1. Na jeho ľavej strane, v tabuľke, sú uvedené hodnoty namerané v 80 bodoch. Z tabuľky dokážeme, napríklad, presne zistiť, že v 15. meraní sme získali hodnotu 2.01. O dátach ako celku však toho veľa z tabuľky zistiť nedokážeme. Tu nám skôr pomôže vizuálna reprezentácia vo forme grafu na pravej strane obrázka. Ihneď vidíme, že ide o zašumený sínusový priebeh, pričom ľahko určíme aj jeho amplitúdu, periódu a úroveň šumu – teda údaje, ktoré nás obvykle zaujímajú najviac.

Vizuálnou prezentáciou číselných, ale aj nečíselných údajov sa zaoberá *vizualizácia*. Na jednej strane, z uhla pohľadu výskumníka, je to vedný odbor, ktorého primárnym cieľom je

0	1.28	20	1.28	40	1.28	60	1.28
1	1.52	21	1.21	41	1.52	61	1.21
2	1.58	22	0.96	42	1.58	62	0.96
3	1.74	23	0.84	43	1.74	63	0.84
4	1.85	24	0.68	44	1.85	64	0.68
5	2.08	25	0.67	45	2.08	65	0.67
6	2.01	26	0.39	46	2.01	66	0.39
7	2.21	27	0.43	47	2.21	67	0.43
8	2.18	28	0.28	48	2.18	68	0.28
9	2.26	29	0.29	49	2.26	69	0.29
10	2.34	30	0.34	50	2.34	70	0.34
11	2.2	31	0.22	51	2.2	71	0.22
12	2.33	32	0.43	52	2.33	72	0.43
13	2.12	33	0.34	53	2.12	73	0.34
14	2.04	34	0.42	54	2.04	74	0.42
15	2.01	35	0.6	55	2.01	75	0.6
16	1.91	36	0.73	56	1.91	76	0.73
17	1.77	37	0.86	57	1.77	77	0.86
18	1.56	38	0.94	58	1.56	78	0.94
19	1.43	39	1.12	59	1.43	79	1.12



Obr. 1: Číselná (vľavo) a grafická (vpravo) reprezentácia dát

vývoj a skúmanie metód na transformáciu vstupných dát do formy, ktorú dokážeme vnímať. Na druhej strane, z uhla pohľadu používateľa, vizualizácia predstavuje celkom konkrétne postupy, ktoré treba aplikovať, aby sme naše dáta zobrazili a tak získali možnosť na ich analýzu, pochopenie a prezentáciu. Aj keď vizualizácia dát je možná aj bez použitia počítača (papier, ceruzka, pravítko), bez počítačov by vizualizácia nikdy nebola tým, čím je dnes.

Pre používateľa je v prvom rade zaujímavá otázka, či je dostupný program, pomocou ktorého dokáže zobraziť svoje dáta. Niet pochýb, možností je veľa. Odhliadnuc od jednoduchých programov na kreslenie dvoj- či trojrozmerných grafov (Gnuplot, Kpl, ...), používatelia majú k dispozícii komerčné (Iris Explorer, AVS), ale aj voľne dostupné vizualizačné systémy ako Data Explorer či Mevislab a najmä veľmi populárny systém VTK, ktorým sa budeme v príspevku venovať. Okrem iného aj preto, lebo je to zaujímavý open-source projekt.

2 VTK – The Visualization Toolkit

VTK je open-source systém na počítačovú grafiku, spracovanie obrazu a vizualizáciu. Pozostáva v prvom rade z knižnice C++ tried, ktoré poskytujú základné nástroje na spracovanie rôznych typov dát, doplnené rozhraním pre interpretované jazyky Python, Java alebo TCL/Tk. Tie používateľovi umožňujú pohodlnejšie zostavovanie vlastných programov určených na riešenie daného problému. VTK dokáže spracovávať skalárne, vektorové a tenzorové údaje, ktoré sú definované nad rôznymi typmi dvoj-, troj a viacrozmerných mriežok, a to od neštruktúrovaných (údaje sú snímané s ľubovoľnou polohou) až po pravouhlé. Podporuje paralelizmus, pri ktorom sa spracovanie dát rozloží na viacero procesorov (prípadne spolupracujúcich počítačov) a tzv. prúdové spracovanie, pri ktorom sú dáta rozdelené

na menšie bloky, ktoré sa spracovávajú postupne na jednom alebo na viacerých procesoroch. Druhá možnosť zabezpečí, že na danom počítači možno spracovať úlohu ľubovoľnej veľkosti – samozrejme, že v primerane dlhšom čase.

VTK je softvér krytý veľmi voľnou licenciu[1], ktorá povoľuje šírenie programu v zdrojovej alebo binárnej forme, za podmienky (odhliadnuc od ďalších, menej podstatných), že bude distribuovaný spoločne so samotnými licenčnými podmienkami. Táto licencia bola prevzatá z operačného systému BSD [2]. BSD licencia je jednou z najznámejších open-source licencií, ktoré povoľujú ďalšie šírenie softvéru bez podmienky zverejňovania zdrojového kódu modifikácií a odvodeného softvéru. Takto sa VTK môže používať nielen v open-source produktoch, ale aj v komerčných riešeniach, pri ktorých sa zdrojový kód nezverejňuje. Táto črta odlišuje open-source (otvorený) softvér od slobodného softvéru, pri ktorom sa požaduje, aby odvodený softvér, poskytovaný používateľovi, bol krytý rovnakou licenciou ako pôvodný softvér (tu obvykle ide o niektorú z verzií licencie GNU General Public License).

Systém VTK bol pôvodne vytvorený ako sprievodný softvér ku knihe o počítačovej grafike a vizualizácii trojice autorov Schroeder, Martin a Lorensen, ktorí vtedy boli zamestnancami spoločnosti General Electric [5]. Autori však knihu aj softvér napísali vo svojom voľnom čase (so súhlasom spoločnosti), takže autorské práva zostali u nich. Vďaka svojej licencii si tento softvér rýchlo získal priaznivcov, ktorí ho nezačali len používať, ale aj sami prispievali k vývoju. VTK neskôr začala podporovať aj spoločnosť GE a ďalšie organizácie. GE začala dokonca aj predávať softvérové produkty založené na VTK – samotné VTK však (samozrejme) nie. Časť autorov neskôr GE opustila a založila spoločnosť Kitware, ktorá sa zaoberá poskytovaním služieb, súvisiacich s VTK a neskôr s ďalšími open-source produktmi a vývojom komerčného softvéru na báze svojich otvorených projektov.

Autori VTK pri vývoji svojho softvéru využili viaceré postupy a technológie [2]. V prvom rade, VTK je open-source projekt. Open-source môžeme vnímať ako technológiu vývoja softvéru, kde jedným zo základných pravidiel je zásada „zverejňuj skoro, zverejňuj často“. Používatelia sú pritom začlenení do „virtuálneho vývojárskeho tímu“ tým, že prispievajú ďalšími návrhmi, upozorňujú na chyby a prípadne ich aj odstraňujú alebo dokonca aj prispievajú svojim vlastným kódom. V tomto open-source technológia v mnohom pripomína technológiu tzv. agilných vývojových metód[3] (agile methodologies), v ktorých sa smerovanie vývoja určuje priamo počas samotného vývoja, a nie vopred vo fáze definovania cieľa, ktorá je obvykle prvým krokom v tradičnom postupe vývoja softvéru. Motivácia pre tento postup práce vyplýva najmä z toho, že v prípade vývoja špeciálneho softvéru v neprebádanej oblasti je ťažko možné vopred presne stanoviť ciele, ktoré má vývoj dosiahnuť. Tento prístup umožňuje modifikovať ciele za behu a priebežne ich prispôsobovať požiadavkám zákazníka – pričom aj samotné jeho požiadavky sa vyvíjajú v procese vývoja.

Softvér, ak sa má použiť v náročných aplikáciách, musí byť kvalitný. V tejto oblasti vývojári VTK využili niektoré postupy tzv. extrémneho programovania, ktoré zdôrazňuje význam

[1] `http://www.vtk.org/VTK/project/license.html`
[2] `http://en.wikipedia.org/wiki/BSD_licenses`
[3] `http://agilemanifesto.org/`

testovania. Konkrétne, využili tzv. testom riadený vývoj, kde testy správnosti sa vyvíjajú súčasne s programom, ba niekedy aj skôr. Napísaním testu sa vlastne stanoví špecifikácia, ktorú potom samotný program implementuje. Existencia testov umožňuje pravidelné vykonávanie regresných testov, ktoré sa v prípade VTK vykonávajú každý deň (či skôr, každú noc). Výsledok testov je dostupný na druhý deň ráno, takže autor má možnosť si overiť, či jeho zmeny mali požadovaný účinok. Testy VTK sa vykonávajú až v 50 rôznych verziách na rôznych platformách[4]. Tento postup zabezpečuje rýchle odstraňovanie chýb (obvykle do 24 hodín) a stálu funkčnosť vývojárskej verzie softvéru.

3 Použitie VTK

3.1 Inštalácia

VTK nebýva súčasťou základnej inštalácie operačného systému, treba si ho doinštalovať. Je dostupný pre všetky bežné operačné systémy. Zdrojový kód poslednej stabilnej verzie možno získať zo stránok `http://www.vtk.org/VTK/resources/software.html`. Pre Ubuntu (a Debian) Linux je dostupná aj binárna verzia VTK, ktorú spolu s príkladmi, dátami a dokumentáciou nainštalujeme príkazom

```
sudo apt-get install vtklib5 vtk-python vtk-examples vtkdata vtk-doc
```

Balíky sú v úložisku `universe` a sú dosť rozsiahle. VTK je na príslušnej stránke dostupný v binárnej verzii aj pre Windows. Ak pre iné prostredia nie je binárna verzia k dispozícii, alebo ak chceme použiť najnovšiu verziu, systém je potrebné preložiť.

3.2 Jednoduchý príklad

Na zobrazenie (vizualizáciu) vstupných dát treba, bez ohľadu na ich typ, vykonať niekoľko základných operácií a definovať niekoľko objektov. Prvým krokom je vytvorenie objektu na reprezentáciu zdroja, ktorým môžu byť buď dáta zo súboru, alebo ktorý môže byť definovaný priamo vo VTK. Na obrázku 2 je vstupným objektom kužeľ, definovaný interne prostredníctvom metódy `vtkConeSource`. Takýto objekt zatiaľ zobraziť nevieme, existuje vo svojej pôvodnej abstraktnej forme. Na ďalšie zobrazenie ho musíme skonvertovať na množinu zobraziteľných primitív, akými sú trojuholníky, úsečky, body a iné (v tomto prípade bude kužeľ reprezentovaný ihlanom). Táto operácia sa nazýva mapovanie a je súčasťou každého vizualizačného programu. Samotné mapovanie môže byť v závislosti na vstupe zložitou operáciou, pri ktorej sa, napríklad, môže povrch objektu zosnímaného tomografom aproximovať sieťou tisícok trojuholníkov.

[4] Výsledky nočných testov VTK zverejnené na stránke `http://www.cdash.org/CDash/index.php?project=VTK`

Riadok

```
coneMapper.SetInput(cone.GetOutput())
```

demonštruje základnú koncepciu spracovania dát vo VTK. Vidíme, že objekt `cone` má akýsi výstup, ktorý sa pripája na vstup objektu `coneMapper`. VTK je vizualizačný systém založený na toku dát – tie „pretekajú“ *filtrami*, kde sa transformujú. VTK filtre sú v mnohom podobné filtrom, ako ich poznáme z unixových operačných systémov. V našom prípade ide o transformáciou ihlanu, ktorý je definovaný svojimi rozmermi a počtom stien, na množinu polygónov.

Takto vytvorený objekt by sme už mohli zobrazovať, potrebujeme však okno (*vtkRenderWindow*), v ktorom sa to bude diať a nástroj, ktorý to dokáže (*vtkRenderer*). S objektom budeme chcieť manipulovať, a tak pridáme manipulátor (*vtkReaderWindowInteractor*). Zatiaľ nám však chýbajú zobrazovacie parametre, akými napríklad sú uhol pootočenia, zväčšenie, farba a podobne. Tie spravuje „herec“ (vtkActor), ktorého pridáme a priradíme mu náš objekt, konvertovaný na polygóny. Posledným krokom je pridanie herca k rendereru a spustenie renderovania a interakcie.

Vidíme, že použitie VTK spočíva v špecifikácii a správnom zoradení modulov, ktoré zabezpečia potrebné a požadované operácie. Horeuvedená základná štruktúra vizualizačného programu pritom zostáva rovnaká, menia sa len jednotlivé moduly – triedy, pomocou ktorých dosiahneme požadovaný účinok. Napríklad (Obr. 3), zmenou dvoch riadkov, ktoré na Obr. 2 definujú kužeľ ako vstupný objekt, za načítanie tomografických dát, špecifikáciu povrchu objektu prostredníctvom prahovej hodnoty a vytvorenie trojuholníkového modelu povrchu dostaneme obrázok objektu zosnímaného tomografom – v tomto prípade ľudskej hlavy. Ostatné časti programu pritom zostanú rovnaké.

4 Softvér na báze VTK

V predchádzajúcej časti sme videli, ako možno VTK použiť na vizualizáciu priamo využitím základných nástrojov, ktoré VTK poskytuje prostredníctvom knižnice tried. Táto možnosť je však dostupná len pre skúsenejších používateľov, ktorí navyše zvládajú programovanie v niektorom z jazykov, ktoré systém podporuje (Python, C++, Java a Tk/Tcl). Triedy VTK sú však priamo navrhnuté tak, aby umožňovali vytváranie nadstavieb, ktoré zložitosť programovania skryjú za používateľsky prívetivejšie grafické rozhranie. Keďže VTK je open-source systém, takýchto nadstavieb vzniklo viac, a to rovnako komerčných ako aj voľne dostupných s otvoreným kódom[5].

Slicer Program Slicer[6] poskytuje nástroje na segmentáciu, registráciu a trojrozmernú vizualizáciu multimodálnych dát so zameraním na vizualizáciu dát pochádzajúcich zo štúdii

[5] na stránke `http://www.vtk.org/Wiki/VTK_Tools` je zoznam 27 aplikácií využívajúcich VTK

[6] `http://www.slicer.org`

```
#! /usr/bin/python

# nacitanie potrebnych rozsireni VTK
from vtk import *

# Vytvorenie abstraktneho objektu, ihlanu
cone = vtkConeSource()
cone.SetResolution(12)

# Priprava objektu na renderovanie jeho
# transformaciou na renderovatelne polygony
coneMapper = vtkPolyDataMapper()
coneMapper.SetInput(cone.GetOutput())

# Vytvorenie renderovacieho okna
renWin = vtkRenderWindow()
renWin.SetSize(300,300)

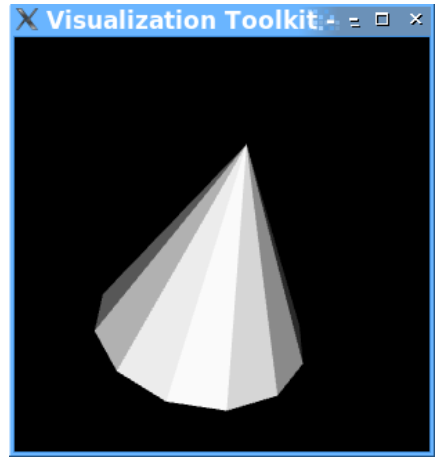
# Vytvorenie renderera
ren = vtkRenderer()
renWin.AddRenderer(ren)

# Povolenie zakladnej interakcie
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

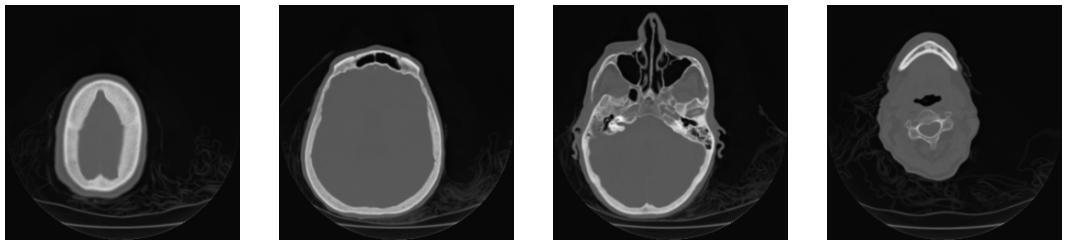
# Spojenie renderovatelneho objektu so scenou
# prostrednictvom herca (actor)
coneActor = vtkActor()
coneActor.SetMapper(coneMapper)

# Pridanie objektu (herca) k~rendereru
ren.AddActor(coneActor)

# spustenie renderovania a~interakcie
iren.Initialize()
iren.Start()
```



Obr. 2: Ukážka jednoduchého VTK kódu v jazyku Python

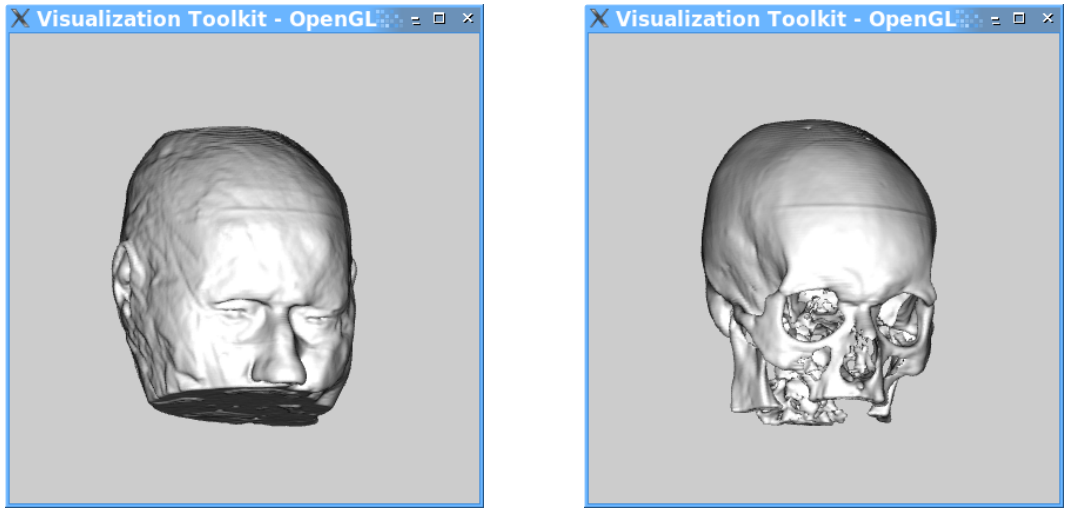


...

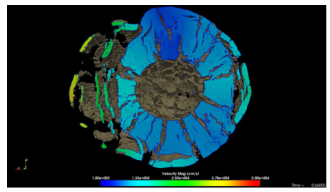
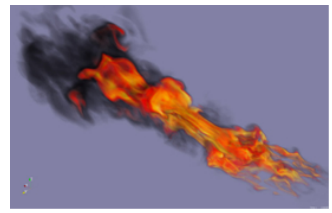
```
# Nacitanie vstupneho suboru s~rozmermi 170x190x184 bodov
# a~nastavanie parametrov
reader = vtkImageReader()
reader.SetFileDimensionality(3)
reader.SetFileName("tot2.raw")
reader.SetDataScalarTypeToUnsignedChar()
reader.SetDataExtent(0,169,0,189,0,183)

# specifikacia objektu prostrednictvom jasovej urovne 'prahova_hodnota'
skinExtractor = vtk.vtkContourFilter()
skinExtractor.SetInput(reader.GetOutput())
skinExtractor.SetValue(0, prahova_hodnota)
```

...



Obr. 3: Jednoduchou zmenou definície objektu z Obr. 2 môžeme zobraziť objekt snímaný tomografom. Hore: štyri zo 184 rezových obrázkov, z ktorých pozostávajú vstupné dáta. Stred: Načítanie dát do VTK, Dolu: Zobrazený trojuholníkový model vytvorený zo vstupných dát. Model bol vytvorený tak, že jasová hladina na úrovni `prahova_hodnota` bola "vydláždená" trojuholníkmi. Vľavo: povrch pokožky, prahova_hodnota=40, vpravo: povrch kosti, prahova_hodnota=100

Obr. 4: Vľavo: vizualizačný klaster RedRose s 264 uzlami v Sandia NL, stred: šírenie trhlín pri výbuch v centre asteroidu, vpravo: simulácia horenia (`http://www.psc.edu/general/software/packages/paraview/`)

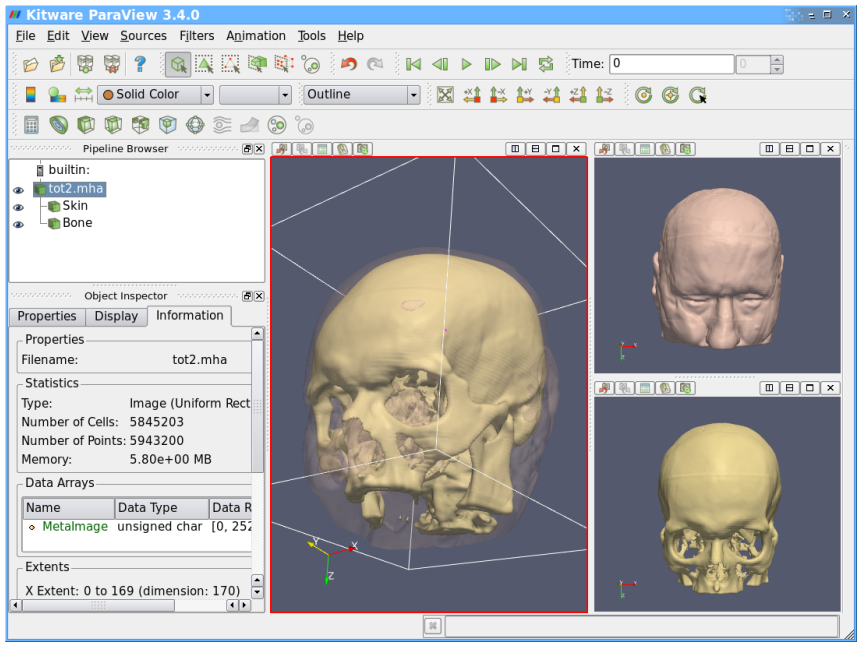
v oblasti medicínskeho zobrazovania (difúzne tenzorové zobrazovania, funkčná magnetická rezonancia) a iné biomedicínske aplikácie.

ParaView Paraview[7] je škálovateľný vizualizačný program určený pre širokú paletu platforiem od jednoprocesorových desktopov, cez viacprocesorové systémy so zdieľanou pamäťou až po počítačové klastre. Je vybavený viacerými vizualizačnými algoritmami na zobrazovanie tokov a na povrchové a objemové zobrazovanie a podporuje zobrazovanie na multiobrazovkových displejoch a stereoobrazovkách. Pri behu na paralelnom systéme Paraview dokáže pracovať s veľmi rozsiahlymi statickými aj dynamickými dátami definovanými na rôznych typoch mriežok (pravouhlé, štruktúrované, neštruktúrované).

ParaView [7] bol vytvorený autormi VTK (spoločnosť Kitware) v spolupráci s Los Alamos National Laboratory. Motiváciou od začiatku bolo vyvinúť škálovateľný vizualizačný nástroj s podporou paralelizmu a distribuovanej pamäte. Jeho vývoj neskôr podporili aj ďalšie národné laboratóriá a iné organizácie v USA. Paraview sa využíva najmä na vizualizáciu výsledkov rozsiahlych simulácií. Napríklad v Sandia National Laboratory v USA sa používa na vizualizácie výsledkov simulácií, ktoré sa robia na 10 000 uzlovom klastri, pričom sám Paraview beží na 264 uzloch s dvoma procesormi AMD a grafickým akcelerátorom NVidia Quadro (Obr. 4 vľavo). Príkladom aplikácií je vizualizácia výsledku simulácie výbuchu nálože s energiou ekvivalentnou 10 megatonám TNT v ťažisku asteroidu Golevka (mriežka s 1.1 mld. buniek) alebo vizualizácia výsledkov simulácie horenia so 150 miliónmi stupňov voľnosti.

VTK Designer 2 Program VTK Designer[8] je grafický editor na vytváranie aplikácií v prostredí VTK. Výstupom je program, ktorý sa zapisuje v ľubovoľnom z jazykov, ktoré VTK podporuje. VTK Designer je určený pre výskumníkov, ktorí potrebujú zostavovať zložité vizualizácie, ale aj pre vývojárov, ktorí chcú využiť VTK vo svojich vlastných aplikáciách. Charakteristickou črtou programu je možnosť jednoducho zostavovať programy z modulov prostredníctvom grafického rozhrania.

[7] `http://www.paraview.org`
[8] `http://www.vcreatelogic.com/oss/vtkdesigner/`



Obr. 5: Použitie Paraview na vizualizáciu tomografických dát. Zobrazené sú povrchy dvoch tkanív – pokožky a kosti (vpravo). V strede sú zobrazené v perspektívnom náhľade, pričom pokožka je priehľadná

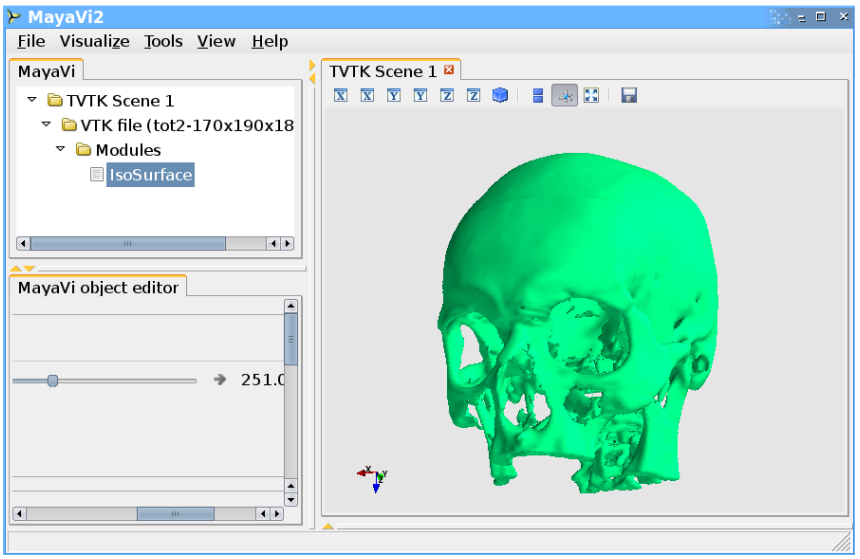
Mayavi2 Mayavi2[9] je mnohoúčelový a viacplatformový nástroj na vizualizáciu 3D dát, so zameraním najmä na vizualizáciu skalárnych, vektorových a tenzorových dát v dvoch a troch rozmeroch, ľahké skriptovanie pomocou Pythonu a ľahké rozširovanie pomocou vlastných modulov a filtrov. Súčasťou balíka je grafická aplikácia `mayavi2`, ktorá umožňuje zostavovanie vizualizácií bez potreby programovania, rovnako ako aj nástroje, ktoré umožňujú mayavi využiť ako kresliaci stroj v iných aplikáciách.

5 Ďalšie OSS programy

Pre autorov VTK a aj pre ich spoločnosť Kitware je open-source základným vývojovým modelom, v ktorom do svojej práce zapájajú komunitu vývojárov-dobrovoľníkov. Okrem základného produktu, VTK, v spolupráci s komunitou boli vytvorené aj ďalšie nástroje na spracovanie dát, ale aj na podporu samotného vývoja softvéru.

The Insight Toolkit – ITK Cieľom vizualizácie je prezentácia dát vo forme obrázkov. Častokrát však pred samotnou vizualizáciou treba dáta upraviť alebo analyzovať, čím sa získa odvodená informácia, ktorá je neskôr jadrom samotnej vizualizácie. Táto fáza tzv.

[9] `https://svn.enthought.com/enthought/wiki/MayaVi`



Obr. 6: Použitie programu Mayavi2 na vizualizáciu tomografických dát. Zobrazený je povrchy kostného tkaniva

predspracovania hrá mimoriadne dôležitú úlohu pri vizualizácii objemových dát, ktoré sú často používané v medicínskej diagnostike (tomografické metódy) a biológii (konfokálna mikroskopia). Objemové dáta sú vlastne postupnosťou dvojrozmerných obrázkov, ktoré zobrazujú priečne rezy snímaným objektom. Takýchto obrázkov možno pri jednom meraní získať až 2000, pričom jeden obrázok môže mať až 1000^2 (alebo aj viac) pixelov.

ITK [3] je nástroj na predspracovanie objemových dát so zameraním najmä na úpravu (filtrovanie, odstraňovanie šumu), segmentáciu a registráciu (priestorové zosúladenie dvoch alebo viacerých dátových objemov do spoločného súradnicového systému tak, aby ich bolo možné spolu ďalej spracovávať alebo zobrazovať). Pri *segmentácii* ide o identifikáciu objektov a tkanív v dátach. Príkladom môže byť detekcia tumoru v trojrozmernom obraze pečene. *Registrácia* je transformácia dvoch dátových objemov do spoločného súradnicového systému, ktorá je potrebná vtedy, ak je pacient snímaný viackrát, buď s cieľom zistiť vývoj choroby alebo jej liečenia, alebo ak bol pacient snímaný viacerými snímačmi, ktoré poskytujú doplnkovú informáciu.

ITK nemá prostriedky na vizualizáciu. Prepojovacie triedy však umožňujú spracovanie výstupov ITK modulov modulmi VTK a naopak. Možno teda povedať, že oba systémy sa v oblasti spracovania a vizualizácie objemových dát dopĺňajú. ITK má mnohé spoločné črty s VTK – licenciu, vývojový proces, štruktúru tried, podporované jazyky a tiež aj to, že je využívaný vo viacerých ďalších komerčných a otvorených systémoch (Mevislab, SCIRUN, [1]).

CMake VTK bol od začiatku vyvíjaný ako multiplatformový systém, s cieľom jeho použitia na všetkých hlavných operačných systémoch. Aj keď je jadro VTK napísané v jazyku C++, v rôznych systémoch sa používajú rôzne vývojové nástroje – prekladače, čo sťažuje prenos softvéru medzi jednotlivými systémami, alebo dokonca aj medzi rôznymi prekladačmi na jednom systéme. Navyše, VTK je rozsiahly systém so stovkami tried a státisícmi riadkov kódu. V čase vzniku VTK neexistoval nástroj, ktorý by umožnil písanie kódu tak, aby bol preložiteľný na všetkých cieľových platformách.

Z tohoto dôvodu autori VTK vytvorili systém `cmake` [4, 6], ktorý takýto preklad umožňoval. `cmake` je svojim účelom podobný dvojici nástrojov Automake/Autoconf, ktoré však podporujú platformovú nezávislosť len v prostredí unixových systémov a len pre kompilátory, použiteľné v programe `make`. `cmake` oproti tomu podporuje aj grafické vývojové prostredia (KDevelop, Eclipse, VisualC++ a ďalšie) tak, že z konfiguračného súboru `CMakeFile.txt` generujú buď súbor Makefile pre štandardné kompilátory používané na príkazovom riadku, alebo konfiguračné súbory jednotlivých IDE.

Úlohou programu CMake je testovanie aktuálneho systému, zistenie, aký kompilátor sa na ňom používa (v prípade prítomnosti viacerých je možný výber) a ktoré jeho nastavenia (prepínače) sú potrebné. Ďalej, zistí, kde sa nachádzajú hlavičkové súbory (v prípade prekladu programov v jazykoch C alebo C++) a potrebné knižnice. V prípade potreby dokáže generovať zdrojový kód, ktorý sa použije v preklade a napokon určí, kam sa majú uložiť výsledky prekladu.

`cmake` dnes postupne nahrádza už zastarávajúcu dvojicu Automake/Autoconf aj pri preklade ďalších systémov, akými sú, napríklad, desktopové prostredie KDE a KDE aplikácie[10].

CDash CDash je sieťový server určený na testovanie softvéru, ktorý sa vyvinul z programu Dart [6]. CDash organizuje testovanie a zbiera, analyzuje a zobrazuje jeho výsledky, ktoré môžu byť posielané klientmi umiestnenými na ľubovoľnom mieste na Internete. Výsledky sú zobrazované v podobe webovej stránky, teda vývojári majú neustály prehľad o aktuálnom stave vyvíjaného softvéru. CDash je súčasťou systému na vývoj softvéru, ktorý zahŕňa open-source programy CMake a jeho doplnkové nástroje CTest a CPack od spoločnosti Kitware ako aj ďalšie externé nástroje na návrh, správu a údržbu rozsiahlych softvérových systémov (napr. Doxygen[11] na generovanie dokumentácie z komentovaného kódu, Valgrind[12] na testovanie správnosti použitia pamäte). Príkladom použitia Servera CDash je prehľadová stránka o stave vývoja samotného CDash a o ďalších projektoch[13].

[10] `http://techbase.kde.org/Development/Tutorials/CMake`
[11] `http://www.doxygen.org`
[12] `http://www.valgrind.org`
[13] `http://www.cdash.org/CDash/`

6 Záver

V príspevku sme si ukázali softvérový systém VTK, ktorý dnes v oblasti vizualizácie dát úspešne konkuruje komerčným produktom – ak ich svojimi vlastnosťami, schopnosťami a flexibilitou skôr nezatieňuje. VTK je súčasne aj významným open-source projektom, a ako taký je príkladom úspešného využitia princípov otvorenosti vo vývoji zložitých softvérových aplikácií.

Poďakovanie

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe Zmluvy č. APVV-20-056105.

Literatúra

[1] Ingmar Bitter, Robert Van Uitert, Ivo Wolf, Luis Ibanez, and Jan-Martin Kuhnigk. Comparison of four freely available frameworks for image processing and visualization that use itk. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):483–493, 2007.

[2] Ron Goldman and Richard P. Gabriel. *Innovation Happens Elsewhere: Open Source as Business Strategy*. Morgan Kaufman Publishers, 2005.

[3] Luis Ibanez, Will Schroeder, Lydia Ng, and Josh Cates. *The ITK Software Guide: The Insight Segmentation and Registration Toolkit*. Kitware Inc., first edition, 2003.

[4] Ken Martin and Bill Hoffman. *Mastering CMake*. Kitware Inc., 2008.

[5] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. Prentice Hall, 1996.

[6] William J. Schroeder, Luis Ibánez, and Ken Martin. Software process: The key to developing robust, reusable and maintainable open-source software. In *ISBI 2004*, pages 648–651, 2004.

[7] Amy Squillacote. *The Paraview Guide*. Kitware Inc., 2008.

Kontaktná adresa

Miloš Šrámek (doc. Ing., PhD),
Komisia pre vedeckú vizualizáciu,
Rakúska akadémia vied, Viedeň
milos.sramek@oeaw.ac.at

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

VOĽNE DOSTUPNÉ LMS

TRNKA, Andrej, (SK)

1 Úvod

Systémy na riadenie výučby (LMS) sú aplikácie, ktoré v sebe integrujú spravidla najrôznejšie on-line nástroje pre komunikáciu a riadenie štúdia (nástenka, diskusné fórum, chat, evidencie atď.) a zároveň sprístupňujú študentom učebné materiály či výukový obsah on-line alebo aj off-line.

Za bežné funkcie systémov riadeného vzdelávania sa môžeme považovať napr. moduly:

1. Evidencia a správa študentov,
2. evidencia a správa skupín študentov,
3. evidencia a správa kurzov,
4. evidencia a správa skupín kurzov,
5. katalóg resp. zoznam výukových kurzov a objektov,
6. správa študijných plánov – vytváranie podmienok medzi kurzami,
7. evidencia hodnotenia študentov – zverejnenie výsledkov,
8. testovanie študentov,
9. správa prístupových práv,
10. komunikačné nástroje,
11. autorské nástroje na vytváranie výukových kurzov a objektov,
12. úložisko výukového obsahu.

Pre všetky tieto funkcie je dôležitá požiadavka na ich prenositeľnosť a štandardizáciu. LMS by mal byť otvorený a schopný napríklad ľahko a rýchlo začleniť výukový obsah, vytvorený napríklad pred zavedením LMS. Medzi štandardizované formáty výučbových jednotiek patria SCORM, AICC, IMS, IEEE a Ariadne [1].

Často sa pojem LMS stotožňuje s pojmom Learning Content Management System (LCMS), ktorý navyše oproti LMS obsahuje širokú škálu nástrojov umožňujúcich tvorbu e-learningového obsahu.

2 Výber LMS

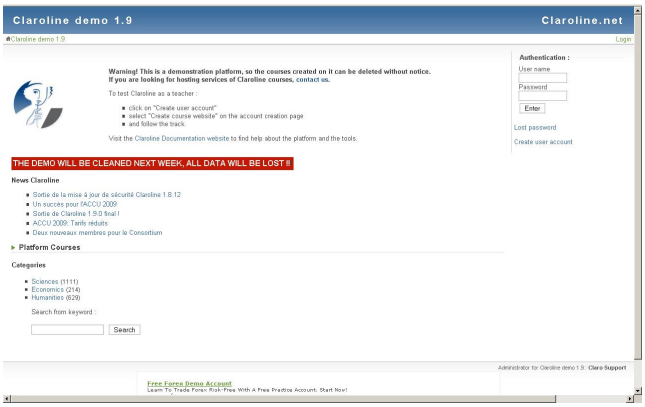
Výber LMS vhodného pre konkrétnu inštitúciu predpokladá v prvom rade určiť, bez ktorých funkcií sa organizácia nezaobíde a ktoré sú menej dôležité. Medzi dôležité sa môže zaradiť podpora blended learningu (kombinovaného vyučovania), integrácia s informačným systémom konkrétnej organizácie, riadenie pedagogického procesu, integráciu obsahu, dodržiavanie štandardov, nástroje na evalváciu, testovanie a hodnotenie [2].

Na celosvetovom trhu sa dnes pohybuje množstvo kvalitných vzdelávacích softvérových balíkov. Na webovej stránke [3] je uvedený zoznam open source LMS systémov. Vybrané z nich stručne opíšeme.

2.1 Claroline

Claroline má za cieľ pomáhať učiteľom vytvárať efektívne online kurzy a riadiť učenie a aktivity s tým spojené na internete.

Claroline Consortium vzniklo počas druhej konferencie používateľov Claroline, ktorá sa konala na Univerzite Vigo v Španielsku. Táto nezisková organizácia má vytýčený cieľ spájať komunitu Claroline, koordinovať vývoj platformy a vytvárať jej reklamu. Claroline má jednu z najväčších komunít používateľov a vývojárov na svete, čomu dosvedčuje aj to, že už bol preložený do 35 svetových jazykov.
Dostupnosť: `http://www.claroline.net/`

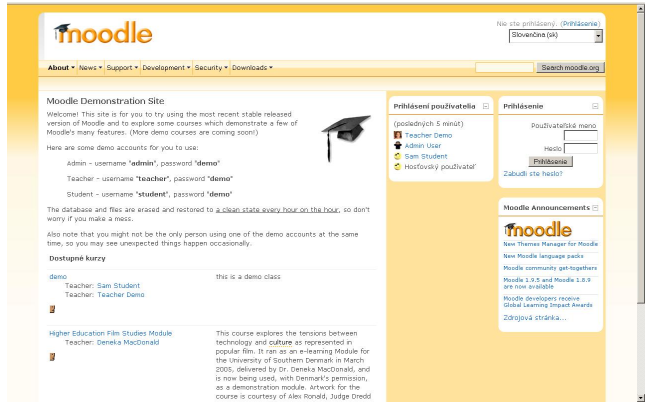


Obr. 1: LMS Claroline

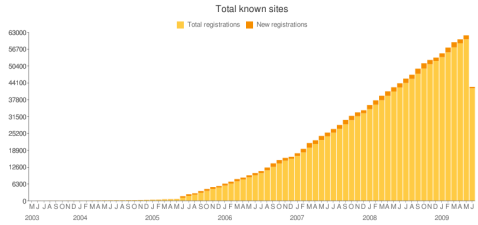
2.2 Moodle

Moodle sa vo svete stal veľmi populárnym. Tento LMS je aktívny a stále sa vyvíjajúci projekt. Ako sa Moodle rozširoval a jeho komunita rástla, začalo do neho prispievať veľa osôb z rôznych školiacich zariadení. Dnes je Moodle používaný nielen na univerzitách, ale aj iných inštitúciách. V tomto prostredí je možné vytvárať, ukladať, meniť a spracovávať obsah. Obsahuje nástroje na vytváranie a kompletizáciu obsahu, nástroje podporujúce ukladanie, nástroje podporujúce prístup k metadátam, jednoduchú správu profilu študenta atď. Ďalšou službou je konvertovanie obsahu (prevod dokumentu z textového procesora alebo prezentácie do výučbového materiálu).

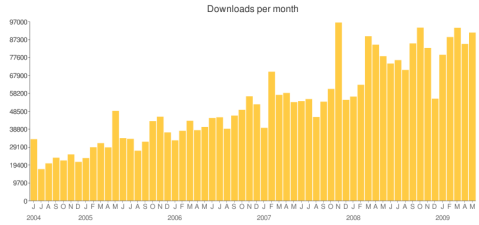
Stránka moodle.org je centrom pre informácie, diskusie a spoluprácu používateľov aplikácie Moodle (administrátorov, učiteľov, študentov či vývojárov). Na tejto stránke sa nachádza aj aktuálna verzia aplikácie Moodle. V Slovenskej republike je podľa moodle.org zaregistrovaných 226 webových lokalít využívajúcich LMS Moodle.
Dostupnosť: `http://moodle.org/`



Obr. 2: LMS Moodle



Obr. 3: Reprezentácia registrovaných stránok používajúcich Moodle v 210 krajinách sveta [4]

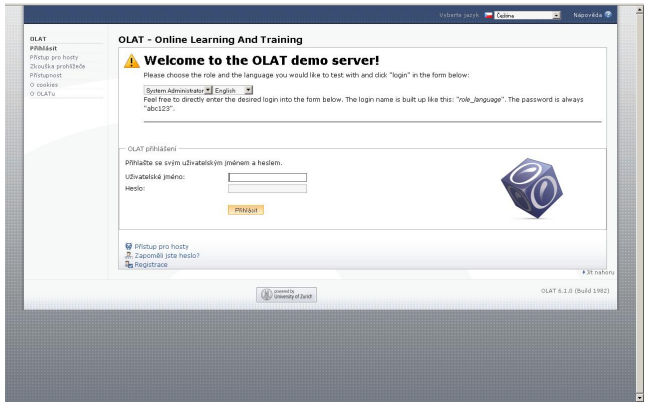


Obr. 4: Počet stiahnutí LMS Moodle [4]

2.3 Olat

Olat (Online Learning And Training) je založený na programe Java (od verzie OLAT 3.0). Jeho vývoj začal v roku 1999 na Univerzite v Zürichu vo Švajčiarsku, kde sa stal strategickým LMS a bol spustený na hlavnom Olat serveri.

Olat získava vo svete stále väčšiu pozornosť a to najmä v Európskej únii. K jeho najväčším používateľov patrí už spomínaná Univerzitu v Zürichu (s asi 50 000 študentami), Bildungsportal Sachsen (asi 40 000 študentov) a v lete 2009 sa k nim pripojí aj Univerzita v Hamburgu (30 000 študentov). Olat je veľmi podobný projektu Sakai (popísaný ďalej), no v porovnaní s americkou iniciatívou má oveľa dlhšiu dobu vývoja a bol v produkcii už niekoľko rokov. Olat bol od začiatku vyvíjaný ako aplikácia pre podporu veľkých univerzít a vysokých škôl a je zrovnateľný napríklad s plateným riešením LMS WebCT Vista. V neposlednej rade Olat ponúka preklady do viac ako 30 jazykov sveta.
Dostupnosť: `http://olat.org/`

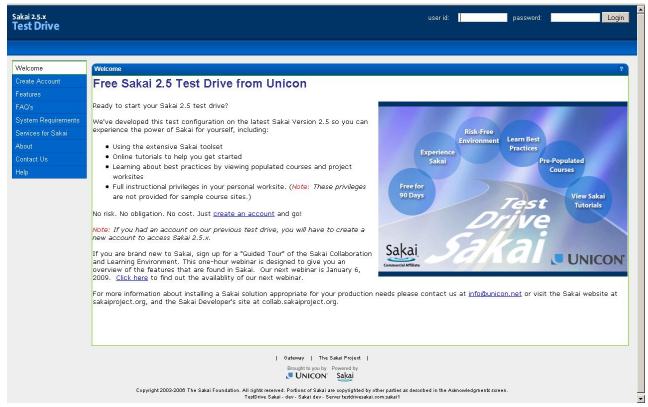


Obr. 5: LMS Olat

2.4 Sakai

Sakai vznikol v roku 2004 v USA, keď Univerzita Stanford, Michigan, Indiana, MIT a Berkeley spolu začali budovať spoločný LMS projekt miesto toho, aby každá z nich používala vlastný alebo si nejaký kupovala. Tieto univerzity pochopili, že je dôležitá vzájomná spolupráca vo vývoji a preto vytvorili Sakai ako tzv. CLE (Collaboration and Learning Environment). V dnešnej dobe je Sakai CLE používané viac ako 100 inštitúciami, pričom v každej je 200 až 200 000 používateľov.

Komunita Sakai používateľov je zodpovedná za všetky aspekty vývoja aplikácie Sakai CLE. Používatelia veria, že Sakai je vytvorený vyššou vzdelanosťou pre vyššiu vzdelanosť a preto sa z neho stane ten najlepší produkt pre využívanie na školách.
Dostupnosť: `http://sakaiproject.org/`



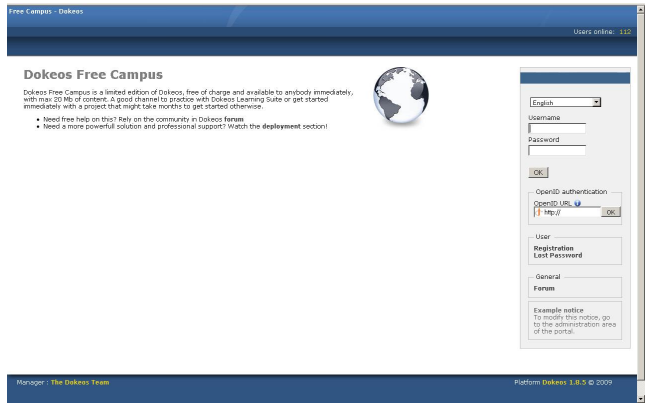
Obr. 6: LMS Sakai

2.5 Dokeos

Tento systém je zaujímavý tým, že využíva jadro zo systému Claroline. Dokeos so všetkými položkami pracuje ako s jednotlivými objektmi. Dokeos je napísaný v PHP a používa iba MySQL databázu. LMS Dokeos umožňuje vytvárať a spravovať rôzne výučbové kurzy.

Používatelia majú možnosť poslať mail ktorémukoľvek používateľovi, avšak nie priamo z prostredia, ale pomocou externých programov. Každý kurz ma fórum, do ktorého majú všetci študenti prístup a spoločne komunikujú. Dokeos poskytuje pre kurzy taktiež komunikáciu prostredníctvom chatu, ktorá je rýchlejšia ako komunikácia cez fórum. LMS Dokeos podporuje aj videokonferencie.

Dostupnosť: `http://www.dokeos.com/`



Obr. 7: LMS Dokeos

3 Záver

Situácia v dostupnosti open source LMS je podľa môjho názoru veľmi dobrá. Viacero LMS poskytuje lokalizáciu do slovenského alebo českého jazyka, čo zjednodušuje spravovanie a prácu s LMS. Pri výbere konkrétneho LMS je potrené určiť kritériá, na základe ktorých sa LMS vyberie. Po viacročných skúsenostiach s výberom LMS musím konštatovať, že vo väčšine prípadov výberové kritéria vždy najlepšie spĺňal LMS Moodle. Tento článok môže byť tiež podnetom na ďalšie hlbšie bádanie v oblasti open source LMS. Zo zoznamu na stránke [3] je vidieť, že množstvo open source riešení LMS nie je až také malé a konkrétne oboznámenie sa s každým LMS by prekračovalo rozsah tohto článku.

Literatúra

[1] Systém na riadenie výučby. Dostupné [on-line] na `http://sk.wikipedia.org/wiki/Lms` [cit.:25.5.2009]

[2] Komunikácia znalostí prostredníctvom LMS a LCM. Dostupné [on-line] `http://sk.wikipedia.org/wiki/Lms` [cit.: 25.5.2009]

[3] Zoznam Open Source LMS. Dostupné [on-line] na `http://www.mc2.cz/lms-opensource` [cit.: 25.5.2009]

[4] Moodle.org: Statistic. Dostupné [on-line] na `http://moodle.org/stats` [cit.: 18.6.2009]

Kontaktná adresa
Andrej TRNKA (Ing.),
Katedra aplikovanej informatiky FPV UCM v Trnave,
Nám. J. Herdu 2, 917 01 Trnava
andrej.trnka@ucm.sk

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

VYUČOVANIE FYZIKY NA GYMNÁZIU S VYUŽITÍM VZDELÁVACEJ STRATÉGIE JUST-IN-TIME TEACHING

TULEJA, Slavomír, (SK)

1 Úvod

V tomto školskom roku sa spustila reforma vzdelávania. Na štvorročnom gymnáziu zasiahla naplno prvákov. Došlo k značnej redukcii časovej dotácie prírodovedných predmetov, medzi nimi aj fyziky. Spolu s redukciou časovej dotácie došlo aj k redukcii štátom predpísaného učiva. Toto sú však len vonkajšie znaky reformy. Podstata reformy vyučovania fyziky spočíva podľa oficiálnych materiálov [1] v transformácii študenta z pasívneho pozorovateľa činnosti učiteľa na aktívneho jedinca, ktorý sa učí porozumieť fyzike pomocou rôznych skupinových laboratórnych či teoretických aktivít. Okrem toho sa objavuje myšlienka, že omnoho podstatnejšie ako to, aby študent získal na hodinách rozsiahle fyzikálne poznatky je to, aby nadobudol tzv. kľúčové kompetencie [2]. Medzi hlavné také kompetencie môžeme zaradiť:

1. Komunikačné kompetencie (čítanie s porozumením a písanie pre porozumenie, vedieť nájsť to, čo je v texte podstatné, nájsť podstatné súvislosti, vedieť niečo slovne alebo písomne vysvetliť).
2. Kompetencia zlepšovať vlastné učenie a výkon (určovať si ciele a plánovať činnosť, realizovať plán na dosiahnutie cieľov).
3. Riešiť problémy (analyzovať problém, formulovať hypotézy pre riešenie, vedieť si urobiť schému súvislostí, nájsť príklady, ktoré niečo potvrdia alebo vyvrátia, dať veci do súvislosti s inými poznatkami).

4. Informačno-technologické kompetencie (vedieť používať počítač na hľadanie, vytváranie, spracovanie a prezentáciu informácií).
5. Sociálne kompetencie (spolupráca v skupine, empatia, diskusia, tolerantnosť).
6. Robiť numerické a symbolické aplikácie (schopnosť pozorovať javy, vybrať v nich dôležité veličiny, hľadať medzi nimi závislosti, formulovať a overovať hypotézy; schopnosť zapísať výsledky experimentu pomocou textu, schém, náčrtov, obrázkov; čítanie takého zápisu)

Ako tieto myšlienky reformy realizovať v praxi? Ako naučiť žiakov rozumieť textu? Ako ich naučiť vyjadrovať sa? Náznakom riešenia tohoto problému môžu byť inovačné snahy vo vyučovaní fyziky vo svete, ktoré sú založené na využití moderných informačno-komunikačných technológií a nových metód výučby.

2 Interaktívne formy vyučovania

Veľmi zaujímavé zmeny sa dejú v poslednom desaťročí vo vyučovaní vysokoškolskej fyziky v USA. Zavádzajú sa tzv. interaktívne formy vyučovania [3].

Asi najznámejším priekopníkom týchto nových foriem je Eric Mazur z Harvard University. Dospel k názoru, že prednášanie nového učiva študentom na prednáške je zbytočné. Popísal ho ako proces, pri ktorom sa poznatky presúvajú z písomnej prípravy učiteľa cez tabuľu do zošitov študentov, bez toho, aby pritom prechádzali aj hlavou kohokoľvek z nich [4]. Došiel teda k presvedčeniu, že tradičný monológ učiteľa pred pasívnou poslucháreň je potrebné nahradiť inými formami, v ktorých sú študenti viac zapojení do vyučovacieho procesu.

Vyvinul metódu Peer Instruction [5], v ktorej študenti na každej prednáške dostávajú niekoľko uzavretých konceptuálnych (kvalitatívnych) úloh s výberom jedinej správnej odpovede a pomocou infračervených hlasovacích zariadení (clickers) na ne odpovedajú. Potom študenti navzájom o svojich prvotných odpovediach diskutujú a každý sa snaží presvedčiť svojich susedov, že práve jeho odpoveď bola správna. Hneď na to nasleduje druhé hlasovanie ohľadom správnej odpovede na tú istú otázku. Pri ňom už väčšina študentov zvykne odpovedať správne, lebo je pravdepodobnejšie, že svojich susedov skôr presvedčí o správnosti svojej odpovede ten študent, ktorého odpoveď je správna, ako ten, ktorý len hádal, alebo v jeho vysvetlení bola nejaká chyba. Takto dochádza k učeniu sa študentov od svojich rovesníkov (z toho pochádza aj názov metódy). Správne riešenie od učiteľa s krátkym vysvetlením sa dozvedia študenti až nakoniec. Potom sa pokračuje ďalšou úlohou v sérii. Podstatou metódy Peer Instruction je téza, že najviac sa naučíme vtedy, ak to, čo sa učíme, musíme niekomu vysvetliť.

Na ďalšej prestížnej univerzite, na MIT, vyvinul John Belcher ich vlastný inovatívny spôsob vyučovania úvodných kurzov fyziky s názvom TEAL (Technology Enhanced Active Learning) [6]. V rámci TEAL pracuje asi 80 študentov v miestnosti s 13 okrúhlymi stolmi, ktoré sú vybavené zosieťovanými počítačmi. Steny dookola miestnosti sú pokryté bielymi

tabuľami a projekčnými plochami. Medzi študentmi sa pohybuje inštruktor a tím jeho asistentov. Inštruktor z času na čas krátko prezentuje základné princípy a zadáva úlohy. Študenti pracujú v malých skupinkách, vykonávajú spolu s asistentmi experimenty. Miestnosť vrie. Skupinky žiakov zapisujú riešenia úloh na tabuliach.

Spočiatku mali študenti MIT možnosť vybrať si medzi klasickými prednáškami a novým spôsobom. Od jesene 2008 sa však vzhľadom na veľmi dobré výsledky študentov v rámci TEAL získané pedagogickými meraniami už realizuje len jediný a to interaktívny spôsob vzdelávania [7].

MIT je priekopníkom aj v inom smere. V rámci iniciatívy OpenCourseWare (OCW) [8] uvoľnil inštitút pre verejnosť webové stránky množstva svojich kurzov (dnes asi 1800). Hocikto si môže pozrieť videozáznamy z prednášok, prečítať učebné materiály zverejnené v rámci týchto kurzov. MIT tvrdí, že hodnota ich vzdelania spočíva v inom ako sú učebné texty a videozáznamy z prednášok. Spočíva v efektívnej spätnej väzbe, ktorú poskytujú študentom, či už online alebo aj pri bežnom vyučovaní.

Sú to moderné informačno-komunikačné technológie, ktoré umožňujú získavať od študentov spätnú väzbu ohľadom preberaného učiva a zároveň im ju poskytovať. Exemplárnym príkladom je metóda Edwina F. Taylora [9], ktorý je autorom učebných textov k špeciálnej a všeobecnej teórii relativity. Taylor vyžaduje od študentov, aby mu vždy po tom, čo si preštudujú z učebného textu učivo k preberanej téme, poslali emailom takzvané *poznámky k čítaniu*, v ktorých majú uviesť podrobne svoje ťažkosti s textom (napr. uvedú čísla riadkov textu, kde sa nachádza pasáž, ktorej nerozumeli a vysvetlia čo konkrétne im robí ťažkosti). Taylor potom každému študentovi emailom písomne odpovedá. Niektoré zaujímavé otázky následne anonymne so študentmi prediskutujú v triede. Okrem toho Taylor používa poznámky k čítaniu na neustále vylepšovanie učebných textov.

3 Just-in-Time Teaching

Aby sa mohli hore uvedené interaktívne metódy realizovať v čase prednášky, je potrebné nahradiť tradičný výklad učiva na prednáške samoštúdiom študentov pred prednáškou. Študenti dostávajú dopredu pred každou prednáškou zadané čítanie z vhodného učebného textu, ktoré sa týka učiva, o ktorom sa bude hovoriť na prednáške. Je na študentoch, aby sa s čítaním oboznámili. Tu treba podotknúť, že americké učebnice fyziky sú na samoštúdium veľmi vhodné, keďže sú známe zrozumiteľnosťou výkladu, množstvom riešených úloh a názorných ilustrácií. Príkladom môže byť učebnica fyziky od Hallidaya, Resnicka a Walkera [10], ktorá bola preložená do češtiny.

Tento postup si vyžaduje zaviesť mechanizmy, ktoré zabezpečia, aby si študenti text naozaj prečítali. Jeden takýto mechanizmus predstavuje vzdelávacia stratégia Just-in-Time Teaching (JiTT) [11].

Metódu JiTT, vyvinuli Gregor Novak a Andy Garvin z Indiana University-Purdue University Indianapolis (IUPUI) a Evelyn Pattersonová z U. S. Airforce Academy. Táto skupina

spolupracovala s Wolfgangom Christianom z Davidson College, s ktorým vytvorili simulácie (physlets [12]), ktoré možno použiť cez internet. Metóda v sebe spája modifikované prednášky, skupinové riešenie úloh a webovú technológiu. Webová technológia sa využíva na vytvorenie spätnej väzby medzi študentmi a učiteľom. Táto spätná väzba sa vytvára tak, že študenti ešte pred stretnutím sa v triede majú vypracovať tzv. zahrievacie zadanie. Proces pozostáva z nasledujúcich častí:

1. Pred každou prednáškou sa študentom na webe zadajú špecifické, starostlivo vybrané zahrievacie zadania. Úlohy v zadaniach sa týkajú témy, ktorá ešte nebola preberaná v triede a ktorej sa bude trieda venovať na najbližšej prednáške, pri diskusiách a pri ďalších aktivitách.
2. Očakáva sa, že ešte pred stretnutím sa v triede si študenti prečítajú učebný text, potom pouvažujú a napíšu najlepšie možné odpovede na zahrievacie zadanie, aké len dokážu podať. Známkovaní sú za snahu, nie za správnosť odpovedí. Termín odovzdania odpovedí je niekoľko hodín pred samotným stretnutím sa v triede.
3. Učiteľ si pred stretnutím v triede prezrie odpovede žiakov, odhadne početnosť jednotlivých žiackych odpovedí (správnych aj nesprávnych), a niektoré z nich vyberie a dá na fólie alebo ich pripraví elektronicky, aby ich mohol použiť priamo v triede, či už pri diskusii alebo pri iných aktivitách.
4. Stretnutie v triede je koncipované tak, že je vybudované na zahrievacích zadaniach a odpovediach študentov.
5. Na konci preberanej témy sa študentom zadá záludná otázka, hádanka, ktorá sa umiestni znovu na web.

Metóda JiTT nie je špecifická len pre fyziku. Je využívaná aj v iných predmetoch ako sú matematika, geológia, informatika, chémia, biológia atď [13]. Metóda sa pritom nemusí používať len na vysokých školách, ale jej obmenená podoba sa používa aj na stredných školách [14].

4 Implementácia JiTT do vyučovania stredoškolskej fyziky s využitím Moodle

V školskom roku 2008/2009 sme sa rozhodli vyskúšať *prispôsobenú* verziu metódy JiTT v predmete fyzika na 56 našich prvákoch z troch rôznych tried. K tomu sme potrebovali vhodný systém na správu online vzdelávacích kurzov (Course Management System – CMS). Voľba padla na Moodle, čo je dosť známy a rozšírený open source CMS.

Vyučovanie prváckej fyziky prebiehalo na jednej dvojhodinovke raz do týždňa. Každá prvácka trieda sa delila na dve skupiny, ktoré učili dvaja rôzni učitelia. V každej skupine bolo okolo 16 žiakov. Predmetom štúdia boli tematické celky *Fyzikálne veličiny a ich meranie*, *Sila a jej pôsobenie na teleso* a *Pohyb a sila*. Používali sme texty z inovovanej učebnice

fyziky pre 1. ročník gymnázia [15, 16]. Texty sme si upravovali pre svoju potrebu a na Moodle sme dávali ich elektronickú verziu vo formáte PDF.

Inovovaný spôsob vyučovania fyziky sa od starého spôsobu líšil v tom, že obsahoval asi dvakrát väčší počet laboratórnych cvičení ako pri vyučovaní fyziky po starom a vyžadoval časté používanie počítačov na týchto cvičeniach. V triede sme mali k dispozícii 4 laptopy pre žiakov a jeden laptop s dataprojektorom pre učiteľa. Na laptopoch bol nainštalovaný operačný systém GNU/Linux, konkrétne Ubuntu 8.10.

Laptopy boli používané hlavne na videoanalýzu v open source programe Tracker a na spracovanie výsledkov experimentov v programe OpenOffice.org Calc. Taktiež sme niektoré laboratórne aktivity (napr. skúmanie šikmého vrhu) vykonávali na počítačových modeloch fyzikálnych javov naprogramovaných v jazyku Java a dostupných voľne na internete [17].

Stratégiu JiTT sme na našom gymnáziu používali v prispôsobenej podobe. Spomenieme niekoľko hlavných odlišností oproti štandardnej verzii JiTT:

- Žiaci si dopredu pred dvojhodinovkou mali prečítať len časť učebného textu. Zvyšok textu čítali až po výklade učiteľom.
- Naše zahrievacie zadania pozostávali vždy len z jednej typovej úlohy k preberanému učivu, ktorá niekedy pozostávala z viacerých častí. Len výnimočne sme zadávali kvalitatívne úlohy.
- Naša implementácia JiTT využívala diskusné fóra na Moodle. Žiaci sa povinne museli pýtať otázky k čítaniu a dávať si otázky na skúšanie, na ktoré si potom navzájom odpovedali.
- Zaviedli sme kvízy [18] ako formu boja proti plagiátorstvu. Online zadania totiž žiaci často odpisovali od spolužiakov.

5 Porovnanie vyučovania fyziky starým spôsobom s vyučovaním fyziky metódou JiTT pomocou Moodle

V tomto odseku je v Tab. 1 uvedený popis našej súčasnej implementácie JiTT na Moodle a jej porovnanie s klasickou výučbou. Počas tohto školského roka sme systém tri krát korigovali, keďže nie všetko spočiatku fungovalo ako sme si predstavovali.

Tab. 1: Porovnanie vyučovania fyziky po starom a vyučovania fyziky metódou JiTT pomocou Moodle

Fyzika starým spôsobom	**Fyzika metódou JiTT pomocou Moodle**
Počet žiakov v triede okolo 32.	**Počet žiakov v triede** 16.

Pokračovanie na ďalšej strane...

Tab. 1 – *Pokračovanie*

Fyzika starým spôsobom	Fyzika metódou JiTT pomocou Moodle
Hodinová dotácia 2 hodiny teórie do týždňa. 2 hodiny cvičení raz za dva týždne.	**Hodinová dotácia** Jedna dvojhodinovka do týždňa. Nerozlišujú sa hodiny cvičení a teórie.
Oboznamovanie sa s novým učivom Učiteľ vysvetľuje všetko nové učivo na hodine. Žiak si číta učebný text k učivu doma po výklade.	**Oboznamovanie sa s novým učivom** Časť učiva si žiaci študujú z učebného textu vo forme PDF čítaní na Moodle dopredu sami doma. Učiteľ sa potom na hodine sústreďuje len na problematické časti učiva (o tom, ktoré to sú vie na základe spätnej väzby od študentov popísanej nižšie). Zapája pritom žiakov do diskusie. Zvyšnú časť učiva vysvetľuje učiteľ na hodine a žiaci si zodpovedajúci učebný text čítajú po hodine.
Poznámky k učivu žiakovi do zošita buď diktuje učiteľ na hodine, alebo si ich má urobiť doma sám.	**Poznámky k učivu** si žiaci robia doma sami na margo vytlačených PDF čítaní alebo do zošita. Učiteľ poznámky nediktuje.
Ústne skúšanie na hodine hodnotené známkou. Žiak hovorí plynule o teórii a rieši úlohu.	**Ústne skúšanie na hodine** hodnotené bodovo. Žiak hovorí plynule o teórii alebo rieši úlohu (zo zahrievacieho zadania, z teoretického cvičenia alebo podobnú).
Laboratórne cvičenia *Výstup:* protokol (laboratórny záznam) v zošite alebo na papieri, ktorý robí každý žiak sám. *Hodnotenie:* známka za každý protokol, alebo len jedna známka za všetky protokoly.	**Laboratórne cvičenia** *Výstup:* protokol (laboratórny záznam) vo forme wiki stránky na Moodle, ktorý robí spoločne štvorčlenná skupina. V každej skupine je určený pisár zodpovedný za protokol. Ostatní členovia sú povinní pridať po 3 návrhy na doplnenia alebo na zmenu. Pisár odovzdá pomocou modulu zadanie v Moodle vyplnený hodnotiaci dotazník, kde hodnotí prácu členov skupiny na hodine aj na vypracovávaní wiki. Funkcia pisára rotuje. *Hodnotenie:* Na základe kvality protokolu a na základe hodnotiaceho dotazníka od pisára. Každý žiak skupiny dostáva vo všeobecnosti rôzne bodové hodnotenie.

Pokračovanie na ďalšej strane...

Tab. 1 – *Pokračovanie*

Fyzika starým spôsobom	**Fyzika metódou JiTT pomocou Moodle**
Teoretické cvičenia *Forma:* spoločné riešenie úloh pri tabuli. *Výstup:* vyriešené úlohy v zošite každého žiaka. *Hodnotenie:* buď žiadne, alebo každý žiak, čo rieši úlohu pri tabuli dostáva známku.	**Teoretické cvičenia** *Forma:* skupinové riešenie série úloh pomocou návodu na papieri. *Výstup:* vyriešené úlohy jednak v zošite každého žiaka ale aj na skupinovom wiki na Moodle, kde sú riešenia úloh aj s textovým komentárom. Je určený manažér skupiny zodpovedný za riadenie vypracovania riešení série úloh. Pridelí ostatným (aj sebe) úlohy na riešenie a vypracovanie na wiki. Ostatní sú povinní vypracovať riešenie im pridelených úloh a skontrolovať správnosť riešení zvyšných členov skupiny. Manažér potom odovzdá pomocou modulu zadanie v Moodle vyplnený hodnotiaci dotazník, kde hodnotí prácu členov skupiny na hodine aj na vypracovávaní wiki. Funkcia manažéra nerotuje. Manažéra určuje učiteľ. *Hodnotenie:* Na základe kvality riešení na wiki a na základe hodnotiaceho dotazníka od manažéra skupiny. Každý žiak skupiny dostáva vo všeobecnosti rôzne bodové hodnotenie.
Písomky hodnotené známkou.	**Písomky** hodnotené bodovo. Možnosť opravnej písomky. Výsledné hodnotenie je priemerom hodnotení v pôvodnej a opravnej písomke.

Pokračovanie na ďalšej strane...

Tab. 1 – *Pokračovanie*

Fyzika starým spôsobom	**Fyzika metódou JiTT pomocou Moodle**
Otázky žiakov Na otázky (sporadické) žiakov na hodine odpovedá priamo na hodine učiteľ. *Výstup:* žiadny *Hodnotenie:* žiadne	**Otázky žiakov** Každý žiak musí do online skupinového diskusného fóra typu Q&A každý týždeň dať jednu otázku „na skúšanie“ z novej látky, ktorá bola zadaná ako čítanie a otázku ohľadom čohokoľvek, čomu v čítaní alebo na hodine nerozumel. Ak rozumel všetkému, uvedie to písomne do fóra. Každý žiak okrem toho povinne odpovedá na otázky „na skúšanie“ všetkých spolužiakov vo svojej štvorčlennej skupine. Voliteľne odpovedá na otázky ohľadom textu čítania a za to môže získať bonusové body. *Výstup:* kedykoľvek prístupné diskusné fóra s otázkami a odpoveďami *Hodnotenie:* dvojzložkové: Jedno bodové hodnotenie za položenie otázok. Ďalšie bodové hodnotenie za odpovede. Hodnotí sa kvalita odpovedí. Žiak môže získať bonusové body za pomoc spolužiakom s chápaním čítania.
Domáce úlohy dostávajú žiaci po výklade učiva na hodine. Doma ich riešia a sporadicky im ich učiteľ skontroluje a dá im spätnú väzbu ohľadom ich správnosti. Na nasledujúcej hodine učiteľ pri ústnom skúšaní zadáva úlohy podobné ako boli na domácu úlohu.	**Zahrievacie zadania** dostávajú žiaci spolu so zadaným čítaním pred hodinou. Po prečítaní čítania riešia zahrievacie zadanie. Riešenie odovzdávajú v elektronickej forme na Moodle. Spätnú väzbu dostane od učiteľa každý žiak priamo v texte svojho riešenia. Učiteľ hodnotí snahu, nie správnosť. Taktiež je pre žiakov zverejnené vzorové riešenie. Na najbližšej hodine môže byť riešenie zahrievacieho zadania predmetom skúšania.
Kontakt žiakov navzájom Rozhovory medzi žiakmi cez prestávky ohľadom učiva, domácich úloh, atď.	**Kontakt žiakov navzájom** Rozhovory medzi žiakmi cez prestávky ohľadom učiva, zahrievacích zadaní, atď. ale aj skupinová četovacia miestnosť na Moodle, kde sa môžu skupiny žiakov z domu dohodnúť na vzájomných konzultáciách ohľadom písania protokolov, najnovšieho čítania, riešenia teoretických úloh a pod.
—	**Cvičný test na Moodle** s okamžitou spätnou väzbou pred písomkou.

Pokračovanie na ďalšej strane...

Tab. 1 – *Pokračovanie*

Fyzika starým spôsobom	Fyzika metódou JiTT pomocou Moodle
Päťminútovka Nepoužívala sa.	**Kvíz k čítaniu** na začiatku každej JiTT hodiny. Žiaci dostanú tri otázky týkajúce sa posledného čítania a jednu otázku týkajúcu sa opakovania. Sú to otázky s voľbou jedinej správnej odpovede. Spolu môžu získať 4 body. Za každú správnu odpoveď 1 bod. Na podporu skupinového ducha získajú bonusový, piaty bod tí žiaci, ktorých skupina dosiahla priemernú úspešnosť aspoň 3 body.
Plagiátorstvo Prejavuje sa odpisovaním domácich úloh a protokolov. V podstate sa nijako neriešilo.	**Plagiátorstvo** Prejavuje sa odpisovaním riešení zahrievacích zadaní, otázok do diskusie a odpovedí na ne, protokolov a riešení teoretických úloh na wiki. Rieši sa nasledovne: Je dovolená a podporovaná spolupráca v rámci skupiny (okrem otázok do diskusie a odpovedí na ne), ale žiak musí uviesť, s kým zo skupiny konkrétne spolupracoval. Je zakázaná spolupráca medzi skupinami navzájom. Pri každom zistení odpisovania sa tomu kto odpísal a tomu kto dal odpísať, odčíta z celkového hodnotenia kurzu 5 trestných percent.
Celková známka za predmet určená na základe subjektívneho rozhodnutia učiteľa.	**Celková známka za predmet** určená na základe objektívneho výpočtu ako vážený aritmetický priemer z jednotlivých hodnotení v kurze.

6 Skúsenosti s používaním JiTT

Opísaný spôsob realizácie JiTT na našom gymnáziu je výsledkom troch postupných korekcií, ktoré sme urobili v priebehu školského roka.

- **Korekcia čítaní**
  Na začiatku školského roka sme čítania študentom pripravovali v module Prednáška. Je to modul Moodle, ktorý umožňuje vytvoriť učebný text rozdelený na malé dávky, pričom za každou dávkou dostáva študent automaticky vyhodnocovanú otázku. Ak na ňu odpovie správne, prejde k nasledujúcej dávke učiva. Ak nie, dostane pomocnú informáciu a je vyzvaný odpovedať znova. Moodle zaznamenáva úspešnosť študenta pri odpovedaní na tieto kontrolné otátky a udeľuje mu za čítanie známku.

Naši žiaci veľmi rýchlo prišli na to, ako sa spojiť cez čet a povedať si, čo sú správne odpovede na kontrolné otázky. Snažili sa maximalizovať svoju známku za čítanie a minimalizovať vynaloženú snahu. Často nemali ani poňatia o čom bolo čítanie.

Na základe týchto negatívnych skúseností sme od polroka začali čítania dávať študentom ako pdf súbory s očíslovanými riadkami a širokým okrajom, na ktorý si mohli dopisovať poznámky. Redukovali sme množstvo čítaní na samoštúdium na polovicu. Čítania sme rozdelili na čítania pred hodinou a čítania po hodine.

Zároveň sme od žiakov začali vyžadovať účasť v diskusii o prečítanej látke. Vyžadovali sme od nich, aby sa k látke pýtali na veci, ktorým neporozumeli a následne sme im na ich otázky vo fóre odpovedali. Toto však bolo príliš časovo náročné zo strany učiteľa a tak sme od písomného odpovedania na otázky po niekoľkých týždňoch upustili.

- **Korekcia písania online testov**
  Neosvedčilo sa nám ani písanie online testov. Takýto test sme žiakom zadali na začiatku školského roka. Dali sme mu malú váhu a výsledok testu sme zarátali do známky. Žiaľ, znovu došlo k odpisovaniu medzi žiakmi. Poučili sme sa a v budúcnosti plánujeme online testy neznámkovať. Ak ich budeme študentom dávať, tak len ako cvičné testy.

- **Korekcia písania protokolov na wiki**
  Postupne sme prispôsobili aj písanie protokolov na wiki. Pôvodne mal protokol napísať pisár a ostatní ho mali skontrolovať a napísať svoje hodnotenie. Viedlo to k tomu, že zvyšní členovia skupiny písali prehnane pozitívne hodnotenia na protokol a vôbec pisárovi pri písaní nepomáhali. Niektorí napísaný protokol ani len neprečítali. Preto sme systém zmenili a v dnešnej podobe sa každý člen skupiny musí zúčastniť písania protokolu. Pisár, ktorý je zodpovedný za vypracovanie protokolu píše na ostatných členov hodnotenie a to často nie je pozitívne. Takto aspoň adekvátne ohodnotíme tých žiakov, ktorí na protokole skoro vôbec nepracovali.

- **Korekcia zadávania zahrievacích zadaní**
  Pôvodne sme zahrievacie zadania zadávali študentom cez Feedback modul. Pôvodné zahrievacie zadania pozostávali z viacerých úloh, z ktorých vždy nejaká úloha bola kvalitatívna. Žiaľ kvôli negatívnemu ohlasu žiakov, ktorí sa cítili byť preťažení, sme výrazne zmenšili rozsah úloh zahrievacích zadaní a aj ich náročnosť. Od polroka sme začali používať modul Zadanie. Tento modul nám umožňuje vpísať žiakom do ich riešení komentár.

V Tab.2 uvádzame priradenie modulov Moodle jednotlivým komponentom nášho JiTT kurzu fyziky.

Tab. 2: Priradenie modulov Moodle jednotlivým komponentom JiTT kurzu

Komponent JiTT	**Modul v Moodle**
Čítanie	Odkaz na PDF súbor (pôvodne prednáškový modul).
Diskusie	Diskusné fórum typu Q & A.
Protokoly	Wiki modul.
Zahrievacie zadania	Modul Zadanie (pôvodne Feedback modul)
Kontakt žiakov mimo triedu	Modul pre čet.
Skúšobné testy	Testový modul.

7 Záver

Po roku používania JiTT pri vyučovaní fyziky v prvom ročníku gymnázia môžem povedať, že používanie JiTT má svoje výhody aj nevýhody.

Výhody:

- Žiaci sú nútení pravidelne sa pripravovať na vyučovanie. Nútia ich k tomu povinné online diskusie, zahrievacie zadania a kvízy na hodine.
- Tento systém vyučovania núti žiakov klásť otázky k preberanej látke a tak „rozbíja“ neochotu žiakov klásť otázky zakorenenú v klasickom spôsobe výučby.
- Učiteľ získava vynikajúcu spätnú väzbu ohľadom toho čo v čítaní a čo v riešení zahrievacieho zadania robilo žiakom ťažkosti. Na základe tejto spätnej väzby potom prispôsobuje priebeh hodiny.
- Žiaci na hodinách vo väčšine prípadov rozumejú tomu, o čom sa práve diskutuje a aktívne sa zapájajú do diskusií.
- Čítania sú žiakom zadávané v elektronickej podobe. Učiteľ si ich môže upraviť a na základe otázok žiakov, ktoré ku každému čítaniu dostane ich postupom rokov vylepšovať. Môže si vytvárať databázu často kladených otázok.
- Vyučovanie na hodine je improvizované. Tá istá hodina odučená v dvoch rôznych triedach môže vyzerať úplne odlišne, keďže spätná väzba od žiakov získaná na Moodle môže byť výrazne odlišná.
- Budujú sa mnohé z kľúčových kompetencií. Práca s textom, tímová práca, zapisovanie laboratórnych protokolov, schopnosť diskutovať a kritizovať na hodine ale aj online v písomnej podobe.

Nevýhody:

- Veľké pracovné zaťaženie pre učiteľa, veľa času stráveného za počítačom. Treba rátať asi tak so štyrmi hodinami prípravy na jednu dvojhodinovku.
- Žiaci si musia spočiatku na nový systém zvyknúť. Je potrebné prekonať ich počiatočný odpor.
- Učebné texty, ktoré máme k dispozícii, nie sú najvhodnejšie na samoštúdium. Je v nich málo riešených úloh. Často sú príliš zložito napísané.
- Neustály boj s plagiátorstvom. Nie je nič jednoduchšie ako odpísať elektronický text, mierne ho upraviť a dať ho na Moodle. Preto sme zaviedli kvízy na hodinách.

Spôsob výučby, ktorý sme zaviedli je pre nás istým krokom do budúcnosti[1]. Vidíme v ňom veľký potenciál na postupné vytvorenie dobrých učebných materiálov pre študentov. Fascinuje nás to, že nám nový systém umožňuje nahliadnuť do myslí študentov a poodhaliť ich miskoncepcie. Tak máme šancu ich prekonávať.

Zároveň cítime, že nový systém je dosť časovo náročný. Preto jeho zavádzanie musí byť postupné. Nemôžeme si dovoliť takto učiť všetky triedy v ročníku, lebo dochádza k prílišnému preťaženiu vyučujúcich, hlavne vo fáze pilotáže JiTT, keď sa priebežne musia chystať všetky učebné materiály. Veríme, že sa nám v najbližších dvoch rokoch podarí vytvoriť vhodné materiály pre druhý a tretí ročník a priebežne ich pilotne otestovať aj na žiakoch. Tiež veríme, že dovtedy dôjde k postupnému vylepšovaniu materiálov, ktoré budeme učiť už po druhý a tretí krát v druhom a prvom ročníku.

Literatúra

[1] *Štátny vzdelávací program - ISCED 0, 1, 2, 3A*. [online], [citované 8. 5. 2009]. Dostupné na `http://www.minedu.sk/index.php?lang=sk&rootId=2319`

[2] TUREK, I. *Didaktika.* Bratislava: Iura Edition, 2008, s. 199–218

[3] REDISH, E. F.: *Teaching Physics with the Physics Suite*. Hoboken, NJ: John Wiley & Sons, 2003

[4] MAZUR, E. *Farewell, Lecture?* In: Science, 2. január 2009, Vol. 323, s. 50–51

[5] MAZUR, E.: *Peer Instruction: A User Manual.* Upper Saddle River, NJ: Prentice Hall, 1997

[6] *TEAL: Technology Enhanced Active Learning*. [online], [citované 8. 9. 2009]. Dostupné na internete `http://web.mit.edu/edtech/casestudies/teal.html`

[1] *Krok do budúcnosti* je zároveň názov projektu, ktorý realizujeme na našom gymnáziu a ktorý je financovaný Agentúrou Ministerstva školstva SR pre štrukturálne fondy EÚ v rámci operačného programu Vzdelávanie. Súčasťou projektu je testovanie stratégie JiTT na strednej škole.

[7] RIMER, S.: *At M.I.T., Large Lectures Are Going the Way of the Blackboard.* [online]. Publikované 13. 1. 2009. [Citované 8. 5. 2009]. Dostupné z `http://www.nytimes.com/2009/01/13/us/13physics.html`

[8] *MITOPENCOURSEWARE.* [online], [citované 8. 5. 2009]. Dostupné na internete `http://ocw.mit.edu/OcwWeb/web/home/home/index.htm`

[9] TAYLOR, E.F.: *Only the student knows*. In: Am. J. Phys., marec 1992, Vol. 60

[10] HALLIDAY, D. – RESNICK, R. – WALKER, J.: *Fyzika*. Praha: VUTIUM a PROMETHEUS, 2000

[11] NOVAK, G.M. – PATTERSON, E.T. – GAVRIN, A.D. – CHRISTIAN, W.: *Just-in-Time Teaching: Blending Active Learning with Web Technology.* Upper Saddle River, NJ: Prentice Hall, 1999

[12] CHRISTIAN, W. – BELLONI, M.: *Physlet Physics: Interactive Illustrations, Explorations, and Problems for Introductory Physics.* Upper Saddle River, NJ: Pearson Education, 2004

[13] *What is Just-in-Time Teaching.* [online], [citované 8. 5. 2009]. Dostupné na internete: `http://serc.carleton.edu/introgeo/justintime/what.html`

[14] NOVAK, G.: *JiTT Impact and Citations.* [online], [citované 14. 5. 2009]. Dostupné na internete: `http://jittdl.physics.iupui.edu/jitt/impact.html`

[15] KOUBEK, V. – ŠABO, I.: *Fyzika pre 1. ročník gymnázií.* Bratislava: Slovenské pedagogické nakladateľstvo, 2004

[16] *Študijné materiály učiteľského štúdia fyziky.* [online], [citované 14. 5. 2009]. Dostupné online: `http://www.ddp.fmph.uniba.sk/~koubek/UT_html/Ucebnice.htm`

[17] *PhET: Interactive simulations*. [online], [citované 8. 5. 2009]. Dostupné na internete `http://phet.colorado.edu/index.php`

[18] FLEISCHER, R.: *Just-in-Time: Better Teaching in Hong Kong.* In: Proceedings of the Second Teaching and Learning Symposium - Teaching Innovations: Continuous Learning and Improvement, HKUST, Hong Kong, May 17, 2004

Kontaktná adresa

Slavomír TULEJA (RNDr., PhD.),
Gymnázium arm. gen. L. Svobodu, Komenského 4,
066 01 Humenné,
stuleja@gmail.com

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
Žilina 2.–5. júla 2009

OPEN SOURCE SKRZE MIKROSPOLEČNOSTI

ZACHOVAL, František, (CZ)

1 Úvod

Příspěvek sleduje sociologické fenomény blízké filozofii Open source (OS), definuje události, které předcházely a i paralelně existují vedle samotného procesu volného sdílení informací v digitálním světě. Autonomní prostory uvedených komunit fungují zejména ve vlastní stimulaci vývoje. Digitální subkultury a zejména OS komunity jsou současně považovány za motor k inovacím a návrhům nových forem a funkcí. Postoje těchto skupin jsou silnou reflexí aktuálního světa a současně silnou inspirací.

Příklady občanských neposlušností, hactivismu a pirátství jsou vždy dočasné a skrze dobové právní prizma prezentovány více méně negativně. Je Open source v současně známe podobě další anomálií určité skupiny ve společnosti? K jejímu přiblížení zde rozebírám pár lokálních příkladů v prostoru od středověku přes evropské separatistické enklávy až po marketingový přístup Creative Commons.

2 Open society ve třech generacích

Svobodné ideály skrze mikrospolečnosti nám detailně zpracoval spisovatel, esejista a básník Peter Lamborn Wilson (aka Hakim Bey), autor *Dočasné autonomní zóny* (Temporary Autonomous Zone), kterou recenzoval Ondřej Slačálek ve čtrnáctideníku A2 v roce 2005,[1] „Dočasnou autonomní zónou jsou různé pokusy vymanit se z područí autorit a vystoupit ze stereotypů, které nám vnucuje oficiální kultura. Jedná se zároveň často o pokus osvobozovat

[1] `http://www.advojka.cz/archiv/2005/8/od-revoluce-k-potulce` A2, 08/2005, ISSN 1803-6635

určitý prostor, vystoupit z mapy světa, již sestavila moc a která nikdy nemůže pokrýt celou skutečnost v měřítku jedna ku jedné, a na níž tedy stále existují místa, ze kterých lze alespoň načas vzdorovat tlakům moci.“

Pokud-li budeme skrze uvedenou šablonu nahlížet na dočasně neovládané územní celky ve střední Evropě, tak významnou genezí nám jsou kolonizační procesy. V českých poměrech tak stojí za zmínku události, které svou periferností nebyli doposud důsledně zpracovány, ale vytvořili ve 14. století významnou lokální anomálií na poli mocenských zájmů českých králů, vratislavských biskupů a slezských Piastovců.[2]

V pomyslném trojúhelníku mezi masivem Králického Sněžníku, Hrubým Jeseníkem a Rychlebskými horami po dobu dvou generací ovládal rod Wüstehubů oblast, která přesahovala rozlohu současně standardizovaných okresů. Moravskoslezská historie je sestavena ze skutečných střípků, kde není přesně známo, odkud zmíněný rod vzešel a kde vznikl impulz pro jejich činy.

Každopádně Wüstehubové vzbuzují jisté rozpaky, neboť v jejich aktivitě se od počátku prolínaly dva obtížně slučitelné principy: drsné vystupování, přerůstající v brutální drobné války a na druhou stranu je jim připisována záslužná tvorba kulturní krajiny v zalesněných oblastech. Jejich ekonomicky nezávislé dominium mělo úzkou vazbu na tehdejší exploataci drahých kovů, které jim zajistilo skutečnou nezávislost.[3]

Pod jejich vládou vznikla důmyslná fortifikační síť hradů Kaltenštejn, Kolštejn, Pleče, Frýdberk a hrad Rychleby, které na obou stranách zemské hranice blokovaly tehdy jediné průchozí obchodní stezky. Ve své největší expanzi jsou nepřímo spojovány se založením nejméně třinácti vsí v dnešní Hanušovické vrchovině. Dlouhodobá důlní činnost nestačila financovat nákladná ochranná sídla a výrazné protěžování hospodaření podřízených vesnic nebylo možné. Wüstehubové zmizeli. Hrady se již neobývaly a započaté osídlování území i po ukončení důlní těžby v těžkých horských podmínkách se po dobu 700 let nadále rozrůstalo[4]. Tehdejší stav nastínil historik Martin Wihoda takto: „V jistém smyslu lze wüstehubská sídla označit za nepřirozený produkt soudobých mocenských poměrů, za hrady bez skutečné vazby na osídlenou krajinu.“[5] Pochopitelně těchto případů je ve středověké historii mnoho, ale uvedený příklad se vymyká svou skutečností, že Wüstehubové svou nezávislostí a mohutnou stavební plodností rozvinuli danou oblast dle svých potřeb. Zvláštnost našeho příkladu je v tom, že refundované přírodní bohatství bylo efektivně investováno zpět do čerstvě kolonizované oblasti a do bezpečnosti samotných lidí.[6]

[2]Archaeologia historica 27/2002,Wihoda, Martin-Kouřil, Pavel: Česká kastelologie na rozcestí? AH 27, 2002, str. 22, ISSN 0231-5823, ISBN 80-7275-031-3

[3]vCasopis Slezského zemského muzea, Wüstehubové / Ďáblovo plémě nebo tvůrcové kulturní krajiny? str. 214, Martin Wihoda, Pavel Kouřil, ISSN 0323-0627

[4]Rapidní úpadek sídel nastal až po reemigračních procesech v letech 1946–1947, kde nové obyvatelstvo společně s komunistickými reformami zemědělství nedokázalo navázat na specifika hospodaření v horské krajině, `http://edata-base.net/`

[5]Časopis Slezského zemského muzea, Wüstehubové / Ďáblovo plémě nebo tvůrcové kulturní krajiny? str. 213, Martin Wihoda, Pavel Kouřil, ISSN 0323-0627

[6]Dochovala se jediná výjimka, kde vytěžené prostředky putovali mimo území rodu. Jednalo se pravděpodobně

Paralelně, tisíc kilometrů západně začali čistě diplomatickou cestou vznikat městské samosprávy, které byly nezávislé na Svaté říši římské. Bohatí občané si od půlky 13. století v městech jako Mainz, Regensburg, Strasbourg, Cologne a Speyer vymohli samosprávu. Započali tak zlatou éru vlastních svobodných měst (Freistadt) vůči císařovi. Jako u dynamického vývoje na moravskoslezském pomezí i zde proběhla zvláštní investice do vlastních obyvatel. Významným krokem byl vývoj v sekularizaci společnosti a výrazná tolerance v otázkách náboženských svobod.[7] U těchto středověkých republik a pak zejména u bohatých jako byl Lübeck a Augsburg se jednalo o širší nezávislost, jelikož důsledně kontrolovali své obchodní aktivity a připouštěli pouze malé zásahy do vlastní politické struktury. Ekonomická síla Augsburgu vedla v roce 1540 k založení historicky první burzy v Německu.[8] Absolutní svobody si kontinuálně tyto celky neužívaly, ale za svou dobu existence se jednalo o výkyvy v řádech generací, tedy většinou osob, které byly schopny danou politickou situaci v politickém klimatu diplomaticky zvládnout.[9] Středověké svobodná města jsou přímým opakem Wilsonových teorií dočasnosti. Jejich dlouhodobé trvání bylo otázkou pouze okolních států, které je konstituovaly, tak jak tomu bylo při vzniku Svobodného města Danzig v roce 1920, díky výsledkům Versailleské mírové smlouvy.[10] Pravým opakem je Svobodný stát Fiume, které vzniklo deklarační teorií tvorby státu,[11] tedy metody již příznačné 20. století, kdy sám na sebe vyhlásil samostatnost a to bez ohledu na uznání okolními státy. Vůdci Gabrielu D'Annuziu svobodný stát vydržel pouhé čtyři roky.

Skutečným prototypem Wilsonových myšlenek se všemi insigniemi je současná enkláva „Podněsterská moldavská republika" (Pridnestrovian Moldavian Republic, PMR), která vyhlásila samostatnost na Moldavské republice hned po rozpadu Sovětského svazu v roce 1990. Je neuvěřitelné jak podobnou genetiku má tento mladý stát, jak tomu bylo u karibských pirátů, a že si někdo může v současně totálně globalizované Evropě vyhlásit nezávislost a to i bez předpokladu výrazné politické nebo ekonomické účasti. PMR je de iure součástí Moldavska, ale útvar je 19 let samostatný. V důsledku národnostního rozložení ústava mladého státu deklaruje unikátní tříjazyčnost.[12] Drtivá většina internetových stránek a oficiální zastoupení

o diplomatické donátorství cisterciáků v nedalekém Kamenci.

[7] `http://de.wikipedia.org/wiki/Augsburger_Reichs-_und_Religionsfrieden`, Augsburské vyznání (25. června 1530), vyjádření evangelických stavů k teologickým otázkám, které se lišily od katolických pozic. `http://de.wikipedia.org/wiki/Confessio_Augustana`, Augšpurský mír (25. září 1555) ukončil první období náboženských válek v Německu.

[8] Pouze devět let po první burze na světě v Antverpách byla založena Augsburská burza, `http://de.wikipedia.org/wiki/Augsburger_Börse`

[9] `http://de.wikipedia.org/wiki/Freie_Stadt`

[10] Versailleská smlouva konstituovala dle mezinárodního práva tvorbu nového státu, kde obyvatelstvo volalo po právu na sebeurčení, které jim bylo odepřeno. Danzig 2. září 1939 po obsazení Polska zanikl, když vyhlásil připojení k Velkoněmecké říši.

[11] Mnoho takto vzniklých útvarů nebylo okolními státy respektováno, ale Fiume bylo okamžitě uznáno Spojenými státy americkými, Francií a Velkou Británií.

[12] Podněstří na začátku 20. století bylo součásti multikulturního knížectví Bessarabia. Expanze Sovětského svazu ve 20. letech se zastavila na řece Dněstr, kde vznikla Moldavská autonomní sovětská socialistická republika. Po připojení Besarábie k Sovětskému svazu (1940) Podněstří vytvořilo spolu s Moldavskou SSR, už

Moldavské republiky v České republice nedoporučují vstup na jimi nekontrolované území. Obyvatelé Podněstří jako středověcí Wüstehubové vedli krvavou válku za samostatnost. Separatisté profitují ze získaných průmyslových zón, které tvořily 40 % hrubého domácího produktu celé bývalé Moldavské republiky a stejně zpětně investují do své ochrany v nárazové zóně. Samotný vznik Podněstří má národnostní pozadí, kde již v roce 1989 proběhly bouřlivé stávky v Tiraspolu vedené současným prezidentem Igorem Smirnovem, který nesouhlasil s čerstvě prohlášeným moldavským úředním jazykem.[13]

Stát, který formálně na mapě světa neexistuje, je vlastně ve svém měřítku suverénní jednotkou. Současný prezident Igor Smirnov skrze deklaratní teorií, prohlásil PMR nositele myšlenek komunistické Moldavské SSR.[14] Původní zájmy lidu tak pirátskou metodou v 20. století nepokrytě přetransformoval do odvážné stavby mladého státu. Současní občané participující společně na tomto projektu jsou nadmíru spokojeni, jako vysokoškolský student z místní tiraspolské univerzity Vladimír Ševčenko: „Necítím se nějak omezen, byl jsem před rokem v Německu na stáži, jsme svobodní."

Jak karibští piráti byli existenčně závislí na vznikajících námořních cestách, tak i PMR je nepřímo kritizována za soudobý neetický prohřešek; ilegální obchod se zbraněmi. Rovina celospolečensky odsuzovaného podnikání se do budoucna negativně projeví, kdy v propleteném obchodním světě stojí transakce na důvěře, což PMR nemá.

3 Open source jako produkt kreativní destrukce?

Jaká je přímá souvislost mezi reálnými sociologickými projekty o nezávislost a tvůrci open source software? Je zde mnoho otázek, které jsou možná zainteresované komunitě zřetelné, ale pro širší společnost je nutná komplexnější interpretace. Bezesporu je jasné, že termín Open source je soustavně kriticky zkoumán (princip filozofie), ale tradiční výklad věcí není na vyvíjející se fenomén lehce aplikovatelný. Výše uvedené kontexty jsou pouze naznačenou paralelou, kde je nutné hledat pravděpodobnost pro samotný proces vývoje OS. Proč? V krátkodobé historii se uskutečnil výrazný vývoj myšlenek „sdílení díla". Výrazně se redefinoval pohled na nezávislost otevřených formátů dat. Mnoho uživatelů pociťuje nutnost vysvětlit problematiku z čistě fenomenologického přístupu.

Podíváme-li se na naše rozebírání „otevřenosti" skrze *Kapitál* Karla Marxe, tak je zde silně patrná analogie Open source s termínem „nadhodnota" - naší společnosti. Před 150 lety objevená definice byla vztahována k rané průmyslové výrobě jako negativní jev kapitalismu. Je současná „nadhodnota" produktem vyspělosti naší společnosti? Nebo je jen negativním produktem současného ekonomického modelu?

nezávislou na Ukrajinské SSR, `http://cs.wikipedia.org/wiki/Podněstří`

[13]Část území za řekou Dněstr (Podněstří) Moldavské sovětské socialistické republiky bylo různými vlády ve 20. století reemigrováno ruskými a ukrajinskými obyvateli. Aktuálně žije v Podněstří 177 156 Moldavců (31,9 %), 168 270 Rusů (30,3 %) a 159 940 Ukrajinců (28,8 %), `http://en.wikipedia.org/wiki/Transnistria`

[14]V porevoluční době (1992) bylo PMR podpořeno armádou Ruské federace.

Z antropologického pohledu má OS společenstvo plno pozitivních rysů jako morálka, etika a svobodná vůle. To co lidé okolo OS mají navíc jak uvádí Francise Wheen ve svém rozboru Marxův *Kapitál* je pojem „volný čas“. Volný čas je příznačná hodnota pro 21. stol., která začíná být mnohými tvůrci využívána pro podporu produktů a které nepřímo ovlivňují nezávislost společnosti na všudepřítomné komercializaci.

Současná situace již byla v hrubých rysech popsána ekonomem Josephem Schumpeterem,[15] který rovinu změn nazývá kreativní destrukce (Creative destruction). Ve svém díle *Kapitalismus, socialismus a demokracie* (Capitalism, Socialism and Democracy) je termín používán k popisu procesu transformace, který doprovází radikální inovaci. Inovačním vzestupem je zde dlouhodobý hospodářský růst, který zničí hodnotu zavedených společností a které využívaly určité ekonomické míry monopolu moci. Současná finanční krize již startuje u mnoha firem skutečnou kreativní destrukci, což pro svět Open source může mít jen pozitivní efekty. V lednovém vyjádření výkonného ředitele Microsoftu Stevena A. Balmera jsme slyšeli oznámení, že ve firmě dochází k velké redukci a že jejich model není schopen k odrazu ze dna.[16] Tak tedy, co vzroste, co se utlumí a co zanikne? Příjmy z operačních systému Windows poprvé v historii padají, předinstalované Linuxové distribuce do nových notebooků v obchodních řetězců logicky snižují a podporují zdravou obchodní soutěž. Linux se ukazuje být velice populární: jeho průnik k uživatelům se rozšířil na telefony, televizní set-top boxy a na nové produkty jako 200 dolarové laptopy. Celý tento trend okomentoval ředitel neziskové společnosti Linux Foundation Jim Zemlin: „Společnosti jako Intel a Qualcomm, které vyrábějí mikročipy budou najímat Linuxové talenty, jak jen to bude možné.“

4 Open source demonstrace?

Paralely tak napomáhají určovat cestu vývoje v celkové sociologicko-filozofické rovině společnosti a testují současné klíčové faktory. Je tedy Open source jako fortifikace na moravském pomezí nepřirozeným produktem soudobých mocenských poměrů? Kdo tedy v budoucnosti bude financovat vývoj Open source aplikací a kde bude v závislosti na něj protěžováno hospodaření? Nejčerstvější související kauzou je nedaleký Městský obvod Ostrava-Jih,[17] který je vzorovým důkazem, že české mocenské poměry a hlavně legitimní lobbing je nevlídný k ctnostným ideálům mnoha nadšenců. Výmluvou zastupitelstva úřadu je otázka kompatibility a problémy s aplikacemi, určenými jen pro MS Windows.[18]

Stejná, současná, ale odvážnější analogie se vyskytuje od roku 2004, kde kontroverzní vlády zemí jako Venezuela a Kuba zahájily program přechodu na svobodné softwary. Následně v rozhovoru pro news.com spoluzakladatel společnosti Microsoft Bill Gates označil Open source za současnou formu komunismu, aby tak dezorientoval veřejnost pošpiněním

[15] `http://en.wikipedia.org/wiki/Joseph_Schumpeter`

[16] `http://www.nytimes.com/2009/01/26/technology/26spend.html`

[17] Vzorový úřad, kde se používají distribuce Linuxu na 10 serverech a 285 pracovních stanic.

[18] `http://www.root.cz/clanky/ostrava-jih-prechazi-pod-tlakem-linux-windows/`

pověsti linuxové komunity. Není se tedy čemu divit, že společnosti jako Microsoft Cor. lobbují ve velkém i u těchto vlád, které stále úspěšně odolávají tlaku softwarového monopolu. Klíčem k jejich postoji je zde důsledná víra přenosu revolučních ideálů (samozřejmě vlastních mocenských cílů) a silná protiamerická kampaň.

Další, ale o mnoho „podvratnější“ činností vůči všemu komerčnímu je čisté pirátství a to oficiální íránská distribuce Windows, Adobe, Autodesk, etc. pod značkou LORD, která se vyskytuje na pultech v přepočtu za pouhých 300,– Kč. Bohužel Írán v boji s monopolem nemá jasnou politickou koncepci jako třeba Venezuela a tudíž je a bude do budoucna odsuzováno jako PMR s obchodem se zbraněmi.[19]

Přesto popularita Open source stoupá, jak nám vychází z prestižní statistiky W3Counter, kde na referenční skupině 64 miliónů unikátních adres bylo v roce 2007 zjištěno 1,26 % uživatelů Linuxu, tak už v dubnu 2009 je zaznamenáno 2,16 % instalací.[20] Zrození internetu stálo na poskytovatelích služeb a proto se nerozšířili systémy jako Prodigy nebo The Source.[21] Internet do té doby omezen pouze na vládní, akademické, výzkumné a firemní nastavení byl v roce 1992 otevřen pro komerční subjekty a urychlil se tak vývoj Internetu směrem k informačnímu a komunikačnímu zdroji pro spotřebitele a podniky. Linux nikdy nebyl v držení akademiků a výzkumníků. Proč se tedy nerozšířil dříve? Nebo čeká na zkrocení konkurence jako u Internetu populárním operačním systémem Ubuntu v době světové krize?

5 Public relations

Nejvýznamnější postavou volného sdílení a volného modifikování je Richard M. Stallman. Jako první začal bojovat právně proti komercializaci myšlenek, což vedlo podle něj k uzavírání samotného software. V roce 1985 založil neziskovou organizaci *Free Software Foundation* (FSF),[22] která zejména právně podporovala projekt GNU. Licence spravované nadací prošly mnoha změnami, ale ve svém základu se jednalo o kosmetické úpravy. Pomyslnou ránou pro komunitu bylo vzniklé schizma v roce 1998, kde založená organizace *Open Source Initiative* (OSI) vychází ze stejných myšlenek jako FSF, ale rozdíl je víceméně sémiotický. OSI na sebe strhla většinu komunity, získala mnoho vznikajících firem a současně obhospodařuje desítky open source softwarových licencí.

Porovnáme-li vývoj společností s dynamickým vstupem na scénu myšlenek sdružení *Creative Commons* (CC), tak zjišťujeme, že OSI ve své síle mnoho ztrácí. Proč? Zakladatel CC Lawrence Lessig je právník a autor mnoha publikací, tedy člověk s kritickým celospolečenským pohledem na problematiku autorsko-právních otázek. Ve své knize *Svobodná kultura* (Free Culture) zejména v historických paralelách sumarizuje hlavní myšlenky, které

[19] František Zachoval: Digitální lobbing / Digital Lobbying. Flash Art, No. 4, 2008

[20] `http://www.w3counter.com/globalstats.php`

[21] `http://en.wikipedia.org/wiki/Online_service`

[22] `http://programujte.com/?akce=clanek&cl=2006021202-historie-open-source-a-free-software-1-cast`

jsou základem pro snadné pochopení samotné problematiky. Z podstaty své právní profese jde dál a odstartoval snahy licenčních podmínek CC lokalizovat do dalších právních řádů. CC má globální aspiraci, která je sice obtížně realizovatelná, ale implementační proces proběhl již v desítkách zemí světa.[23] Mikrospolečenství uživatelů licencí CC svou dynamikou a pozitivní PR je pro nejednotné a hlavně nesrozumitelné Open source hnutí stále aktuální inspirací a výzvou.

6 Závěr

Zde zmíněné reálné historické a současné modely vytváří obraz situací Open source společenství vůči Open society. OS chybí kromě povedeného loga – styl komunikace s veřejností a prezentace myšlenek. Roztříštěnost samotné komunity např. v částečné nedůvěře komunity vůči projektu Ubuntu a nedůslednost v kooperaci s výrobci hardware. Z organizačního pohledu zde není silná osobnost, která by spojila ušlechtilé zájmy v jednotný, jasný program a cíl. Nepostradatelnou rovinou prezentace vnímám v marketingu, kde třeba CC je v dnešních dnech denně skloňována s přechodem licencování článků na celosvětové informační bráně Wikipedia. Díky globalizaci a internetové síti na nás čeká do budoucna mnoho nástrah co se týká komercializace myšlenek a v poslední době osobní bezpečnosti, která ve spojení s proprietárními nástroji zdravě paranoidního člověka velice znervózňuje.

Celkově Open source má skvělou startovní pozici, jelikož se odlišuje od všech svých předchůdců tím, že je okolím vnímán v podstatě pozitivně. Komunita má oproti zde uvedeným reálným projektům docela reálný potenciál posunout společnost směrem k relativní osobní svobodě a demonstrovat tak svou pozici proti monopolu, který se zdá být v budoucnu plošně nebezpečný.

Literatúra

[1] BEY H.: Dočasná autonomní zóna. Překlad: Blumfeld, 2004, 1. vydání, ISBN 80-903452-1-2

[2] LESSIG, L.: Free Culture, How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity Hardcover. The Penguin Press, 2004, ISBN 1594200068

[3] MASON, M.: The Pirate's Dilemma. Londýn : Penguin Books, 2008

[4] MARX, K.: Kapitál: kritika politické ekonomie. Praha : Svoboda, 1978–1980

[23] František Zachoval: Demýtizace licencí Creative Commons. Ikaros [online]. 2008, roč. 12, č. 12 [cit. 2009-04-26]. Dostupný na World Wide Web: `http://www.ikaros.cz/node/5131`. URN-NBN:cz-ik5131, ISSN 1212-5075.

[5] SCHUMPETER, J. A.: Kapitalismus, socialismus a demokracie (z angl. orig: Capitalism, Socialism and Democracy). Brno : Centrum pro studium demokracie a kultury (CDK), 2004 (orig. 1942), ISBN 80-7325-044-6

[6] SLAČÁLEK O.: Od revoluce k potulce. In A2, 08/2005, ISSN 1803-6635

[7] WIHODA, M. – KOUŘIL, P.: Česká kastelologie na rozcestí? In Archaeologia historica 27/2002, ISSN 0231-5823, ISBN 80-7275-031-3

[8] WIHODA, M. – KOUŘIL, P.: Wüstehubové / Ďáblovo plémě nebo tvůrcové kulturní krajiny? In Časopis Slezského zemského muzea, ISSN 0323-0627

[9] WHEEN F.: Marxův Kapitál. Praha 2007

[10] ZITTRAIN, J. L.: The Future of Internet – And How to Stop It. USA : Yale University Press, 2008

Kontaktná adresa

František ZACHOVAL (MgA.),
DigiLab AVU v Praze, U Akademie 4,
170 22 Praha,
zachoval@avu.cz

Abstrakty ukážok a prezentácií

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

INTERAKTÍVNA TABUĽA POMOCOU WIIMOTE

BLAHO, Lukáš, (SK)

Projekt interaktívna tabuľa pomocou Wiimote som robil na v rámci bakalárskej práce pod vedením doc. Miloša Šrámka.

Tento projekt ponúka alternatívne riešenie pre interaktívnu tabuľu. Kľúčovou časťou je Wii Remote, čo je diaľkový ovládač k hracej konzole Wii od Nintenda.

Wii Remote sa dá pripojiť k PC pomocou Wiimote. Wii Remote má infračervenú kameru, ktorá dokáže snímať signál infračerveného pera. Softvér v PC dokáže spracovať signály z Wiimote a na základe nich vie simulovať prácu myši. Infračerveným perom môžeme teda písať na tabuli, na ktorú sa projektorom premieta obraz z počítača, a tým ovládať operačný systém (klikať, presúvať okná, . . .).

Funkcionalita tohto finančne nenáročného riešenia je porovnateľná s profesionálnymi interaktívnymi tabuľami, ktoré stoja niekoľko stoviek až tisíc €.

DIDAKTICKÝ NÁSTROJ iTalc

FEDORIK, Slavko, (SK)

Pri vyučovaní sa stáva použitie prostriedkov výpočtovej techniky bežným javom. Počítače sa tak stávajú učebným nástrojom i učebnou pomôckou, čo so sebou prináša viacero problémov, ktoré v klasickej učebni nebolo treba riešiť. Didaktický nástroj iTalc (`http://italc.sourceforge.net`) umožňuje niektoré z týchto problémov výrazne zjednodušiť alebo dokonca odstrániť.

Jednak zjednodušuje správu počítačovej učebne tým, že umožňuje centrálne zapínať a vypínať počítače, a to vrátane prihlasovania a odhlasovania používateľov, čím si pedagóg ušetrí obiehanie strojov niekoľkokrát denne. A jednak pomáha pri samotnom vyučovacom

procese, už len tým, že poskytuje vizuálny prehľad činnosti študentov zobrazením ich obrazoviek. Takto učiteľ získava okrem prehľadu, či žiaci pracujú na tom čo majú za úlohu, ako aj prehľad o tom, či pri riešení úlohy postupujú správne. Pomocou vzdialeného prístupu k počítačom zase môže pedagóg jednoducho nasmerovať študenta smerom k správnemu riešeniu, či ho upozorniť na chybu.

Pomocou posielania správ jednotlivcom, či všetkým naraz, možno jednoducho upriamiť pozornosť na určitý cieľ a v prípade, že počítače pôsobia skôr rušivo a odvádzajú študentov od preberaného učiva, je možné vzdialene zamknúť ich vstup (myš, klávesnica), prípadne zamknúť obrazovku a tým ich pozornosť znova pritiahnuť k vzdelávaciemu procesu. Samozrejme, je len na osobnosti pedagóga, či sa tento program stane v jeho rukách (počítači) nástrojom na obmedzovanie, zväzovanie a naháňanie študentov, alebo to bude nástroj, ktorý skvalitní a zefektívni vyučovací proces. A toto zefektívnenie môže nastať ako v prostredí Windows (2000 a XP), tak i v prostredí GNU/Linux (32 i 64 b) a dokonca aj v prostredí zmiešanom.

TERMINÁLOVÁ UČEBŇA

FEDORIK, Slavko, (SK)

Problémy bežnej počítačovej učebne pozná každý, kto v nej učí i kto sa stará o jej funkčnosť. Spravovať viac samostatných počítačov je časovo náročné, počítače i softvér zastarávajú, hučia, hrejú. . .

Pritom už viac ako dvadsať rokov je riešenie tohto problému známe a volá sa terminál, presnejšie grafický terminál. Pod týmto pojmom sa skrýva počítač, ktorého dvojjedinou úlohou je zabezpečiť vstup od používateľa (myš, klávesnica) a zobraziť výstup programov (obrazovka, reproduktory). Táto dvojjediná úloha je zabezpečovaná viacerými spôsobmi a jedným z nich je Linux Terminal Server Project (LTSP – `http://ltsp.org`).

Riešenie na báze LTSP umožňuje použitie tenkých klientov (bezdiskových staníc), ktoré same o sebe neobsahujú nič o čo by sa bolo treba starať. Pri štarte si stiahnu operačný systém zo sieťového servera, spustia ho a umožnia používateľom prihlásenie. Prihlásenie sa však nevykonáva lokálne, ale opäť sa prihlasujú na server, na ktorom bežia všetky aplikácie.

V úlohe tenkého klienta môže byť použitý aj starý počítač, ktorý by mal mať minimálne 300 MHz procesor a aspoň 64 MB RAM. Keďže sa všetko vykonáva po sieti, potrebuje sieťovú kartu a k tomu vybudovanú sieť. Výhoda je pravdepodobne zrejmá na prvý pohľad. Všetko beží na serveri, a tak netreba strácať čas obiehaním jednotlivých strojov pri aktualizácii, či nastavovaní softvéru. Táto výhoda je zároveň nevýhodou, pretože ak nastane porucha servera, je nefunkčná celá učebňa. Každopádne však jednotlivé klienty morálne nezastarávajú, pretože naozaj obsluhujú len myš, klávesnicu, či monitor.

Riešenie LTSP je zahrnuté v mnohých distribúciách GNU/Linuxu, pričom medzi najlepšie implementované patria distribúcie Ubuntu, Debian a OpenSuse, a umožňujú nainštalovať

celé prostredie pomocou niekoľkých príkazov. Súčasťou tejto prednášky bude nielen ukážka činnosti takéhoto LTSP prostredia, ale aj možnosť osobne si vyskúšať, ako dokáže fungovať taká malá skrinka, so spotrebou 15 W.

SOFTVÉR GeoGebra A PROJEKT STOČ NA PF KU V RUŽOMBERKU

GUNČAGA, Ján, (SK)

Cieľom prezentácie je ukázať niektoré možnosti využitia softvéru GeoGebra vo vyučovaní matematickej analýzy, ako aj projekt Študentskej tímovej odbornej činnosti na PF KU v Ružomberku ako formy ŠVOČ pre študentov učiteľstva matematiky, zameranej na tvorbu apletov v GeoGebre.

KVM – VIRTUALIZÁCIA V JADRE LINUXU

KAUKIČ, Michal, (SK)

V ukážke bude predvedená konfigurácia spojazdnenie virtuálnych strojov pomocou KVM (Kernel-based Virtual Machine) a VDE (Virtual Distributed Ethernet) v distribúcii Debian GNU/Linux. Použitím KVM môžeme na jednom PC vytvoriť niekoľko virtuálnych PC. Na nich môžu bežať nemodifikované operačné systémy (GNU/Linux, Windows, FreeBSD, atď.). Každý virtuálny stroj má svoj vlastný virtualizovaný hardvér: sieťovú kartu, disky, grafickú kartu, atď. KVM sa dá použiť na systémoch, kde procesor má hardvérovú podporu virtualizácie.

Na zosieťovanie používame virtuálny switch z balíka vde2 (časť projektu Virtual square, `http://www.virtualsquare.org`), čím sa konfigurácia výrazne zjednoduší. Rýchlosť siete a diskových operácií sa dá zvýšiť použitím modulov jadra pre paravirtualizáciu (`virtio_blk`, `virtio_pci`, `virtio_net`).

Na virtuálnom PC v prostredí Gnome (resp. xfce4, pre úsporu zdrojov) môžeme pohodlne a rýchlo pracovať cez NX Nomachine, o čom bude reč v ďalšom príspevku.

NX Nomachine (VZDIALENÝ DESKTOP)

KAUKIČ, Michal, (SK)

NX Nomachine (`http://www.nomachine.com`) je spôsob, ako sa vzdialene pripojiť na grafické desktopové prostredie či už v GNU/Linuxe, alebo aj vo Windows. Je veľmi rýchly a máme odskúšané napr. vzdialené pripojenie z Bratislavy či Kežmarku na server do Žiliny.

Ukážeme, ako si nainštalovať slobodný NX server, bez obmedzení na počet používateľov, ako nakonfigurovať NX klientov. Na konferenčnom DVD bude video o tejto konfigurácii. Ukážeme aj naživo pripojenie pomocou NX na viacero vzdialených počítačov.

Sage a Pylab NA ŠKOLÁCH

KAUKIČ, Michal, (SK)

Systémy Sage a Pylab sú vysokoúrovňové nadstavby, postavené na základoch pozostávajúcich z programovacieho jazyka Python, ale aj veľa OSS programov a knižníc. Hlavne Sage sa snaží byť hodnotnou OSS alternatívou ku komerčným systémom ako Maple, Mathematica and Matlab. V ukážke načrtneme možnosti týchto systémov pri výučbe na stredných a vysokých školách, hlavne v oblasti grafiky a symbolických výpočtov. Ukážeme tiež možnosti interaktivyty systému Sage, ktoré sú veľmi podobné analogickej funkcionalite v programe Mathematica.

PREHĽAD A VYUŽITIE OS GIS A SLOBODNÝCH PRIESTOROVÝCH DÁT

KUBÍK, Dominik, (SK)

Priestor a poloha determinujú činnosti človeka od nepamäti. Aj preto sa človek od začiatku snažil o ich pochopenie, zaznamenávanie a spracovanie. Výpočtová technika vysokou mierou prispela k automatizovanému a rýchlemu spracovaniu priestorových údajov a tvorbe máp. Už v počiatkoch softvérového vývoja vznikali viaceré aplikácie pre spracovanie priestorových grafických údajov, ktoré sa naďalej vyvíjajú a zdokonaľujú. Špecifickú oblasť spracovania priestorových dát neobišli ani aktivity priaznivcov open source softvéru a množstvo jeho podporovateľov pracuje a rozvíja viaceré kvalitné GIS aplikácie.

Najstaršou skupinou aplikácií pre spracovanie priestorových údajov sú desktopové aplikácie (silní GIS klienti). Vznikajú už od čias veľkých sálových počítačov. Rozšírenie osobných počítačov, legendárny príkazový riadok a nástup GUI pre OS ešte viac podporili ich vývoj.

Druhou skupinou nepriamo previazanou s desktop GIS aplikáciami sú databázové systémy a ich rozšírenia na špecializované spracovanie priestorových dát (spatial database). S nástupom internetu vznikla potreba zdieľať a publikovať priestorové údaje a dynamické mapy. Pre tieto potreby vznikajú internetové mapové servery. V náväznosti na tieto aplikácie definujeme aj ďalšiu skupinu aplikácií, tzv. tenkí GIS klienti, resp. prostriedky umožňujúce ich vytvorenie (GIS frameworks). Poslednou podskupinou aplikácií, ktorú nie je možné úplne

oddeliť od predchádzajúcich, sú aplikácie na prácu a spracovanie údajov z GNSS systémov (GPS údajov).

Bezplatné a voľne šíriteľné priestorové údaje predstavujú špecifickú skupinu dát. Nie všetky sú dostupné pod GNU/GPL a ich využitie limitujú viaceré druhy licencií ako napr. *Creative Common licence*. Vďaka nim je možné získať prístup k údajom o uličných sieťach, k satelitným snímkam, modelom terénu, ale aj metadátam.

Celkový rozvoj a smerovanie OS GIS aplikácií dozorujú viaceré organizácie, ktoré prispievajú k tvorbe a usmerňovaniu štandardov s cieľom jednoduchého a jednotného šírenia priestorových dát.

SPRACOVANIE ZVUKU V SYSTÉME GNU/Linux

LAJČIAK, Pavol, (SK)

Na začiatku ukážky budú stručne vysvetlené pojmy ako sú zvuk, perióda, frekvencia zvuku a jeho amplitúda. Ďalej budú vysvetlené pojmy ako sú vzorkovanie, kvantovanie, vzorkovacia frekvencia, perióda vzorkovania a rozlíšenie (bitová hĺbka). Tiež bude poukázané na rozdiel medzi analógovým a číslicovým zvukovým signálom a jeho spracovaním.

Vysvetlené budú aj pojmy ako online (realtime) a offline (postprocessing) spracovanie a deštruktívne a nedeštruktívne spracovanie zvuku.

V rámci ukážky bude predvedená inštalácia aplikácie Audacity pod systémom Windows a GNU/Linux. Následne bude predstavená slovenská lokalizácia aplikácie a jej inštalácia pre systémy Windows a GNU/Linux. Inštalácia knižnice Lame potrebnej na vytváranie `mp3` súborov.

Účastníci budú zoznámení s prácou so základnými nástrojmi, úpravami, generovaním zvukov, efektami ktoré môžu na zvuk aplikovať a možnosťami rozšírenia Audacity o ďalšie efekty (Nyquist, LADSPA a VST).

V závere bude stručné pojednanie pojednanie o ďalších editoroch zvuku, distribúciach určených pre spracovanie multimédií, zvukových systémoch ALSA a OSS a PULSE AUDIO používaných v systéme GNU/Linux a o otvorených formátoch `speex`, `ogg vorbis`, `flac`.

MAPTILER – MAPY HROU

MICHÁLEK, Juraj, (SK)

Kartografia a spracovanie máp je pomerne starou vednou disciplínou. Digitálne technológie prinášajú nové možnosti spracovania a prístupu k mapám. Pri zobrazovaní máp sa používajú dva prístupy. Zobrazovanie vektorových máp, ktoré je veľmi presné, avšak pri zväčšovaní je nutné neustále prepočítavať rôzne *zoom levely* (úrovne priblíženia). Druhý typ

zobrazovania je pomocou rastrových obrázkov, ktorý je bežne využívaný v aplikáciách ako Google Maps.

Softvér Maptiler umožňuje napočítať jednotlivé zoom levely veľmi presne a prehľadne. Takže z geograficky lokalizovaného obrázku, napríklad z naskenovanej mapy, je možné vytvoriť digitálnu mapu. Túto mapu je potom možné zobraziť nad inými mapami ako napríklad GoogleMaps alebo OpenStreetMap. Softvér umožňuje export do formátu KML, ktorý je možné zobraziť pomocou aplikácie Google Earth ako 3D mapu. Domovská stránka projektu Maptiler je `http://www.maptiler.org`.

Ako vstup pre Maptiler sa dajú použiť georeferencované obrázky `tiff`, prípadne aj obyčajný naskenovaný obrázok, ktorý je ale nutné pred samotným výpočtom správne georeferencovať, tzn. umiestniť správne do priestoru geografických súradníc. Projekt Maptiler je open source, napísaný v programovacom jazyku Python. Na rasterizáciu a rýchle matematické transfotmácie súradníc používa knižnicu Geospherical Data Abstraction Library – GDAL. Maptiler funguje na operačných systémoch Linux, Windows a Mac OS.

Pomocou programu Maptiler je možné jednoducho a rýchlo vytvoriť adresár obsahujúci vyrendrovanú rastrovú mapu. Adresár s vyrendrovanou mapu stačí umiestniť na server. Nie je nutná žiadna serverová aplikácia, ani žiadna zložitá inštalácia na serveri. Pri výpočte zoom levelu mapy je možné špecifikovať, ktoré *zoom levely* používateľ požaduje. Typicky sa počíta level 3 až level 14. Pod level 14 môže byž výpočet veľmi náročný a môže trvať viac než týždeň. Pre zrýchlenie výpočtu autor aplikácie Petr Přidal poskytuje paralelizovanú verziu renderovacieho mechanizmu, ktorá dokáže znížiť dobu výpočtu zo siedmich dní napríklad na niekoľko málo hodín.

Maptiler je veľmi užitočný nástroj na tvorbu máp s podporou zoomovania. Je jednoduchý na používanie a vďaka výpočtom pomocou GDAL je aj veľmi rýchly. Používa sa napríklad pri digitalizácii starých máp v knižniciach v projekte Old Maps Online.

MIGRÁCIA NA OPEN SOURCE NÁSTROJE VO VÝVOJI KOMERČNÝCH PRODUKTOV

STRÁŽOVEC, Peter, (SK)

Pri vývoji riešení v oblasti IT je základnou otázkou voľba správnych platforiem na jednotlivých úrovniach. Nezávisle od konkrétneho riešeného problému sú týmito platformami hardvér, operačný systém a knižnice, ktoré sú navrhnuté pre tvorbu a použitie vo finálnom produkte.

Tieto platformy navzájom úzko súvisia a výber jednej z nich priamo ovplyvní možnosti výberu zvyšných. Okrem výberu z konkrétnych typov možností pre jednotlivé úrovne platforiem je veľmi podstatná otázka licencie a dostupnosti zdrojových kódov. Knižnice a API pre vývoj, operačný systém alebo niekedy dokonca aj samotný hardvér – sú dostupné od rôznych dodávateľov, s rôznymi vlastnosťami a so špecifickým licencovaním.

Vlastnosti open source licencií a pridané hodnoty, ktoré produkty licencované týmto spôsobom prínášajú, majú veľmi dôležité výhody pre vývojarov IT riešení, ktoré ich robia neraz tou najlepšou voľbou platformy pre finálny produkt.

Pracoval som na vývoji viacerých produktov, kde výber open source platformy výrazne ovplyvnil a vylepšil vývoj, či už to bolo rozhodnutie na začiatku vývoja, alebo migrácia počas behu vývoja s účelom zlepšenia.

SLOBODNÝ A OTVORENÝ SOFTVÉR NA ŠKOLÁCH & PREDSTAVENIE KONFERENČNÉHO DVD

ŠRÁMEK, Miloš, (SK)

Zrejme najpríťažlivejšou črtou slobodného a otvoreného softvéru (SOS) je to, že je dostupný zdarma a pritom plne legálne. Keď zvážime ďalšie črty, ktorými sú šírka výberu programov a možnosť nasadenia od mininotebookov až po výkonné servery, vidíme, že SOS má veľkú šancu na uplatnenie v najrôznejších oblastiach, školy a vyučovanie nevynímajúc.

V príspevku sa zameriame na viaceré témy. Uvedieme krátky pohľad do histórie, priblížime licenčné podmienky, ktoré sú právnym základom odlišnosti SOS od komerčného softvéru a priblížime oblasti jeho aplikácie. V druhej časti sa zamyslíme nad príčinami, prečo SOS dnes na školách nehrá takú rolu, akú by podľa nás hrať mohol a prečo sa šanca, ktorú SOS poskytuje, dostatočne nevyužíva. V poslednej časti poukážeme na možnosti, ako tento stav zmeniť, či už používaním SOS pre OS Windows, ktorý je dnes na školách najrozšírenejší, alebo používaním OS GNU/Linux a jeho mnohých aplikácií.

V poslednej časti predvedieme konferenčné DVD, ktoré je okrem štandardného vybavenia konferenčnými príspevkami aj nosičom spúšťateľného operačného systému GNU/Linux vo verziu Ubuntu 9.04, ktorú sme pre potreby konferencie rozšírili o viaceré programy použiteľné na školách. Ukážeme, ako z DVD spustiť systém, ako ho bez inštalácie preniesť na USB kľúč a napokon, ako ho plnohodnotne nainštalovať na počítač.

OpenSolaris – STRUČNÝ ÚVOD A POROVNANIE S GNU/LINUXOM

TELKA, Marcel, (SK)

Prezentácia začína úvodom do projektu slobodného operačného systému OpenSolaris, dostupného na `http://opensolaris.org/`, jeho vznik, účel a rôzne distribúcie OpenSolarisu vychádzajúce z tohoto projektu. Dôležitá časť prezentácie je venovaná unikátnym technológiam a inováciam, ktoré OpenSolaris prináša (napr. DTrace, ZFS, Time Slider, virtualizácia, IPS, NWAM, SMF, Distribution Constructor, atď.). Ďalej sa dozvieme o najnovšom vydaní OpenSolaris 2009.06 a získame prehľad o novinkách dostupných v tejto

distribúcii. Záverečná časť prezentácia je venovaná porovnaniu OpenSolarisu s GNU/Linuxom z pohľadu bežného používateľa, správcu systému a vývojára. Nebude chýbať ani prehľad možností využitia projektu OpenSolaris vo vzdelávaní. Na prezentácii bude možné získať LiveCD s OpenSolaris 2009.06.

PREKLAD GNOME DO SLOVENČINY

TELKA, Marcel, (SK)

Prezentácia stručne uvádza GNOME ako pracovné prostredie (desktop environment), spomína históriu GNOME a jeho najbližšiu budúcnosť. Ďalej sa zameriava na všeobecný popis fungovania Projektu prekladov GNOME (GNOME Translation Project) a jeho organizáciu. Postupne sú uvedené rôzne nástroje, ktoré majú jednotlivé prekladateľské tímy k dispozícii (depozitár zdrojových kódov, Prekliate lži – Damned Lies, bugzilla, atď.). Nasleduje konkrétny popis fungovania slovenského prekladateľského tímu a spôsob použitia nielen všeobecných, ale aj špecifických nástrojov v tíme. Na záver sú uvedené problémy, s ktorými sa slovenský prekladateľský tím v súčasnosti potýka, spolu s plánmi do budúcnosti, stručnými štatistikami a možnosťami využitia projektu prekladu GNOME do slovenčiny vo vzdelávaní.

POUŽITIE SYSTÉMU DYNAMICKEJ GEOMETRIE C.a.R.

TULEJA, Slavomír, (SK)

Prostredia dynamickej geometrie sú počítačové programy, ktoré používateľovi umožňujú vytvárať geometrické konštrukcie a po vytvorení nimi manipulovať. Primárne ide o planimetrické konštrukcie. Konštrukcia sa začne vytvárať na základe niekoľkých objektov, z ktorých sú potom zostrojené nové objekty ako sú priamky, kružnice alebo body. Po tom, ako sa konštrukcia vytvorí, možno meniť polohy pôvodných definujúcich objektov a sledovať ako sa mení konštrukcia.

V ukážke budú vysvetlené základy práce s programom C.a.R. na príkladoch vytvorenia niekoľkých jednoduchých planimetrických a stereometrických konštrukcií. Ukážka bude obsahovať aj využitie programu C.a.R. na vytváranie interaktívnych modelov pre fyziku.

Red Hat a Fedora – NOVINKY, VÝVOJ

VOKÁL, Radek, (CZ)

Novinky v distribuci Fedora 11 a plánování další verze Red Hat Enterprise Linuxu. Přehled nových projektů a vlastností, které ovlivní i další Red Hat Enterprise Linux. Rozdíl mezi komunitní a enterprise distribucí. Jak se zapojit do vývoje distribuce Fedora. Přehled projektů v brněnském vývojovém centru.

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie OSSConf 2009

Prvé vydanie 2009
Počet strán 155
Elektronická sadzba programom pdfTEX
Vytlačili C–Press, s. s r. o. Košice

ISBN 978-80-89276-16-5