Moduly eLearning systému Moodle pro potřeby výuky na UTB ve Zlíně

Bc. Pavel Jura

Diplomová práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel JURA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Moduly eLearning systému Moodle pro potřeby výuky na UTB ve Zlíně**

Zásady pro vypracování:

Vytvořte následující moduly do výukového systému Moodle:
- modul systému Moodle, který umožní kontrolovat správnost úkolů i v jiných jazycích, než C/C++ - například v jazyce Pascal, Matlab, PHP atd
- Pro některé z těchto jazyků navrhněte způsob kontroly plagiátů

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

KOSEK, J. HTML – Tvorba dokonalých WWW stránek, podrobný průvodce.
KOSEK, J. PHP – tvorba interaktivních internetových aplikací, podrobný průvodce
VÁCLAVEK, P. JavaScript, Hotová řešení.
THOMSON, L., GILMORE, W., WELLING, L. PHP a MySQL – rozvoj webových aplikací.
MASLAKOWSKI, M.: Naučte se MySQL za 21 dní.
Dokumentace systému Moodle: http://moodle.org/doc

Vedoucí diplomové práce: **Ing. Tomáš Dulík**
Ústav aplikované informatiky

Datum zadání diplomové práce: **14. února 2006**

Termín odevzdání diplomové práce: **26. května 2006**

Ve Zlíně dne 14. února 2006

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Užitečným prostředkem pro podporu výuky na UTB ve Zlíně je eLearningový systém Moodle. Tento systém je využíván také při výuce základních kurzů programování. Kontrolu a hodnocení odevzdaných programů můžeme provádět automaticky. Ušetříme tím čas učitele a omezíme riziko chyby lidského faktoru.

Tato diplomová práce se zabývá tvorbou nového modulu pro systém Moodle. Modul umožňuje provádět dvě základní činnosti: automatickou kontrolu funkčnosti programu pomocí sady testů a odhalování plagiátů odevzdaných zdrojových kódů.

Modul je užitečný pro podporu základních kurzů programovacích jazyků C/C++, Pascal a PHP.

Klíčová slova: Moodle, eLearning, plagiátorství, kurz programování, PHP, Pascal, C/C++

ABSTRACT

ELearning system Moodle is very useful aid for teaching in UTB in Zlín. System has also been used in basic programming courses. Checking and grading of student's assignments can be performed automatically. We can spend teacher's time and reduce human faults this way.

This master thesis deals with the creation of a new module for Moodle. This modul is designed for two basic operations: automatic program functionality checking and plagiarism detecting in source codes.

This module is helpful in C/C++, Pascal and PHP programming language courses.

Keywords: Moodle, eLearning, plagiarism, programming course, PHP, Pascal, C/C++

Děkuji vedoucímu práce, Ing. Tomáši Dulíkovi, za odbornou pomoc a cenné rady.

Ve Zlíně, 26.5. 2006 .....

podpis

OBSAH

ÚVOD

Internet je bezesporu médiem, které nejvíce a nejrychleji ovlivňuje dnešní svět. Velký vliv na masový rozvoj Internetu mělo a má školství. Zejména vysoké školy jsou v dostupnosti a kvalitě připojení k Internetu na nejvyšší úrovni. Nelze se tedy divit, že projekty pro podporu vzdělávání pomocí Internetu jsou dnes na vysokých školách běžným standardem, neli nutností.

Podpora vzdělávání pomocí Internetu prochází neustálým vývojem. Jedním z posledních a světově uznávaných projektů je i systém Moodle. Moodle je na UTB ve Zlíně využíván převážně Fakultou technologickou a Fakultou aplikované informatiky. Tento systém se vyznačuje modulární strukturou, takže lze podle potřeby vytvářet další funkční bloky. V minulém roce byly vypracovány dvě práce, jejichž výsledkem jsou dva samostatné moduly pro podporu kurzů programování. Výsledkem první práce je modul, umožňující automatickou kontrolu programů v jazyce C/C++. Druhým modul poskytuje učiteli možnost odhalovat plagiátorství zdrojových kódů odevzdaných úkolů.

Cílem této práce je vytvoření modulů, které by obě tyto užitečné funkce spojily a navíc umožnily podporu dalších programovacích jazyků, které jsou na UTB vyučovány. Pro vyučujícího znamená tento modul velmi cennou pomůcku a především úsporu času.

Úkol v kurzu programování má vést k pochopení probírané látky a ověření schopností studenta aplikovat nabyté znalosti v praxi. Požadavkem je samostatné vypracování programu tak, aby odpovídal zadání učitele. Výsledné hodnocení studenta musí vycházet z výsledků splnění obou požadavků – správné funkčnosti programu i posouzení samostatnosti při jeho vypracování.

Nově vzniklý modul poskytuje učiteli oba tyto výsledky. Ihned po odevzdání úkolu jsou dostupné výsledky provedených testů. Na jejich základě může být úkol automaticky ohodnocen. Druhá funkce modulu pomáhá odhalovat podobné zdrojové kódy a zobrazuje učiteli vzájemnou podobnost mezi odevzdanými úkoly.

Studenti mají možnost vidět ihned po odevzdání programu hodnocení, případně i výsledky testů. Dostanou šanci program opravit a získat vyšší počet bodů. Z tohoto hlediska může nový modul přispět ke zlepšení jejich výkonů.

I. TEORETICKÁ ČÁST

1 E-LEARNINGOVÝ SYSTÉM MOODLE

Moodle je zkratka *Modular Object-Oriented Dynamic Learning Environment* - Modulární objektově orientované dynamické prostředí pro výuku.

1.1 Popis Moodle

Moodle je softwarový balík určený pro podporu prezenční i distanční výuky prostřednictvím online kurzů dostupných na WWW. Moodle je vyvíjen jako nástroj umožňující realizovat výukové metody navržené v souladu s principy konstruktivisticky orientované výuky. Moodle je volně šiřitelný software s otevřeným kódem. Běží na *Unix, Linux, Windows, Mac OS X, Netware* a na jakémkoliv dalším systému, který podporuje PHP. Data jsou ukládána v jediné databázi (největší podpora pro *MySQL* a *PostgreSQL*, nicméně lze použít i *Oracle, Access, Interbase, ODBC* atd.).

Koncepce a celý vývoj systému Moodle jsou založeny na jistém směru v teorii učení, na způsobu myšlení, který bývá někdy stručně označován jako "sociálně konstrukcionistická pedagogika".[1]

1.2 Modul Úkol (*Assignment*)

Zadání práce vychází z tohoto modulu, který je standardní součástí Moodlu. V dokumentaci se dočteme základní charakteristiku tohoto modulu:

- U úkolů lze stanovit termín odevzdání a maximální počet bodů.
- Studenti mohou úkoly nahrát na server (jako soubor v libovolném formátu); každý odevzdaný úkol je při tom označen časovým razítkem.
- Opožděné odevzdání úkolu je přípustné; učiteli se však zřetelně zobrazí, s jakým zpožděním byl úkol odevzdán.
- Hodnocení úkolu i s komentářem lze vyplnit pro celou třídu na jediné stránce prostřednictvím jediného formuláře.
- Hodnocení je studentovi přidáno na stránku s odevzdaným úkolem a zároveň je mu e-mailem zasláno upozornění.
- Učitel si může zvolit, jestli je úkol po ohodnocení možné odevzdat znovu (k novému ohodnocení).

1.3 Jiné moduly pro kontrolu programů

Modul pro automatickou kontrolu a hodnocení úkolů v kurzech programování pro systém Moodle zatím neexistuje. Při důkladném průzkumu však byla nalezena práce, zabývající se podobným problémem.

Podpora distančního vzdělávání v předmětu Systémové programování a asemblery byla tématem týmových projektů studentů Slovenské technické univerzity v Bratislavě na Fakultě informatiky a informačních technologií. Výsledkem práce jsou moduly pro systém Moodle, umožňující testování funkčnosti programu a hledání podobností. Předmětem testování jsou pouze programy pro asembler x86.

Projekt týmu Dagwood [10] využil pro testování OS Linux. Pod tímto systémem je spuštěna služba pro odhalování plagiátů a služba pro testování funkčnosti. Celý modul pak spolupracuje s touto službou. Testování funkčnosti probíhá v emulátoru systému MS-DOS.

Projekt týmu Seagles [11] využil systém Moodle pouze pro odevzdávání úkolů. Samotné testování funkčnosti i shody programů řeší samostatná aplikace pro systém Windows. Učitel tedy musí odevzdané úkoly nejdříve stáhnout, otestovat a následně obodovat.

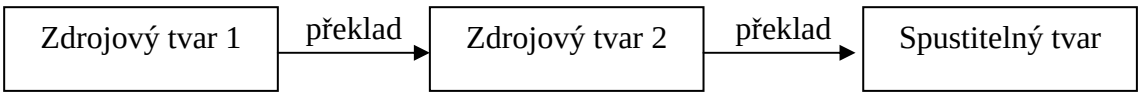
2 TEST FUKNČNOSTI PROGRAMU

Správný programovací styl nemusí zaručovat správnou funkčnost programu. Mohou se v něm vyskytovat syntaktické chyby, ale nemusí být správný ani po sémantické stránce. K testování syntaxe slouží překladače jazyka. Test funkčnosti (sémantiky) se vykoná samotným spuštěním přeloženého programu. V konečném důsledku nás zajímá jen program ve spustitelném tvaru. Jen tak lze funkčnost ověřit.

2.1 Kompilace

Kompilací rozumíme transformaci programu ze zdrojového jazyka do jiného tvaru (například do strojového kódu). Překlad nemusí skončit vždy úspěšně. Pokud je program syntakticky nesprávný, vrátí nám překladač chybu a její popis (například řádek kde se chyba vyskytuje).

Kompilace je typická pro jazyk *C, C++, Pascal, Delphi* a další. Výhodou je přeložení programu do strojového kódu, což umožňuje jeho rychlé vykonávání. Nevýhodou je závislost na platformě (operační systém, použitý hardware).



Obr. 1. Schéma překladu programu

2.2 Interpretace

Interpretace by se dala volně přeložit jako výklad. Narozdíl od kompilátoru interpret překládá zdrojový kód řádek po řádku a provádí jej. V případě chyby skončí program většinou chybovým hlášením.

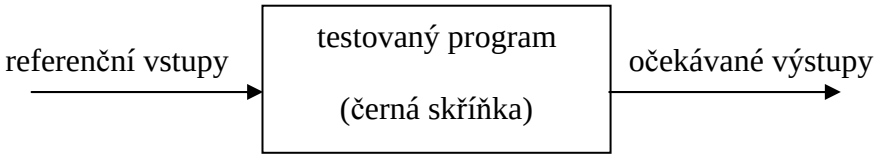
Nevýhodou interpretovaného kódu je především rychlost vykonávání programů ve srovnání s kompilovaným kódem. Výhodou je naopak velikost a přenositelnost těchto programů.

Typickými zástupci interpretů jsou jazyky *Basic, PHP, Perl, LISP* a další. Existují však také interprety jazyka *C/C++* nebo *Pascal*. Tato varianta je výhodná zejména proto, že je nezávislá na platformě a z hlediska bezpečnosti máme nad interpretovaným kódem větší kontrolu.

2.3 Způsob testování

Testování úkolů je založeno na principu černé skříňky. Cílem testování je zjištění, zda vstupně - výstupní chování programu odpovídá specifikaci. Při této metodě neuvažujeme vnitřní strukturu a logiku programu. Testovací vstupy se odvíjejí přímo ze specifikace programu a musí být zadány tak, aby otestovaly všechny třídy ekvivalence vstupů. Třídy ekvivalence mají tu vlastnost, že pokud se při daném vstupu zjistí chyba, potom stejnou chybu je možné odhalit použitím jiného vstupu z dané ekvivalence. Použitím této vlastnosti je možno redukovat množinu testovacích údajů tak, že se do množiny přidají jen „hranice“ a „střed“ třídy ekvivalence.

Testované vstupy jsou nasměrovány na vstup testovaného programu, který vrátí výsledek. Potom jsou tyto výsledky porovnány s očekávanými.



Obr. 2. Model testování programu

2.4 Omezení testů

Z předchozích odstavců vyplývá pro automatické testování programů několik možných druhů omezení:

- Chceme-li program testovat, musíme mít k dispozici překladač či interpret pro daný jazyk.
- Při vykonávání programu nesmí být porušena bezpečnost počítače, na kterém je program testován.
- Musí existovat způsob, jak do programu zadat vstupy a jak zachytit výstupy bez interakce s uživatelem.
- Všechny programy musí mít jednoznačný vstup a výstup – toto omezení je kladeno zejména na učitele, aby program správně zadal.

3 ODHALOVÁNÍ PLAGIÁTŮ

3.1 Definice problému

Alan Parker a James Hamblen, autoři práce Computer Algorithms for Plagiarism Detection (1989), definují softwarový plagiát jako *„program který byl vytvořen na základě jiného programu s malým počtem změn.“*

Při zadávání stejného úkolu celé skupině studentů v kurzu je běžné, že studenti mezi sebou o problému diskutují a následně použijí podobný nebo stejný algoritmus. Tento postup je běžný, a poučka „nedělat, něco co už někdo vymyslel“ je obhájitelná. Jedním z problémů je tedy způsob, jak odlišit zkopírovaný kód od toho, který vznikl spoluprací. Lze využít toho, že v programovacích jazycích lze jeden problém řešit vícero způsoby (použitím jiných funkcí, jiných konstrukcí atd.).

V kurzech programování se tedy zaměříme na odhalování plagiátů zdrojových kódů, což jsou obyčejné textové soubory. Odhalováním plagiátů budeme tedy uvažovat vzájemné porovnání textových souborů.

3.2 Metody hledání plagiátů

3.2.1 Metoda počítání atributů

Tyto techniky jsou založené na zjišťování počtu parametrů zdrojového textu programu, které jsou nejčastěji modifikované. Liší se pouze počtem zjišťovaných parametrů, případně mírou, jakou se údaje podílejí na celkovém hodnocení podobnosti. Jejich největší společnou nevýhodou je, že zanedbávají programovou strukturu a tok řízení. Nejčastěji jsou zjišťovány tyto parametry:

- Počet unikátních operátorů.
- Počet unikátních operandů (identifikátorů).
- Celkový počet operátorů.
- Celkový počet operandů.
- Počet řádků kódu (komentáře a prázdné řádky jsou vynechány).
- Počet proměnných deklarovaných a použitých.

- Celkový počet řídicích příkazů.

3.2.2 Metoda porovnání struktury

Tato metoda pracuje pouze s elementy, které mají význam pro funkčnost programu. Programy pracují obvykle ve dvou fázích:

- Vytvoření postupnosti informačních znaků (tokenů) pro každý program. Takto vytvořená postupnost obsahuje informace o struktuře analyzovaného programu. Jednotlivé tokeny popisují příkazy, případně bloky příkazů.
- Porovnání souborů tokenů jednotlivých programů mezi sebou se zahrnutím specifik jazyka, v kterém jsou analyzované programy napsané.

Pro tuto činnost lze velmi dobře využít možností počítače. Popsání struktury programu může být podobné, jako když kompilátor překládá zdrojový kód do podoby spustitelného programu. Odtud vyplývá fakt, že zkompilované programy originálního a upraveného zdrojového kódu budou minimálně velmi podobné. Proces vytvoření struktury programu bude dále označován jako ***tokenizace***.

3.3 Systémy pro hledání plagiátů

3.3.1 Halsteadova metrika

Patří mezi první detekční metody, uvedena je například v literatuře [2]. Je založena na principu rozložení programu na množiny operátorů a jejich operandů, kterými jsou proměnné (x, y, …) a čísla. Halstead definoval několik měřítek, které vychází z počtu operátorů a operandů. Jeden z nich je následující metrika:

$$E = (N1+N2)*log(n1+n2)$$

kde:

N1 – je počet jedinečných operátorů

N2 – je počet jedinečných operandů

n1 – je počet operátorů

n2 – je počet operandů

3.3.2 MOSS

Moss (*Measure Of Software Similarity*,) je automatický systém pro určování podobnosti mezi zdrojovými kódy jazyku *C, C++, Java, Pascal, Ada, ML, List a Schneme*. Jeho hlavní použití je právě v kurzech programování.

Moss funguje jako internetová služba, pomocí skriptu odešlete soubory zdrojových kódů ke zkontrolování na server, kde systém běží. Jakmile ***Moss*** kontrolu dokončí, odešle vám zpět výsledky v podobě HTML souboru, kde jsou uvedeny podobnosti mezi zdrojovými kódy. V podobných souborech jsou zvýrazněny části, které se jeví jako individuální (nezkopírované). Program také automaticky přeskakuje části kódu, které jsou součástí knihoven a jsou vždy stejné.

Další podrobné informace o systému uvádí zdroj [3].

3.3.3 YAP (YAP3)

Je to metoda přeložení zdrojového kódu na popis struktury kódu, která reprezentuje algoritmus zkoumaného programu, a vzájemné porovnání struktur. Toto by mělo vést k odhalení zdrojových kódů, které používají minimálně velmi podobnou posloupnost operací.

Proces se skládá ze dvou fází:

- Tokenizace.
- Porovnání tokenů.

První fází je ***tokenizace***, neboli převedení textu na posloupnost funkčních elementů zdrojového kódu. V této fázi je použito standardních funkcí operačního systému Linux a celý proces tokenizace je napsán v jazyce *bash* (viz. 4.1). Proces tokenizace zahrnuje především tyto činnosti:

- Odstranění komentářů a řetězců, které jsou konstanty.
- Změna všech písmen na malá písmena.

- Mapování synonym na společnou formu (například *strcnpy* je mapováno *strcmp*[1]).
- Uspořádání funkcí podle jejich volání a každé volání je nahrazené tokenem FUN.
- Odstranění všech tokenů, které nejsou ve slovníku daného jazyka[2].

Systém YAP prošel vývojem až k nynější verzi 3, která používá k porovnání tokenů algoritmus *Running Karp-Rabin, Greedy String Tiling* ([4],[5]). Tato metoda byla navržena pro určení stupně podobnosti párů řetězců. Její výhodou je, že je imunní vůči záměně pořadí (transpozici), podřetězců. Podrobnější informace o YAP3 naleznete například v [6].

[1] tyto funkce jsou naprosto identické. V každém jazyce existuje několik takových funkcí.

[2] především klíčová slova a standardní funkce jazyka (například *for, while, return, strcpy, write, printf, ...*)

4 VYUŽITÍ OS LINUX

Sytém Moodle, jak bylo řečeno v 1.1, může využívat různé operační systémy. Na UTB je pro potřeby výuky využit systém Linux. Na výběru systému je závisí možnosti testování programů a použité metody pro hledání plagiátů. Následuje popis důležitých utilit OS Linux.

4.1 BASH

Jako většina *shellů* (prostředí příkazové řádky) dostupné v *Linuxu* není ***bash*** (*Bourne Again Shell*) pouze výbornou příkazovou řádkou, ale zároveň také plnohodnotným skriptovacím jazykem. Skriptování v *shellu* umožňuje zautomatizovat mnoho úloh, které by jinak vyžadovaly psaní spousty příkazů. Mnoho programů v jednotlivých distribucích *Linuxu* či *Unixu* jsou pouze skripty příkazové řádky.

Bash patří mezi tzv. interpretované jazyky, to znamená že napsaný program není přeložen do strojového kódu, místo toho interpreter jazyka čte instrukce ze zdrojového kódu a podle obsahu tyto instrukce vykonává. Protože interpreter čte každou instrukci jednotlivě, zpracování skriptu není úplně nejrychlejší (program je pouze částečně překládán za běhu). Hlavní výhoda skriptovacích jazyků spočívá v jejich přenositelnosti na jiné platformy či operační systémy, jejich jednoduchosti a také velikosti.

Skripty v ***bash*** jsou jakousi obdobou *.bat* souboru v systému *MS-DOS*, ale jelikož operační systémy UNIX jsou založeny na ovládání pomocí příkazů, lze pomocí sekvencí volání těchto příkazů zautomatizovat opravdu velké množství činností, kvůli kterým bychom jinak museli psát a překládat speciální program v některém z běžných jazyků (C, C++, Pascal).

V této práci je ***bash*** využit především jako skriptovací jazyk. Funkce ***bash*** jako *shellu* je použita pouze pro spuštění konkrétního programu.

4.2 Editor sed

Sed je neinteraktivní, neboli dávkově ovládaný editor. Interpretuje skript a provádí příkazy skriptu. Sed je dávkově orientovaný, protože podobně jako mnoho jiných unixových programů je vstup přijímán programem a posílán na standardní výstup. Vstup editoru sed

zpravidla pochází ze souboru, ale může být také přesměrován z klávesnice. Výstup je implicitně směřován na obrazovku, ale může být také posílán do souboru.[7]

Příkaz se skládá z:

[adresa[,adresa]] instrukce [argumenty]

Adresy nemusí být uvedeny, sed potom provede instrukci na každou řádku.

Sed pracuje takto:

Pro všechny řádky:

- Načte řádek ze souboru.
- Pro všechny příkazy:
    - Vezme první příkaz a jestli adresa platí na soubor provede instrukci.

Adresou muže být číslo řádku, speciální znak (začátek a konec řádku) nebo regulární výraz. Některé základní instrukce:

- d (*delete*) – mazání řádku
- n (*new*) – načtení nového řádku
- a (*append*) – připojení jednoho nebo více řádku k aktuálnímu
- s (*substitute*) – nahrazení vzorku jiným textem
- další

4.3 Jazyk awk

Awk je skriptový jazyk pro kontrolu vzorků, který zpracovává soubory složené z vět stejné nebo proměnné délky. Věty jsou odděleny nějakým oddělovačem (implicitní oddělovač je nový řádek). ***Awk*** lze používat z příkazového řádku nebo ze skriptů ***awk***. Většinou se tento program vyvolává zapsáním *awk* na příkazový řádek.[7]

Pomocí ***awk*** lze provádět například tyto operace:

- Pohodlně zpracovávat text, jakoby se skládal z polí a vět a byl uložen v textové databázi.
- Používat proměnné k provádění změn v databázi.

- Ze skriptu volat příkazy příkazového interpretu.
- Provádět aritmetické operace s řetězci.
- Používat programátorské konstrukce, jako jsou skoky a podmínky.
- Definovat své vlastní funkce.
- Další.

Awk obsahuje mnoho vlastních příkazů a je to vlastně jazyk podobný *shellu*, který je specializovaný pro práci s řetězci.

4.4 Bezpečnost

Spouštění programů představuje jisté bezpečnostní riziko. Prvním rizikem je možnost „zamrznutí“ programu. Tuto otázku lze řešit vymezením času pro běh programu. Po uplynutí určené doby je program ukončen bez ohledu na jeho stav.

Druhé riziko představuje samotné spouštění programů. Jeho chování by mohlo způsobit poškození systému, na kterém je spouštěn (například mazání souborů, získávání důvěrných dat, spouštění nevhodných programů a podobně). Ideální variantou je spuštění programu ve virtuálním prostředí. Toto prostředí je spuštěno v rámci hlavního operačního systému a tím vyloučíme jakékoliv možné poškození hlavního operačního systému i přístup k jeho datům.

4.4.1 User-mode Linux (UML)

UML je jedním z mnoha virtuálních prostředí pro systém Linux. Umožňuje vytvoření virtuálního počítače, který může mít více hardwarových nebo softwarových virtuálních zdrojů, než váš fyzický počítač. Diskový prostor virtuálního systému je umístěn na vašem fyzickém počítači. V rámci virtuálního systému lze běžným způsobem spouštět aplikace v uživatelském prostředí. Omezené virtuální prostředí umožňuje provádět cokoliv bez obav, že změny se projeví na vašem stávajícím systému nebo poškodí váš počítač.

UML je užitečný v několika oblastech. Například při vývoji nebo testování jádra systému, při výuce systému Linux, jako testovací prostor pro vývoj aplikací, při simulacích havarijních stavů a podobně.

II. PRAKTICKÁ ČÁST

5 ANALÝZA ŘEŠENÍ

Základním předpokladem úspěšného vyřešení cíle této práce, je kvalitní analýza stavu a možnosti řešení úkolu. Pokud analýze věnujeme dostatečnou dobu, projeví se to na kvalitě výsledné práce a také na úspoře času, potřebného k řešení úkolu. Častou chybou je především tvorba již existujících řešení nebo výběr nevhodných prostředků pro dané podmínky.

5.1 Současný stav na UTB

Systém Moodle na UTB funguje již přes dva roky. Za tuto dobu bylo vypsáno několik bakalářských prací, jejichž cílem byla tvorba nových doplňkových modulů pro tento systém. Na základě zkušeností s těmito pracemi a jejich osvědčení v praxi se lze vyhnout zbytečným chybám. Pro tuto práci jsou použitelnými zdroji dva existující moduly.

5.1.1 Modul Program

Tento modul vznikl v roce 2005. Autorkou je Bc. Veronika Vašková.

V této práci jsou zdokumentovány veškeré postupy vedoucí ke vzniku nového modulu pro výukový systém Moodle. Změnou databázových tabulek a skriptu modulu Úkol bylo dosaženo možnosti interpretovat odevzdané programy, porovnat je se vzorovým učitelským řešením a zobrazit a uložit výsledky pro potřeby studenta i učitele. Formulář určený k zadání programu navíc obsahuje několik nových položek, které rozšiřují možnosti interpretace a porovnávání programu (program je možné interpretovat se vstupními daty i bez nich, s rozšířením voleb *diff*[3] nebo bez tohoto rozšíření).[8]

Testování funkce programů je implementováno v interpretu *CH*. Tato metoda je bezpečná pro spouštění programů na straně serveru. Pro učitelský model je vždy vytvořen referenční výstup, podle kterého jsou pak porovnávány výsledky testů programů studentů.

Zkušenosti ukázaly, že by bylo vhodné přepracovat logiku tak, aby se při testu programu vždy spustil program učitelský i studentský se stejnými parametry. Tato změna by přinesla

[3] ***diff*** je jedním z příkazů systémů Unix/Linux pro porovnání rozdílů mezi soubory

výhodu v možnosti použití více různých testovacích hodnot. Například by bylo možné tyto hodnoty náhodně vybírat z učitelem definované testovací sady.

5.1.2 Modul PD4M

Autorem tohoto modulu je Bc. Jiří Baroš a byl vytvořen jako bakalářská práce v roce 2005. Tato práce se zabývá problémem automatické detekce plagiátorství v kurzech programování, čili vyhledáváním podobných souborů se zdrojovými kódy. Popisuje možnosti detekce, možné řešení tohoto problému, a integraci automatické detekce do online výukového systému Moodle.[9]

Tento modul pracuje velmi spolehlivě. Nevýhodou však je, že neposkytuje učiteli možnost automatického testování odevzdaných prací. Umožňuje kontrolovat úkoly v jazycích Pascal a C/C++. Pro detekci využívá systém YAP3 (viz. 3.3.3), který v základní distribuci poskytuje podporu kontroly jazyka C/C++ a Pascal.

5.1.3 Prostředky na UTB

Jelikož nový modul ACA je primárně určen pro potřeby výuky na UTB, můžeme uvažovat nad prostředky, které může server poskytovat. Základní údaje obsahuje tabulka:

Prostředek	Typ, verze
Operační systém	Linux Debian Woody 3.0
Databázový systém	MySQL 4.0.24
Skriptovací jazyk PHP	PHP 4.3.10-16
CH - interpret jazyka C/C++	4.7.0
FreePascal Compiler (FPC)	2.0.0
LMS Moodle	1.4.2

Tab. 1. Softwarové prostředky na UTB

5.2 Navržené změny

Cílem této práce bylo vytvoření modulu pro systém Moodle, který by umožnil automaticky kontrolovat úkoly v jiných jazycích než C/C++. Možné jazyky byly *Pascal, Matlab, PHP*. Tyto jazyky jsou na UTB používány a studenti programy v těchto jazycích odevzdávají pomocí systému Moodle. Druhým bodem zadání bylo navržení způsobu kontroly plagiátů pro některé z těchto jazyků.

5.2.1 Analýza programovacích jazyků

V první fázi bylo potřeba zjistit možnosti kontroly úkolů u navržených jazyků. Hlavním kritériem pro výběr byla především bezpečnost spouštění těchto programů na straně serveru.

Pro jazyk Pascal sice existují volně dostupné interprety, ale nepodařilo se úspěšně implementovat na současný server takový interpret, který by umožnil kontrolu úkolů používaných při výuce tohoto jazyka. Jediným řešením tedy zůstala kompilace, přičemž pro Linux existuje hned několik kompilátorů jazyka Pascal (*FreePascal*, *GNUPascal*, a další).

Jazyk *Matlab* sice umožňuje přes *Matlab Web Server* (dále jen *mws*) spouštění tzv. *m-file* (zdrojový soubor *Matlabu*). Tento způsob je však navržen pro aplikaci, kdy uživatel vytvoří HTML šablonu k požadovanému skriptu. Následně do této šablony vloží testovaná data, odešle je aplikaci *mws*, která následně vrátí výsledek do připravené výstupní HTML šablony. Generování HTML šablon by ještě bylo průchozí, nicméně nutnost uživatelského odeslání dat a následného získání výstupních údajů zpět je příliš komplikovaná. Další komplikací by bylo porovnávání grafických výstupů *Matlabu*. Tento jazyk byl tedy prozatím z podporovaných jazyků vyloučen.

Obdobná situace jako u *Matlabu* byla rovněž s prostředím *Mathematica*. Pro tento jazyk navíc nebyla prozatím služba *WebMathematica* na UTB zprovozněna.

U jazyka PHP je už situace jiná. Interpret PHP na serveru existuje. Navíc je to jazyk, v němž je celý systém Moodle naprogramován. Jeho výstupy lze rovněž zachytit jak ve formě výstupu (například HTML stránka) tak ve formě obsahu proměnných.

5.2.2 Univerzálnost modulu

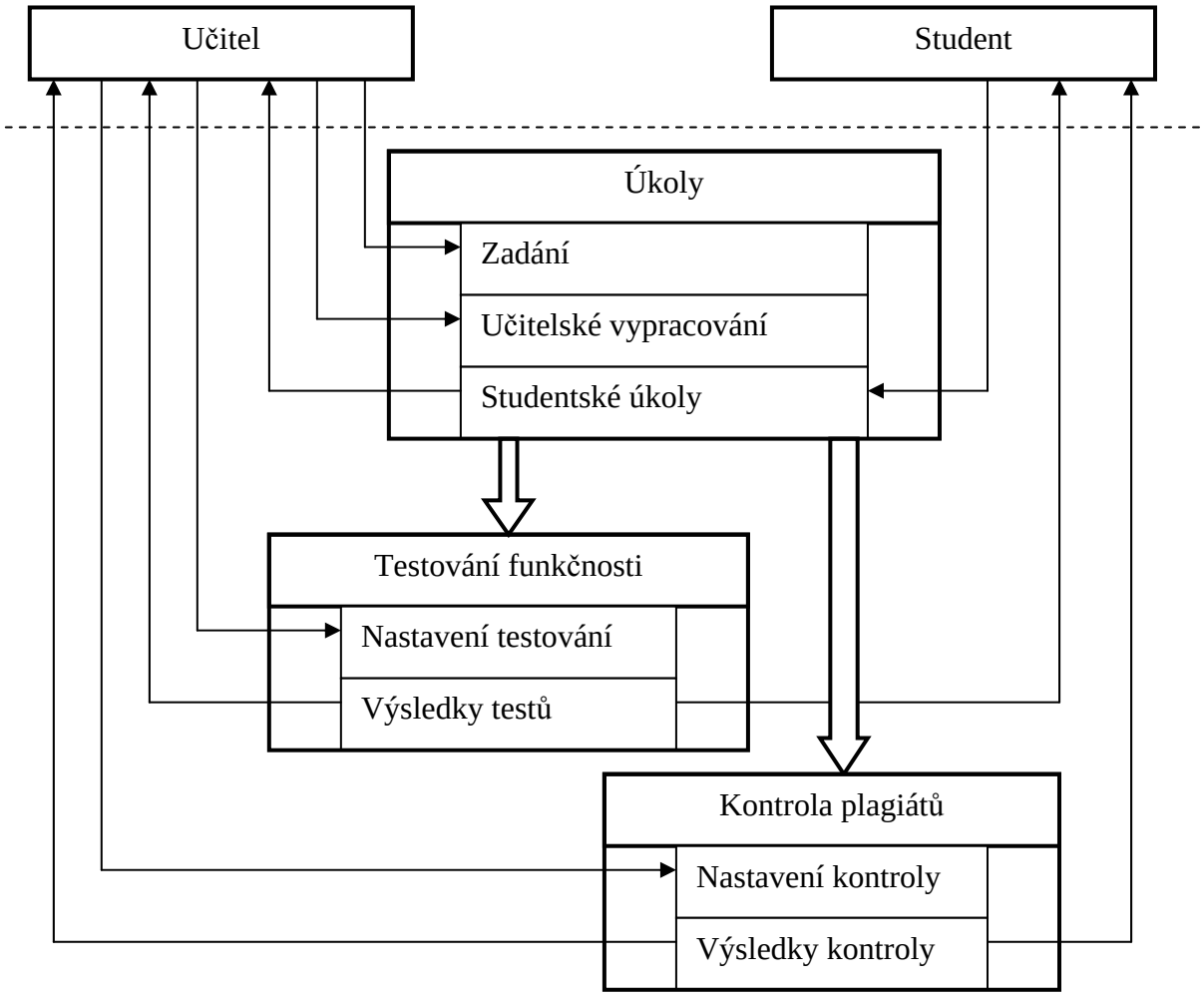
Z důvodu přehlednosti byl nový modul navržen jako univerzální pro více programovacích jazyků. Současně pro tyto jazyky nabídne možnost definování testovacích hodnot a možností hledání duplikátů. S přihlédnutím ke specifikům každého z jazyků byly submoduly pro testování funkce a hledání duplikátů navrženy tak, aby bylo možno je později přidávat a upravovat. Tím je do budoucna otevřena možnost jak pro podporu dalších jazyků, tak pro nové možnosti testování funkce i hledání duplikátů, jenž mohou být například obsahem některé z dalších závěrečných prací studentů UTB.

6 POPIS FUNKCE MODULU

6.1 Architektura modulu

Jelikož modul je součástí systému Moodle, máme vyřešen přístup uživatelů k systému. Základní činnosti nového modulu jsou tyto:

- Vytváření nových úkolů.
- Sběr zadání.
- Kontrola funkčnosti.
- Kontrola plagiátů.
- Zobrazování výsledků.



Obr. 3. Schéma činností v modulu

Mezi modulem a uživateli je čárkovanou čarou naznačeno rozhraní. V našem případě jím je WWW prohlížeč.

Jednoduché šipky představují datový tok ovládaný uživatelem.

Plné šipky naznačují automatickou činnost modulu. Tyto automatické procesy může ovládat učitel také manuálně. Například při kontrole podobností je vhodnější variantou kontrola na přímou žádost učitele. Automaticky se tedy vykonávají pouze přípravné procesy pro kontrolu.

6.1.1 Součást pro testování funkčnosti

Jak vyplynulo z analýzy řešení a předchozích zkušeností, musí tato součást obsahovat následující možnosti:

- Připojit k libovolnému úkolu libovolný počet testovacích hodnot.
- Definovat parametry testu s ohledem na možnosti programovacího jazyka.
- Provádět pro každý úkol test učitelského i studentského programu.
- Archivovat a zobrazovat výsledky všech provedených testů.
- Automaticky oznámkovat úkol na základě výsledků provedených testů.

6.1.2 Součást pro hledání duplikátů

Z výsledků analýzy víme, že stávající modul dobře funguje. Bude jej třeba integrovat do nového modulu. Rozšířením bude možnost aplikovat na úkol více testovacích metod, budou-li pro daný programovací jazyk existovat.

6.2 Návrh databáze

Podle navrženého modelu činností (viz. Obr. 1.) a návrhu funkce (viz. 5.2) sestavíme model databáze. Nový modul byl založen na modulu Úkol. Při návrhu tedy budeme vycházet z již hotového datového modelu (z důvodu zpětné kompatibility) a přidáme pouze další funkční bloky pro kontrolu funkčnosti a podobnosti. Výběr databázového systému vychází z výsledků uvedených v Tab. 1. Návrh se dále musí řídit obecnými pokyny pro vytváření tabulek v systému Moodle (uvedeno v dokumentaci systému). Proto například všechny tabulky mají stejný prefix (předponu) ***aca***.

Schéma propojení tabulek je uvedeno v příloze P I. Podrobný výpis tabulek včetně jejich typů je uveden v příloze P II.

6.2.1 Změny tabulky úkolu (tabulka aca)

Oproti původní tabulce modulu Úkol přibylo nastavení týkající se zobrazování výsledků testů a typu automatického bodování úkolu. Části pro testování funkce a podobnosti jsou pouze volitelnými součástmi, a proto v tabulce úkolu nebudou zastoupeny.

Pro podporu zobrazování byly přidány sloupce: *showcopy* a *showtest*. Udávají typ zobrazení výsledků testů.

Pro podporu testování funkčnosti přibyl pouze sloupec *sourcefile*, kam je uložen název zdrojového souboru poslaného učitelem jako vzorové vypracování.

Pro možnosti automatického známkování na základě výsledků tesů funkčnosti přibyly sloupce: *gradetype*, *grademin*, *grademax* a *graderandval*. Tyto hodnoty jsou samozřejmě využity jen v případě, že byl na úkol aplikován test funkčnosti.

6.2.2 Tabulky pro testování funkce programu

Splnění všech požadavků uvedených v 6.1.1 dosáhneme vytvořením více tabulek.

Základní tabulka pro funkci testování má název ***aca_tvs***. Představuje seznam použitelných „metod“ testování. Tyto metody jsou použitelné pro konkrétní typ jazyka (*langtype*). K této tabulce se potom vážou nastavené testovací hodnoty (viz. níže) pro úkoly.

Každý úkol může (a nemusí) mít nastaven tzv. *testovací hodnotu (**t**est **v**alue)*. Jejím obsahem jsou vstupní data pro testování a případné další parametry testu. V databázi jsou testovací hodnoty zastoupeny tabulkou ***aca_tv_active***, jenž se váže ke konkrétní metodě (*tvid*) a zároveň ke konkrétnímu úkolu (*acaid*). Tabulka obsahuje několik dalších databázových sloupců pro uchování nastavení parametrů testu (parametry příkazové řádky, použité datové soubory atd.).

Pro archivaci výsledků testů slouží tabulka ***aca_tv_results***. Testy se provádí pro každý odevzdaný úkol zvlášť. Data jsou tedy vázána ke konkrétní testovací hodnotě (*activetvid*), k úkolu (*acaid*) a odevzdanému úkolu (*submissionid*). Tabulka dále obsahuje výstupy učitelského a studentského testu a popis (většinou popis rozdílů ve výstupech). Navíc je zde jakoby duplicitní hodnota vstupu (*indata*), která však slouží pro případné rozšíření testování (viz. 11.1.2).

6.2.3 Tabulky pro hledání duplikátů

Při vytváření byl výchozím modelem předchozí modul PD4M (viz. 5.1.2). Abychom však nenarušili univerzálnost nového modulu, řešení je obdobné jako u testování programů.

Tabulka ***aca_pds*** je seznamem použitelných metod hledání duplikátů. Každá z položek je vázána opět ke konkrétnímu jazyku (*langtype*).

Nastavení testovací metody pro konkrétní úkol je uloženo v tabulce ***aca_pd_active***. Důležitou položkou je povolená hranice podobnosti (*border*) a dále možnosti automatického ohodnocení úkolu při zjištění překročení povolené hranice podobnosti. Toto nastavení musí být vázáno na konkrétní metodu (*pdid*) a úkol (*acaid*).

Po spuštění hledání duplikátů jsou uloženy pouze ty výsledky, u nichž byla překročena povolená hranice podobnosti. Tímto úložištěm je tabulka ***aca_pd_similarity***. Obsahuje údaj o podobnosti a podobném úkolu (*similarity,id_sec*) a váže se na konkrétní úkol (*acaid*), nastavenou metodu (*activepdid*) a odevzdání (*id_pri*). Pro každou metodu a každý úkol je tedy výsledek uložen zvlášť.

7 MODUL ACA

Oproti předchozím řešením (viz. 5.1.1, 5.1.2) bylo již využito nové verze systému Moodle, který obsahuje nový modul Úkol (*Assignment*). Toto řešení sice předpokládá verzi Moodle 1.5.3, ale aktualizace systému není složitou operací. Navíc aktualizace obsahuje bezpečnostní záplaty a opravy chyb předchozích verzí systému.

Nově vytvořený modul byl nazván **ACA** (*automatically checked assignment*). Aby bylo možné tento modul zařadit do systému, musely být všechny názvy funkcí *assignment_XX* nahrazeny *aca_XX*. Dále byly upraveny databázové soubory, jazykové soubory a další. Postup vytvoření nového modulu dobře popisuje dokumentace systému Moodle – Příručka vývojáře.

Při programování modulu ACA byla snaha dodržet programátorský styl podle Průvodce kódováním, jenž je součástí dokumentace Moodle.

7.1 Přidání nového jazyka

Nová verze modulu Úkol je řešena objektovým přístupem. To je výhodné zejména pro přehlednost projektu. Následující odstavec popisuje, jakým způsobem jsou nový jazyk do modulu přidat.

Hlavní skript *lib.php* obsahuje základní třídu *aca_base*. Její metody jsou hlavním jádrem celého modulu. Každý z programovacích jazyků však obsahuje specifické metody. Tyto metody jsou do třídy *aca_base* přidány voláním třídy jako ***extends***.

Třída jazyka je nazvána ***aca_<jazyk>***. Důležité je, že třída jazyka musí obsahovat konstruktor, v němž se odkazuje na rodičovskou třídu *aca_base*.

Příklad pro jazyk Pascal:

```
class aca_pascal extends aca_base {
      function aca_pascal($cmid=0) {
            parent::aca_base($cmid);
      }
      ... další metody třídy
}
```

Hlavní soubor *lib.php* musí z důvodu použitelnosti v systému Moodle obsahovat i některé další funkce, které jsou volány jinými moduly systému při tvorbě, mazání, zálohování a dalších operacích. Tyto funkce musí být umístěny ve skriptu *lib.php* ve formátu:

<název_modulu>_<jméno_funkce>. Aby tyto funkce mohly využít třídu *aca_base*, je v nich využito specifické konstrukce:

```
...
require("$CFG->dirroot/mod/aca/type/$assignment->langtype/aca.class.php")
$acaclass = "aca_$assignment->langtype";
$ass = new $acaclass($mod->id, $assignment, $mod, $course);
return $ass->VOLANA_FUNKCE($user);
...
```

Ve výše uvedené konstrukci využíváme proměnnou *$assignment*, jenž je objekt řádku tabulky *aca* s aktuálně zvoleným úkolem. Pomocí *require* načteme skript pro aktuální jazyk (je vždy v podadresáři s názvem jazyka). Díky znalosti jazyka známe i název třídy *aca_<langtype>*. Zbývá vytvořit novou třídu (která již zná rodičovské metody *aca_base*) a zavolat požadovanou metodu.

Dalším souborem v podadresáři jazyka je *mod.html*. Je volán při vytváření a aktualizaci nového úkolu po klepnutí na tlačítko Další. Na základě zvoleného jazyka skript načte správnou konfigurační stránku (ze správného adresáře).

Přidání modulu by se dalo stručně shrnout do těchto kroků:

1) Vytvoření skriptu *aca.class.php*, obsahujícího třídu *aca_<jazyk>* s konstruktorem, volajícím třídu *aca_base*.
2) Přidání nových metod pro daný jazyk do této třídy.
3) Vytvoření stránky *mod.html* pro nastavení modulu ACA.
4) Vytvoření adresáře ***<jazyk>*** v podadresáři *type* a zkopírování předchozích dvou souborů do tohoto adresáře.

Nejdůležitější je dodržení řetězce ***<jazyk>***. Měl by být pokud možno krátký, bez diakritiky a malými písmeny. Vyhneme se tím možným problémům.

7.2 Důležité metody a skripty

Tato část popisuje pouze důležité metody a význam některých nově vytvořených skriptů modulu. Výchozím dokumentem pro psaní nových modulů je Příručka vývojáře v dokumentaci systému.

Detailní popis metod pro testování funkce a odhalování plagiátů jsou popsány v kapitolách 8 a 9. Veškeré metody jsou ve skriptu *lib.php* nebo *aca.class.php* v příslušném podadresáři jazyka.

7.2.1 Metody pro upload souborů

Základní metoda ***upload()*** pro nahrávání souborů již v modulu Úkol existovala. Navíc byla přidána metoda ***upload_source()***, která slouží pro nahrání učitelského zadání.

Odevzdání úkolu umožňuje vložit pouze jeden soubor. Pokud se program skládá z více souborů, musí být nahrán jako archiv souborů ve formátu ZIP. Pro testování funkce potřebujeme z archivu vyextrahovat jednotlivé soubory. Tento proces obstarává funkce Moodlu ***unzip_files***. Při nahrávání souborů se nyní spouští metoda ***unzip_check***, která kromě rozbalení archivu do pomocného adresáře kopíruje povolené soubory (podle přípony) do prostoru pro odevzdané úkoly. Obsah pomocného adresáře poté vymazán. Nastavení povolených souborů je v proměnné ***$ACACFG->allowext***.

Úpravou prošla také metoda ***file_area_name***. Ta vrací název adresáře, kde se nachází odevzdané úkoly. Pro učitelské vypracování vrací nyní cestu k podadresáři s názvem ***source***.

7.2.2 Metody pro nastavení testovacích hodnot

Každá testovací hodnota je představována jedním záznamem v tabulce *aca_active_tv*. Jak bylo uvedeno v 6.1.1, každá hodnota obsahuje několik sloupců tabulky pro konkrétní nastavení testu. Musí však existovat vazba mezi daty v databázi a uživatelským vstupem. Zároveň u každého testu není nutné využít všechna nastavení a data mohou být reprezentována různými prvky HTML formulářů (textové pole, roletkové menu, checkbox a podobně). Propojením mezi pohledem uživatele a nastavením je metoda ***<jazyk>_tv_setting***. Ta musí být opět součástí skriptu *aca.class.php* pro daný jazyk.

Metoda ***<jazyk>_tv_setting*** je volána pro každou testovací hodnotu. Uživatel toto nastavení vidí jako jeden řádek tabulky, obsahující prvky HTML formulářů. Vstupem jsou data z tabulky *aca_tv_active* pro aktuální hodnotu (parametr ***$tv***). Jelikož je Moodle koncipován jako jazykově nezávislý, funkce musí vracet také záhlaví tabulky. Proto je nutný parametr ***$type***, určující zda požadujeme záhlaví tabulky nebo HTML formulář. Parametr ***$multi*** určuje, jestli je zobrazen formulář pro jednu nebo více testovacích hodnot. V případě více hodnot musíme totiž HTML prvky indexovat jako pole. Výstupem metody

je pole, obsahující buď textová data pro hlavičku tabulky, nebo prvky HTML formuláře. Pořadí dat je závazné, ve stejném pořadí budou data vypsána v tabulce.

Nutným požadavkem pro správnou funkci je dodržení atributu ***name*** u HTML prvků. Ty musí souhlasit s názvy databázových sloupců. V opačném případě data nebudou uložena.

Pro uložení dat z formuláře do databáze slouží metoda ***process_tvs***. Ta pouze projde pole *$_POST* a vyhledává zadaná data, která následně uloží do databáze.

Ukázka stránky s nastavením testovacích hodnot je na obrázku Obr. 8.

7.2.3 Ostatní metody

Dále ještě uvedu tabulku dalších podstatnějších metod, rozdělených podle účelu funkce.

metoda	popis funkce
print_tvs_summary	zobrazení tabulky možných testovacích metod pro úkol
get_new_tv	vytvoření objektu testovací hodnoty pro uložení do DB
new_tv_result	vytvoření objektu výsledku testu pro uložení do DB
display_tv_results	zobrazení výsledků provedených testů a obodování úkolu
grading_module	obodování úkolu na základě výsledků testu

Tab. 2. Další metody pro testování funkce programu

metoda	popis funkce
print_pds_settings	zobrazení tabulky metod pro hledání plagiátů
process_pds	uložení nastavení metod pro hledání plagiátů
create_pd	vytvoření objektu metody pro hledání pro uložení do DB
delete_pd	smazání metody pro hledání z DB
grade_after_pd	obodování úkolu na základě výsledků hledání plagiátů

Tab. 3. Další metody pro kontrolu plagiátů

7.2.4 Skript testvectors.php

Slouží jak k nastavování testovacích hodnot, tak k zobrazování výsledků testů. Obě tyto možnosti poskytuje metoda ***testvectors***. Skript může být spouštěn pouze učitelem kurzu.

Parametry korespondují částečně s funkcí *testvectors*. Důležitý je parametr *mode*, určující v jakém režimu se nacházíme (editace testovacích hodnot nebo zobrazování výsledků).

Parametr *tv* je metoda testování (*ID* z tabulky *aca_tvs*). Parametr *uid* identifikuje uživatele pomocí jeho *ID* z tabulky *users*.

7.2.5 Skripty descriptions.php a similarities.php

Tyto jednoduché skripty slouží pouze ke zobrazování výsledků testů. Jsou přístupné pouze učitelům kurzu. Obsah je volán metodami ***descriptions*** a ***print_similarities*** třídy *aca_base*.

7.2.6 Skript pd4m.make.php

Jedná se o přejatý skript z modulu PD4M (viz. 5.1.2) s drobnými úpravami. Stará se o volání funkcí pro vytvoření tokenů odevzdaných úkolů a spuštění programu YAP (viz. 3.3.3).

Mimo tyto funkce umožňuje také export a import tokenů všech úkolů ve formě ZIP souboru. Všechny funkce jsou začleněny jako metody do třídy *aca_base*. Tato stránka poskytuje pouze rozhraní mezi metodami pro kontrolu a uživatelem (zobrazení formuláře pro upload a podobně).

7.3 Instalace a nastavení modulu

Instalace modulu ACA představuje kopírování souborů do podadresáře pro moduly a následné načtení stránky pro správu systému. Systém Moodle automaticky rozpozná nový modul a provede potřebné změny v databázi.

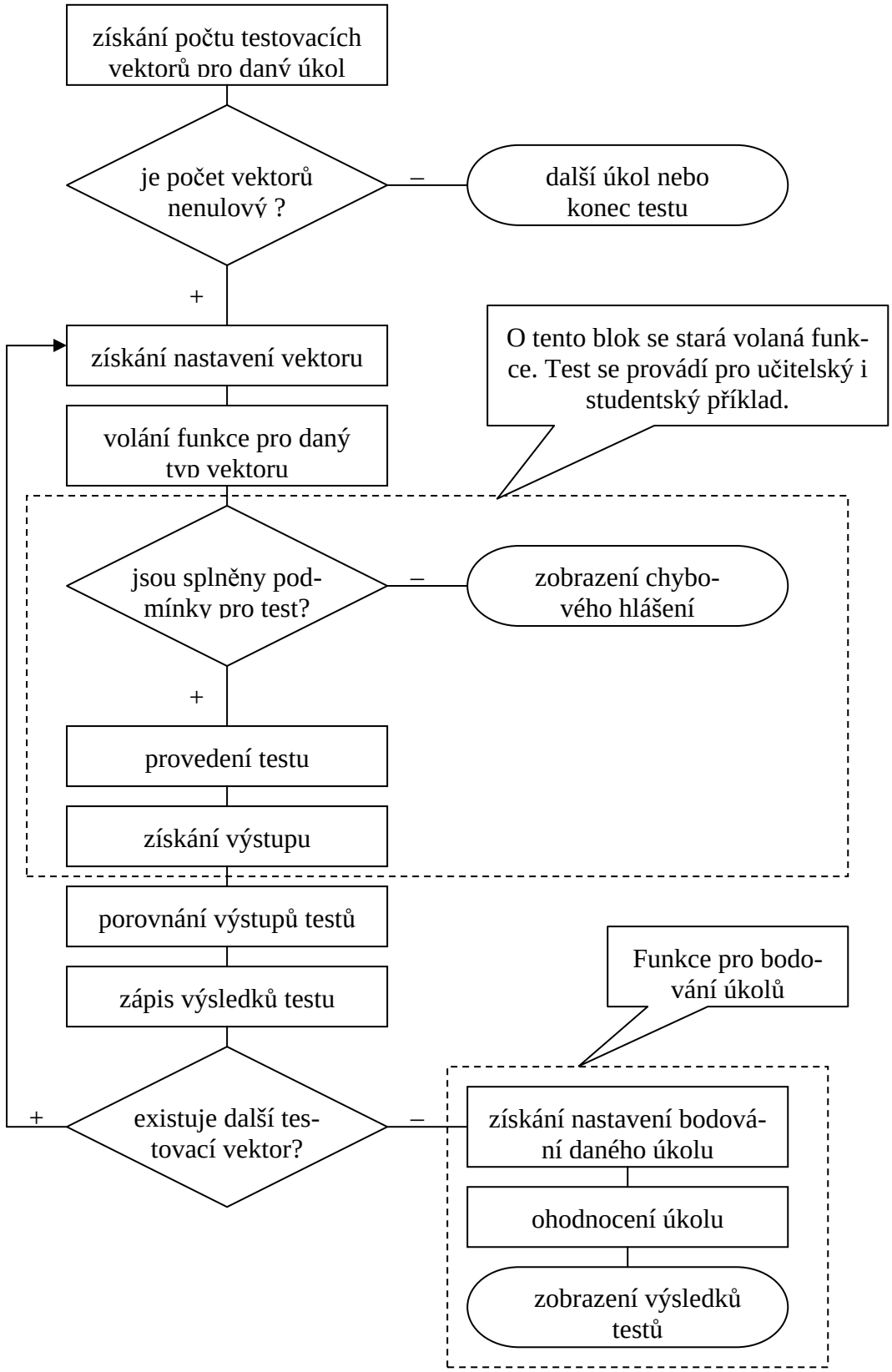
Modul však obsahuje také proměnné závislé použitém operačním systému. Tyto proměnné jsou uvedeny na začátku skriptu lib.php a jsou uloženy v objektu ***$ACACFG***. Tyto proměnné jsou pracovní adresáře nástrojů, které modul ACA využívá.

Dalšími proměnnými jsou povolené přípony souborů pro každý z jazyků. Tato nastavení jsou opět uložena v objektu ***$ACACFG*** a uvedena na začátku každého ze skriptů *aca.class.php* pro každý z jazyků.

Poslední nutnou operací je kopírování nových adresářů. Jedná se o dočasné adresáře a součásti programu YAP. Tyto pracovní adresáře musí odpovídat nastaveným proměnným *$ACACFG* ve skriptu *lib.php*.

Všechny tyto adresáře jsou součástí přiloženého CD. Obsahuje ukázkovou instalaci modulu ACA v systému Moodle včetně dalších nástrojů a adresářů potřebných pro funkci modulu. CD obsahuje také několik testovacích příkladů.

8 TESTOVÁNÍ FUNKCE PROGRAMU



Obr. 4. Schéma procesu testování programu pro jeden odevzdaný úkol

Jak můžeme vidět z obrázku (Obr. 4), testování funkce probíhá v několika fázích. Modul umožňuje testovat jak konkrétní odevzdaný úkol (*submission*), tak všechny odevzdané úkoly. Budeme tedy postupovat od elementárních metod pro testování až ke globálním.

Pro každý programovací jazyk může existovat více metod pro testování. Vykonání jednoho testu provedeme metodou ***<jazyk>_tv_testing***. Tato metoda je napsána v každém ze skriptů *aca.class.php* pro daný jazyk.

Prvním parametrem je ***$tv***, jako objekt z tabulky *aca_tv_active* obsahující právě testovací hodnoty. Druhým parametrem je ***$submission*** identifikující odevzdaný úkol. Uvnitř metody je skrytý proces testování. K určení použité metody testování slouží příkaz *switch*. Metoda je identifikována pomocí svého *ID* z tabulky *aca_tvs*.

Nadřazená metoda ***tv_testing*** provádí všechny nastavené testovací hodnoty pro jeden odevzdaný úkol. Nad touto metodou stojí metoda ***process_test_all***, která provede všechny nastavené testy pro všechny odevzdané úkoly.

Aby bylo možné výsledky testů zobrazit studentovi ihned po odevzdání úkolu, provádí se metoda ***tv_testing*** ihned po uploadu souborů. Zobrazení výsledků se pak řídí nastavením u každého z úkolů. Test všech odevzdaných úkolů je učiteli dostupný pro případ, že by změnil nastavení testovacích hodnot – tím by se samozřejmě změnily i výstupy. Učitel může také provést test u vybraného úkolu.

Z bezpečnostních důvodů je spouštění prováděno pomocí skriptu ***myexec***. Ten zajišťuje omezení běhu programu na stanovenou dobu a zápis výstupu a chybového výstupu. Pokud se program do stanovené doby neukončí, je ukončen příkazem *kill*.

Spouštění programů ve virtuálním systému prozatím není plně funkční. Do jisté míry totiž závisí na použité distribuci a verzi operačního systému. Zdrojové kódy by bylo potřeba upravit pouze při spouštění testů, které jsou popsány níže.

8.1 Jazyk C

Testování funkce programu v jazyce C/C++ vychází ze zkušeností s předchozí práce (viz. 5.1.1). V současné době existuje pouze jedna metoda testu.

Pro spuštění programu pomocí interpretu CH je vytvořen spouštěcí soubor, do kterého jsou pomocí direktivy ***#pragma import*** vloženy všechny soubory jazyka C, uvedené

v proměnné *$ACACFG->includeext*. Spouštění probíhá v testovacím podadresáři (*sourcetest*), kam jsou kopírovány vždy učitelské a poté studentské soubory.

Následně je soubor spuštěn i s parametry uvedenými v nastavení testovací hodnoty. Výstupy z programu jsou následně porovnány a zapsány do výsledků testu.

8.2 Jazyk Pascal

Jak byl uvedeno v analýze (viz. 5.2.1), nepodařilo se najít interpret jazyka Pascal, který by spolehlivě fungoval. Současná metoda testu využívá kompileru ***FreePascal***, který je na systémech Linux velmi rozšířený.

Při testování je tedy nejprve program zkompilován. Pokud kompilace proběhne úspěšně, je vytvořen spustitelný soubor. Pokud kompiler najde chybu, je zapsána jako výsledek testu. Po úspěšné kompilaci je program spuštěn s nastavenými parametry podobně jako program v jazyce C/C++. Testování také probíhá v adresáři *sourcetest* a výstupy jsou zapsány do výsledků testu.

8.3 Jazyk PHP

Tento jazyk je narozdíl od předchozích dvou interpretovaný a jeho syntaxe je „volnější“. Například není nutné definovat typ proměnných ani proměnné deklarovat. Skripty se do sebe dají jednoduše vnořovat pomocí klíčového slova ***#include***. Dále lze simulovat vložení superglobálních proměnných jakoby byly poslány z formuláře. Tedy částečně simulovat interakci uživatele vložením hodnot do pole *$_POST* a *$_GET*. Po stránce testování je jazyk PHP velmi flexibilním a byly pro něj napsány 2 testovací metody.

8.3.1 Test obsahu proměnných

V této variantě jsou testovány pouze určité proměnné. Výstup programu nás nezajímá. Je sestaven pomocný testovací skript, do kterého jsou vloženy vstupní proměnné. Následně je pomocí ***#include*** vložen samotný testovaný skript. Ještě před vložením však potlačíme výpis na standardní výstup funkcí PHP ***ob_start()***. Tato funkce zajistí, že výstup bude bufferován až do volání funkce PHP ***ob_flush()***. Obsah proměnné můžeme jednoduše získat voláním funkce PHP ***var_dump()***. Ta vypisuje jak použitý typ proměnné, tak jeho hodnotu. Výsledný testovací skript tedy vypadá následovně:

```
<?php //generated testing PHP file. Do not edit.
      ob_start();   // start buffering
      // BEGIN input variables from test vector
      $promenna = 'hodnota';
      // END input variables from test vector
      include 'testovanyskript.php'; // auto included file
      ob_end_clean();   // clean buffer
      // automatic added testing variables output
      var_dump($testovanapromenna);
?>
```

Výstupem tohoto testovacího skriptu je tedy obsah proměnné vrácený funkcí ***var_dump***.

8.3.2 Test výstupu skriptu

Tato metoda je jednodušší variantou. Je užitečná hlavně u příkladů, kde student má za úkol generovat například dynamickou stránku. Výstupem je tedy například HTML kód. Postup při testování je obdobný jako u výše uvedené metody, avšak není zapotřebí bufferovat výstup. Do skriptu tedy pouze vložíme vstupní proměnné a spustíme interpret PHP na testovací skript.

9 ODHALOVÁNÍ PLAGIÁTŮ

Z výsledků analýzy (5.1.2) vyplynulo jednoznačně, že nejlepším a nejjednodušším řešením je implementace programu YAP. Program byl úspěšně použit pro programovací jazyky Pascal a C/C++. Bez dalších úprav je tedy možné YAP na tyto jazyky použít. Novým jazykem je PHP, pro který prozatím neexistuje tokenizer (viz. 3.3.3). Jeho vytvořením je vše připraveno k použití programu YAP i pro jazyk PHP.

Součást pro odhalování plagiátů je možným doplňkem úkolu, ale zároveň je vše připraveno pro přidání jiné metody pro odhalování. V současném stavu je pro všechny jazyky vytvořena pouze metoda testování programem YAP3.

9.1.1 Přípravná fáze

Odhalení plagiátů programem YAP zahrnuje dvě fáze. První je tokenizace, druhou potom porovnání tokenů. V praxi však nemá smysl snažit se odhalit plagiáty už ve fázi odevzdávání úkolů. Lepším řešením je počkat na odevzdání všech úkolů a poté pouze porovnat podobnosti mezi nimi.

Přípravnou fází je tedy vytvoření tokenů pro každý z úkolů. Vytvoření probíhá ihned po odevzdání úkolu voláním metody ***create_tokens***. Tokeny jsou vytvořeny v pomocném adresáři nastaveném v proměnné ***$ACACFG->newpath***. Každý úkol je zda zastoupen souborem ***<číslo_studenta>.tokens***. Procesem tokenizace procházejí pouze soubory s příponou uvedenou v proměnné ***$ACACFG->tokenizeext***.

Nadřazenou metodou ***preprocess_pd_files*** jsou spouštěny všechny podobné přípravné fáze nastavených metod testování. Tím je do budoucna umožněno rozšíření o nové metody odhalování plagiátů.

9.1.2 Hledání plagiátů

Na žádost učitele (kliknutím na tlačítko) může být spuštěn proces hledání podobností mezi odevzdanými úkoly. Tento proces zastupuje metoda ***ExecYap***. Při bližším pohledu zjistíme, že pouze spouští *bash* skript a poté volá metodu ***LoadYapSummary***.

Spouštěný skript ***check2*** pracuje následovně:

- V adresáři s vytvořenými tokeny pro zadaný úkol vyhledá všechny soubory s příponou *tokens* a zapíše je do souboru *yap.submissions*.
- Projde postupně přes všechny soubory s příponou *tokens*, každý z nich zapíše do souboru *yap.newfiles* a spustí hledání podobných souborů programem ***rkr_gst*** (viz. 3.3.3).

Výsledkem této činnosti je soubor *yap.summary*. Každý soubor tokenů je porovnán se všemi ostatními soubory a je zjištěna jejich podobnost.

Metoda ***LoadYapSummary*** pracuje s výsledky porovnaných souborů. Načte soubor *yap.summary* a řádek po řádku jej testuje, zda u některého z úkolů byla překročena povolená hranice podobnosti. Pokud takový soubor najde, zapíše výsledek do tabulky *aca_pd_similarities*. Při zjištění vyšší než povolené podobnosti je volána metoda ***grade_after_pd***, která provádí automatické hodnocení úkolu podle nastavení učitele.

Nakonec je ještě zapotřebí zobrazit výsledky nalezených podobností. Metoda ***print_similarities*** zobrazuje tyto výsledky ve dvou variantách. Buď jako tabulku, nebo jako pouhý text. Textová forma je využita pro vyplnění odpovědí učitele na odevzdaný úkol.

Uvedený popis by opět fungoval jen pro metody YAP. Ve skutečnosti se spouští při testu metoda ***process_all_pds***, která spustí všechny nastavené metody hledání plagiátů.

10 UŽIVATELSKÉ ROZHRANÍ

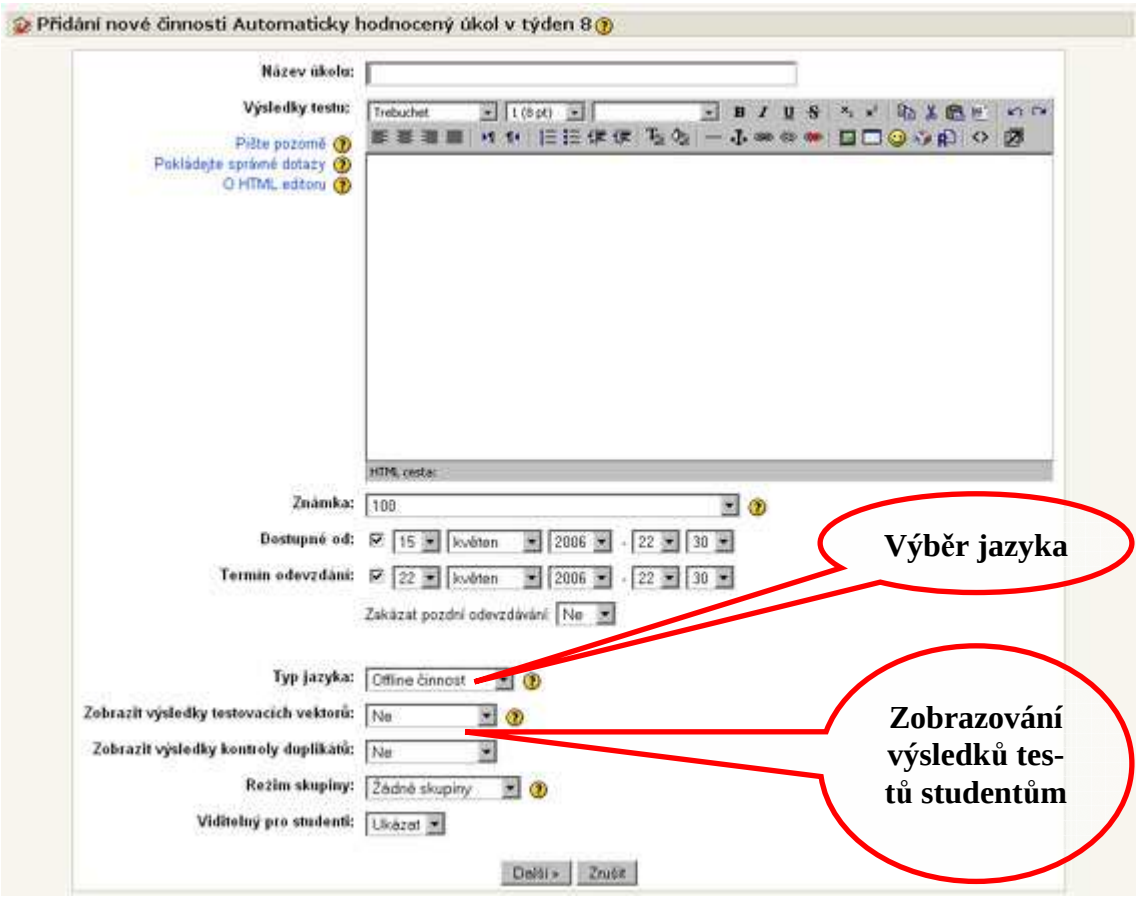
Jedna z hlavních výhod systému Moodle je, že existuje v mnoha jazykových mutacích. Pro každý jazyk jsou přeloženy veškeré jazykově závislé texty a stránky nápovědy. Pro modul ACA byly vytvořeny jazykové soubory pro češtinu a angličtinu. Následují ukázky uživatelského rozhraní pro český překlad.

Další vlastností systému Moodle je možnost volby motivů, čili vzhledu stránek. V následujících ukázkách je použito téma s názvem ***formal_white***, které je standardní součástí distribuce Moodle 1.5.3.

10.1 Rozhraní učitele

10.1.1 Založení a úprava nového úkolu

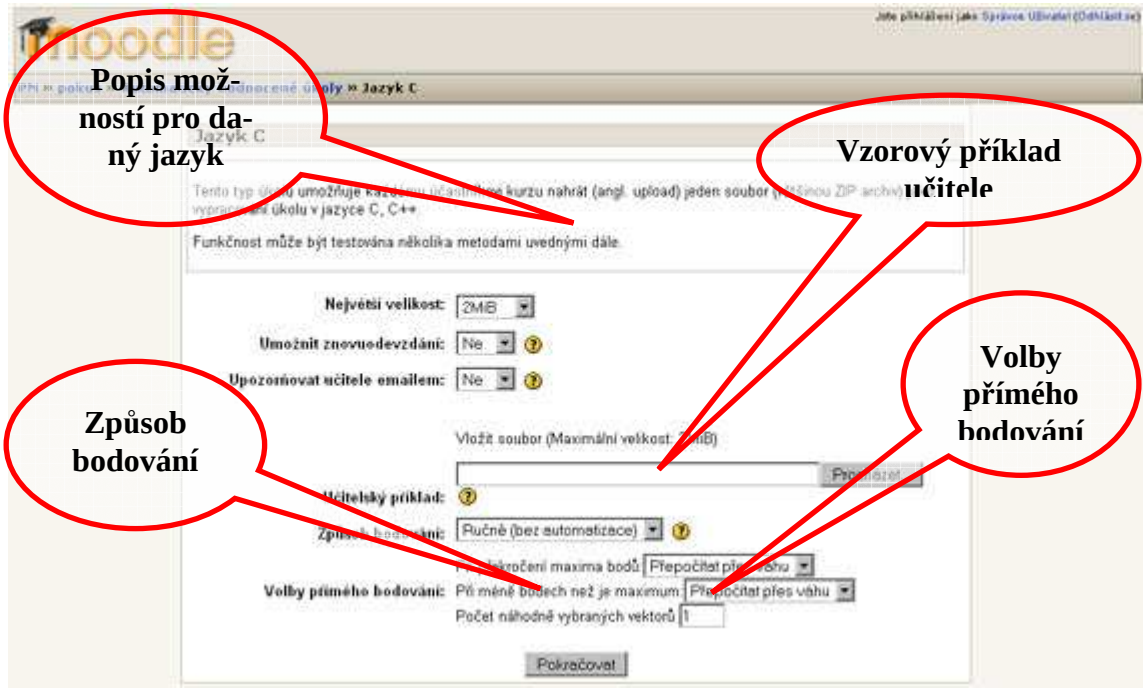
Základem každého úkolu je založení nové činnosti. Provádí se obvyklým přidáním nové aktivity v kurzu. První stránka nastavení je univerzální pro všechny programovací jazyky.



Obr. 5. Založení nové činnosti Automaticky hodnocený úkol

V současné verzi modulu jsou na výběr jazyky C/C++, Pascal a PHP. Nadále je možné zvolit také typ Odevzdat soubor a Offline činnost (jako v modulu Úkol).

Možné volby zobrazování výsledků testů studentům jsou: Ano, Ne a Pouze rozdílné (výstupy odevzdání studenta a výstupy příkladu učitele se liší).



Obr. 6. Upřesňující nastavení pro programovací jazyk

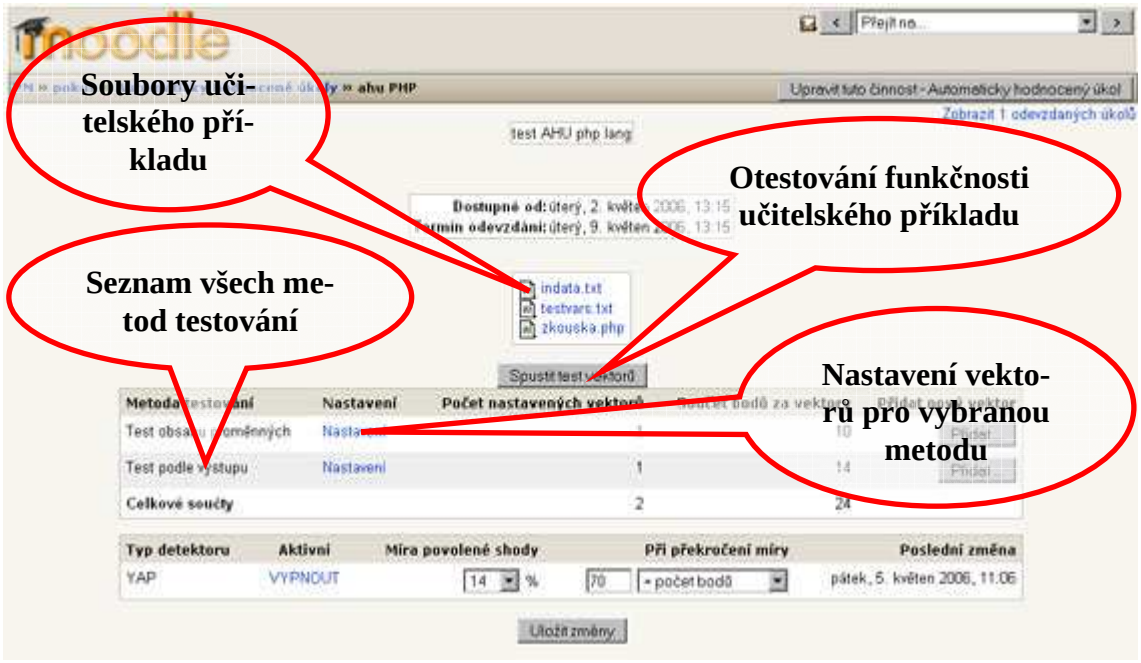
Po výběru programovacího jazyka je načtena stránka s upřesňujícím nastavením pro úkol. V záhlaví je uveden stručný popis, co vybraný submodul umožňuje. Dále se od učitele očekává vzorové vypracování úkolu (samostatný soubor nebo archiv ZIP). Poslední novou volbou je způsob bodování. Bližší popis je v souboru s nápovědou. Možnosti bodování budou více pochopitelné poté, co zjistíte jakým způsobem se nastavují testovací hodnoty.

Veškerá nastavení lze po prostudování způsobu kdykoliv upravit. Soubory vzorového vypracování jsou mazány pouze v případě, vložení nového souboru. Pokud je pole prázdné, stávající soubory zůstávají nezměněny.

10.1.2 Nastavení testovacích hodnot

Po založení nového úkolu je možné přiřadit libovolný počet testovacích hodnot. Každý jazyk může obsahovat více metod testování.

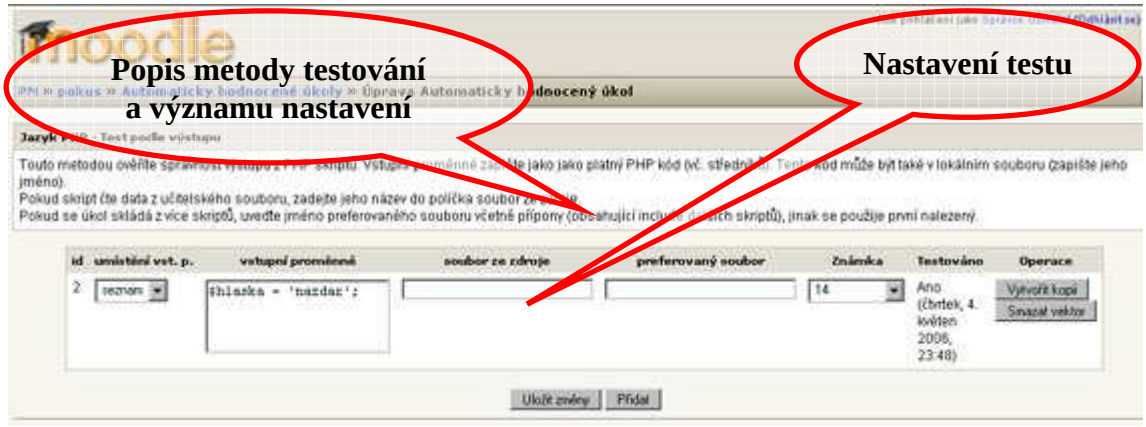
Nastavení testovacích hodnot však stále zůstává pouze **možností**, nikoliv nutností. V případě že nejsou využity testovací hodnoty, poskytuje modul pouze funkci rozbalení archivů a filtraci souborů v archivu.



Obr. 7. Hlavní stránka pro nastavení testů

První tabulka zobrazuje všechny možné metody testování pro vybraný jazyk. Pro každou metodu je zobrazen počet aktuálně nastavených testovacích hodnot a součet bodů za ně. Po kliknutí na odkaz Nastavení je možné testovací hodnoty upravovat, mazat a přidávat. Přidání nové testovací hodnoty usnadňuje také tlačítko pro přidání umístěné vpravo.

Ještě předtím, než budou testovány studentské příklady, je vhodné vždy ověřit funkčnost nastavených testů. Při tomto testu je sám proti sobě testován učitelský příklad pro všechny nastavené testovací hodnoty. Výsledek testu je poté zobrazen v tabulce.

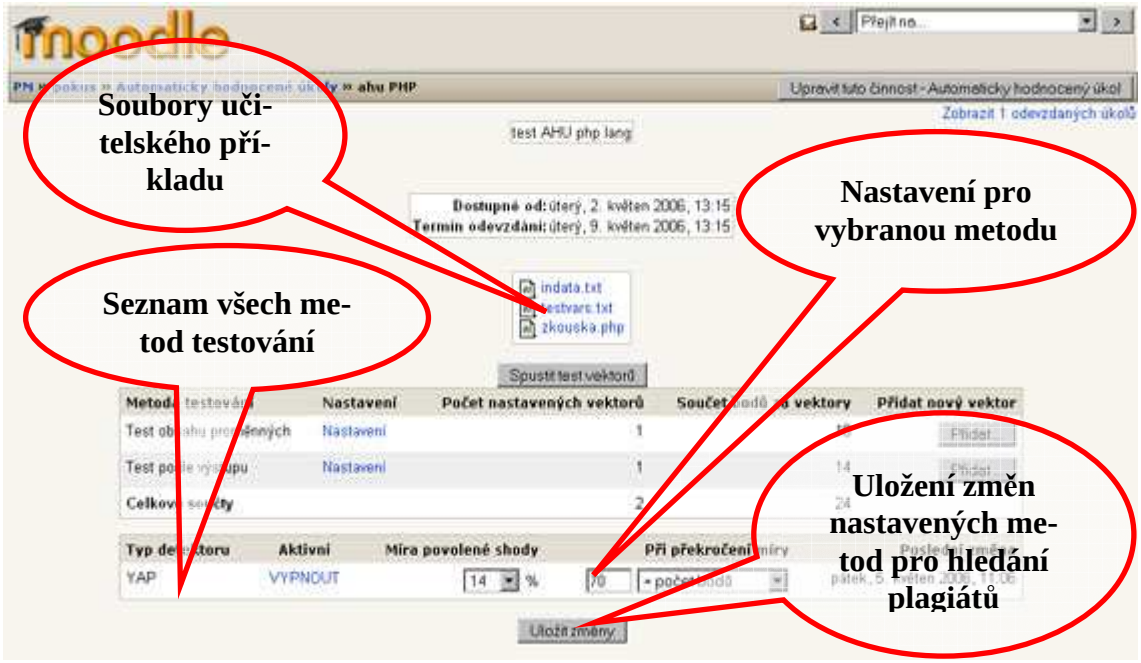


Obr. 8. Nastavení jedné testovací hodnoty

U nastavení testů je vždy uvedeno jakým způsoben test probíhá a jaké hodnoty jsou očekávány v jednotlivých položkách pro nastavení. Zároveň je možné vytvořit kopii vybrané testovací hodnoty nebo ji smazat. Smazáním dojde také ke smazání případných provedených výsledků pro danou testovací hodnotu. Důležitým údajem je Známka. Tato položka souvisí s metodou bodování úkolu. V případě rovnoměrného rozložení nebo náhodného výběru testovacích hodnot nemá význam, ale v případě Přímých bodů a váhové funkce je tato hodnota klíčová. Formulář obsahuje ještě tlačítko pro uložení změn a přidání nové testovací hodnoty.

10.1.3 Nastavení metody pro hledání plagiátů

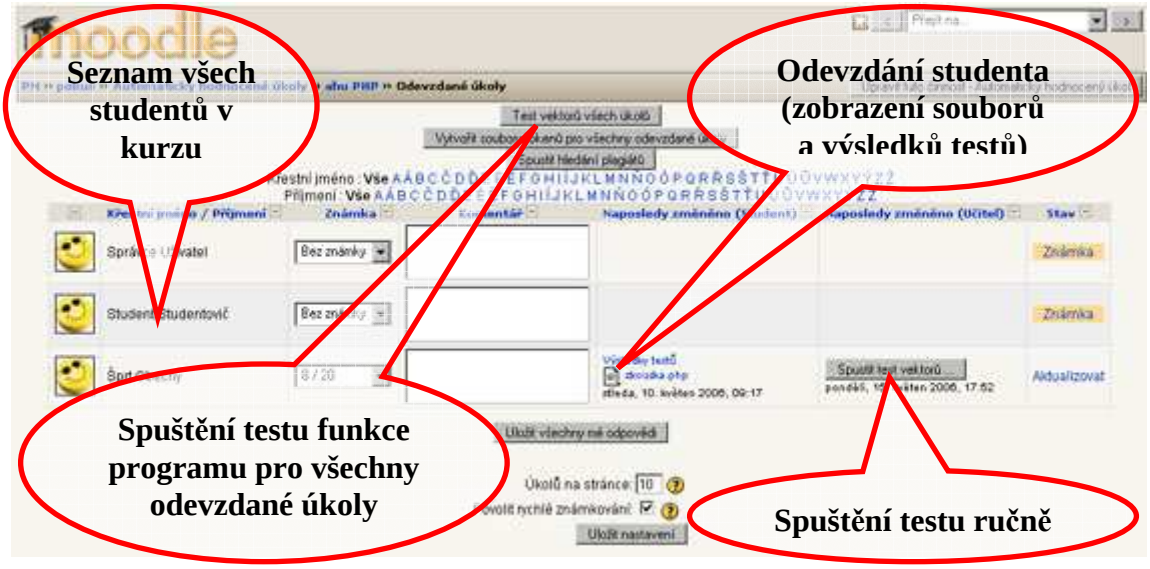
Obdobně jako nastavení metod testování je zobrazena i druhá tabulka metod pro hledání plagiátů.



Obr. 9. Nastavení metod pro hledání plagiátů

Ke každému úkolu lze různé metody ZAPNOUT nebo VYPNOUT. Po zapnutí je možné nastavit parametry. Důležitá je Míra povolené shody. Nad tuto hranici je výsledek testu brán jako plagiát. Dále je možné nastavit automatické hodnocení úkolu, u kterého byla překročena povolená míra shody.

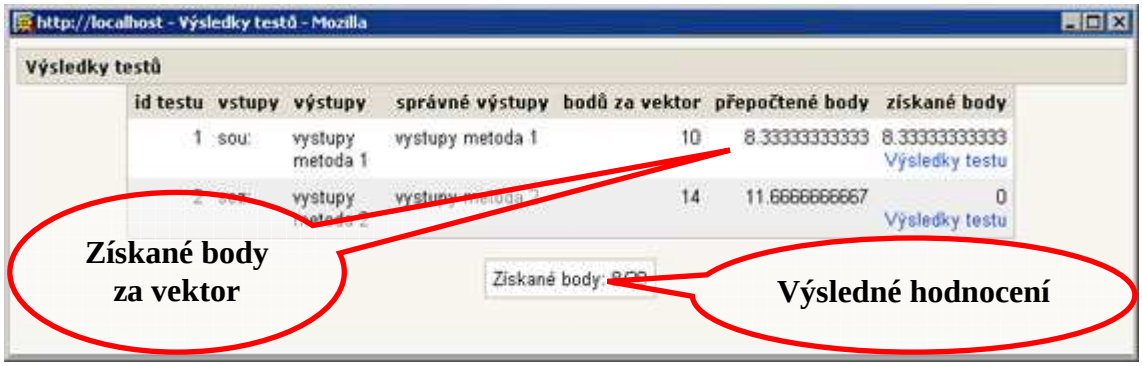
10.1.4 Zobrazení odevzdaných úkolů



Obr. 10. Hodnocení studentských úkolů

Všechny odevzdané úkoly studentů jsou dostupné na jedné stránce. Pokud je povoleno Rychlé známkování, může učitel hodnotit a psát komentáře k více úkolům zároveň. Pokud

jsou nastaveny testovací hodnoty, jsou zobrazeny jejich výsledky a je možné provést test ručně jak pro jeden úkol, tak pro všechny odevzdané úkoly.

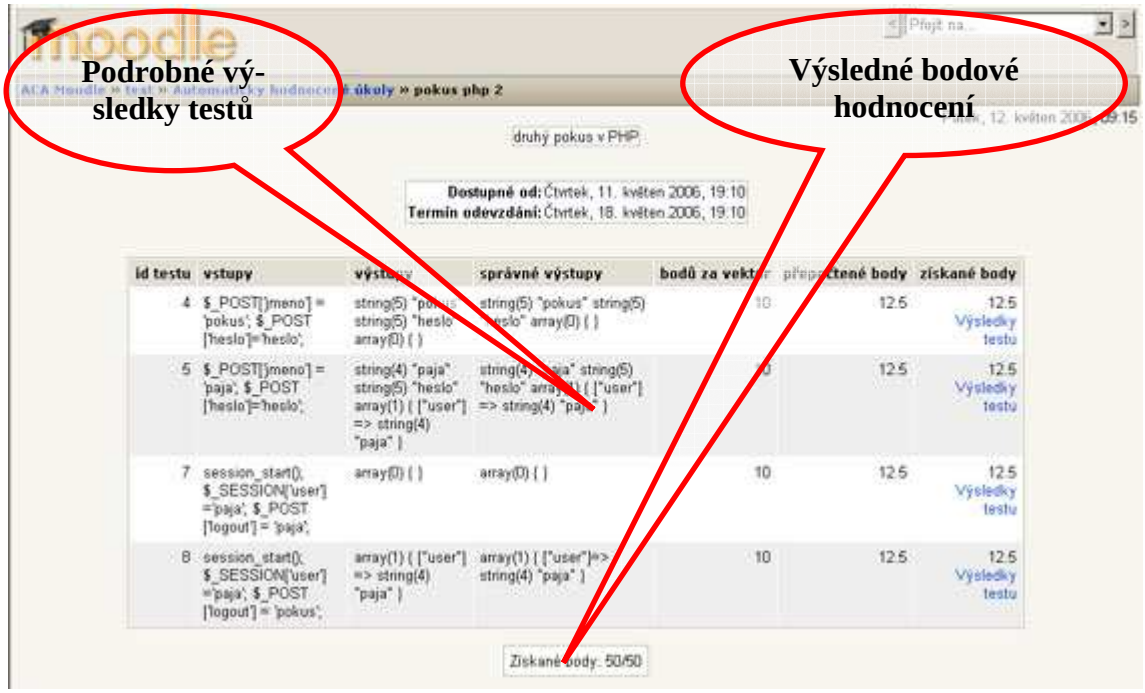


Obr. 11. Výsledky testovacích hodnot a získaných bodů

Výsledky testů se otevírají v samostatném okně. Mimo vstupní data a výstupy z programu studenta a učitele jsou zobrazeny také zjištěné rozdíly výstupů a získané body za každou z testovacích hodnot, přepočtené podle typu bodování.

10.2 Rozhraní studenta

Činnost studentů zahrnuje pouze odevzdání úkolu a případné zobrazení výsledků testů, pokud jej učitel povolí. Jedinou změnou, kterou studenti v novém modulu mohou vidět je tabulka výsledků testů.



Obr. 12. Výsledek testů odevzadného úkolu

11 MOŽNOSTI ROZŠÍŘENÍ

Cílem práce bylo vytvořit moduly pro systém Moodle, které by umožnily kontrolu úkolů v jiných jazycích než C/C++. Po analýze možných řešení vznikl pouze jeden modul, který ovšem plní funkci modulu univerzálního.

11.1.1 Přidání nové metody pro testování

Nový modul ACA je prvním krokem k integraci funkcí pro testování programů do jednoho modulu. Přidávání nových metod testování není prozatím z hlediska programátora příliš jednoduché. Na stranu druhou vytvoření nové testovací metody vyžaduje jisté znalosti jazyka *PHP*, operačního systému *Linux* a samozřejmě i příslušného programovacího jazyka, pro který testovací funkci vytváříme.

Ještě před vytvořením je třeba analyzovat daný problém. Určit, zda je nutné program kompilovat nebo interpretovat, jaké jsou možnosti jeho otestování a vzít ohled na bezpečné spuštění a ošetření chybových stavů testu. Východiskem musí být záměr testu, tedy jaké typy úkolů studenti dostanou a co bude cílem úkolu.

Po analýze víme, jakým způsobem lze úkol otestovat. Pro nastavení parametrů testu slouží několik datových sloupců z tabulky ***aca_tv_active***. Tyto sloupce jsou: ***test_opt, test_val*** a dále ***in_opt, in_string a in_text*** a popis jejich datových typů je v příloze P II. Obecně řečeno hodnoty *test_opt* a *in_opt* by měly určovat jakým způsobem bude s hodnotami zacházeno (načtení souboru, předání parametru při spuštění atd.). Samotnými nosiči nastavených hodnot (například parametr při spuštění, proměnná atd.) jsou pak sloupce *test_val*, *in_string* a *in_text*.

Dalším krokem je přidání nové metody do tabulky ***aca_tvs***. Důležité je správně přiřadit metodu k programovacímu jazyku. Název submodulu pro jazyk musí být shodný s řetězcem ve sloupci ***langtype***. Název metody je libovolný, sloupec *timemodified* je pouze orientační. Měl by spíše umožnit přehled o případné změně verze testovací metody. Po vložení nové metody musíme zjistit hodnotu sloupce ***ID*** nově přidané metody. Tato hodnota je klíčová pro další postup.

Pro vybraný jazyk nyní upravíme metodu ***<jazyk>_tv_testing*** (viz. 8). Každá tato metoda musí obsahovat příkaz ***switch ($tv->tvid)*** pro určení metody testu. Novou metodu zatupuje číslo *ID* z databáze. Všechny nutné operace pro otestování jsou obsahem jedné větve toho-

to příkazu. Návratovou hodnotou funkce *<jazyk>_tv_testing* je objekt ***$result***. Ten představuje data v tabulce *aca_tv_results*. Každý test musí poskytnout na výstupu tyto hodnoty:

proměnná	význam
$result->indata	vstupní data testu
$result->teacher_out	výstup z učitelského programu
$result->student_out	výstup ze studentského programu
$result->result	výsledek testu (0 – neúspěšný, 1 – úspěšný)
$result->description	detailní popis výsledku testu

Tab. 4. Výstupní proměnné funkce pro testování

Další úpravou musí projít také metoda ***<langtype>_tv_setting***, popsaná také v 7.2.2. Je použita stejná logika jako u předchozí funkce. Tedy rozdělení příkazem *switch* podle metody testování. Návratovou hodnotou funkce je pole, zobrazené později jako záhlaví nebo obsah buněk tabulky (podle předaného parametru *$type*). Podstatnou částí je zde především provázanost jména prvku HTML formuláře a nastavované proměnné. Rovněž je vhodné pro některé HTML prvky využít funkcí systému Moodle ze skriptu *weblib.php*.

Poslední nutnou úpravu musíme provést v jazykovém souboru pro modul ACA, která se nachází v souboru: *lang/<zkratka_jazyka>/aca.php*.

Další podrobnosti jsou patrné po prostudování již realizovaných funkcí pro testování, případně ze šablony na přiloženém CD.

11.1.2 Nové návrhy na testování funkce programu

Návrh a tvorba základní funkce nového modulu byla časově náročnou operací. Proto nebylo možné realizovat všechny další možnosti rozšíření. Stručný přehled návrhů je zveřejněn v této kapitole.

Testovací hodnoty by bylo možné nastavovat například pomocí oboru hodnot (číselná řada, náhodný řetězec, náhodně vybraná hodnota z množiny hodnot). Pro takovou definici vstupní hodnoty by bylo nutné vytvořit „pseudojazyk“. Učitelský i studentský program by byly otestovány vždy stejnými vstupy a musely by dávat stejné výstupy. Tímto řešením bychom předešli možným podvodům studentů sestavit program tak, aby jen na požadované vstupy „správně odpověděl“.

Testování funkce programu je ve většině případů omezeno definicí vstupů jako parametrů při startu programu. Bylo by vhodné realizovat například emulátor stisku kláves, který by simuloval interakci s uživatelem. Podstatně by to zvýšilo užitnou hodnotu tohoto modulu. Podobnou funkci prozatím v omezené míře umožňuje kontrola pro jazyk PHP, což ovšem vychází z vlastností jazyka samotného.

Jednou z hlavních výhod systému Moodle je nezávislost na platformě. Ačkoliv cílem této práce bylo vytvoření modulů pro potřeby výuky na UTB, bylo by přínosné, aby modul byl nezávislý na platformě a také aby jeho instalace probíhala stejně jednoduše, jako instalace ostatních modulů systému.

ZÁVĚR

Cílem práce bylo vytvoření nových modulů pro systém Moodle, který je na UTB ve Zlíně používán při výuce v základních kurzech programování. Výchozími body byly moduly vzniklé v minulém roce. První slouží pro automatickou kontrolu programů v jazyce C/C++ a druhý pro odhalování plagiátorství zdrojových kódů. Práce spočívala ve vytvoření podobné podpory dalších programovacích jazyků, které jsou na UTB vyučovány.

Po analýze současného stavu a možných řešení vznikl pouze jeden modul, který však umožňuje další rozšíření o nové programovací jazyky. Současně integruje možnost kontroly funkčnosti programu i odhalování plagiátorství. Tato možnost doposud chyběla. Stávající verze umožňuje podporu pro základní kurzy programování v jazycích C/C++, Pascal a PHP. Další jazyky, jako například *Matlab* nebo *Mathematica*, prozatím neumožňují automaticky testovat funkčnost programu bez zásahu uživatele. Proto jejich podpora v současné verzi modulu neexistuje.

Součást pro testování funkčnosti pracuje na principu černé skříňky. Vyžaduje vypracování učitelského programu, který je brán jako referenční. Na vstup obou programů přivedeme shodné testovací hodnoty a porovnáme rozdíly na jejich výstupu. Modul umožňuje definovat libovolné množství testovacích hodnot. Výsledné hodnocení tak může být výsledkem provedených testů (student dostane tolik bodů, kolik si „zaslouží“).

Druhý funkční blok využívá systému YAP3 (viz. 3.3.3) pro odhalování podobností mezi zdrojovými kódy. Tento systém pracuje na principu porovnání struktury zdrojového kódu a odhaluje tak většinu běžných postupů studentů při vytváření plagiátů.

Rozšiřitelnost modulu je reálnou vizí. V této práci jsou podrobně popsány postupy vedoucí k podpoře nového programovacího jazyka, vytvoření nové metody testování funkčnosti nebo odhalování plagiátů.

Současná verze je prvním krokem k integraci metod testování a podporu více programovacích jazyků do jednoho modulu. V průběhu realizace i samotného testování vzniklo několik návrhů na zlepšení, které jsou uvedeny v kapitole 11.1.2. Nový modul by se po odstranění všech nedostatků a důkladném otestování mohl stát plnohodnotnou pomůckou pro široké spektrum uživatelů systému Moodle po celém světě.

SEZNAM POUŽITÉ LITERATURY

[1] *MOODLE.CZ* [online]. [cit. 2006-05-12]. Dostupný z WWW: <http://www.moodle.cz / >.

[2] LEACH, Ronald J.: *Using Metrics to Evaluate Student Programs*, SIGSCE Bulletin, vol. 27 No. 2, Červen 1995, s. 41-43.

[3] *Systém Moss* [online]. [cit. 2006-04-20]. Dostupný z WWW: <http://www.cs.berkeley.edu/~aiken/moss.html>

[4] WISE, Michael J.: *Running Karp-Rabin and Greedy String Tiling*, technical report number 463, 1993 [online]. Dostupný z WWW: <http://www.it.usyd.edu.au/research/tr/tr463.pdf>

[5] WISE, Michael J.: *String similarity via Greedy String Tiling and Running Karp-Rabin matching*, 1993, Dostupný z WWW: <http://www.bio.cam.ac.uk/~mw263/ftp/doc/RKR_GST.ps>

[6] WISE, Michael J.: *YAP3: Improved detection of similarities in computer program and other texts* [online]. [cit. 2006-04-22] Dostupný z WWW: <http://www.bio.cam.ac.uk/~mw263/ftp/doc/yap3.ps>

[7] SIEVER, Ellen a kolektiv O'Reilly & Associates, Inc.: *Linux v kostce*. 1. vyd. Praha Computer Press, 1999. 560 s. ISBN 80-7226-227-0.

[8] VAŠKOVÁ, Veronika. *Automatické hodnocení úkolů v kurzech programování*. [s.l.], 2005. 25 s. Vedoucí bakalářské práce Tomáš Dulík.

[9] BAROŠ, Jiří. *Systém pro automatickou detekci plagiátorství v kurzech programování*. [s.l.], 2005. 55 s. Vedoucí bakalářské práce Tomáš Dulík.

[10] Dagwood.: *Technická dokumentácia produktu*, [online]. [cit. 2006-05-15]. Dostupný z WWW: <http://www2.dcs.elf.stuba.sk/TeamProject/2003/team10/docs/Dagwood.System.e3.pdf>

[11] Seagles.: *Finálna projektová dokumentácia*, [online]. [cit. 2006-05-15]. Dostupný z WWW: <http://www2.dcs.elf.stuba.sk/TeamProject/2003/team11/files/tim11-timovy_doc-final.pdf>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

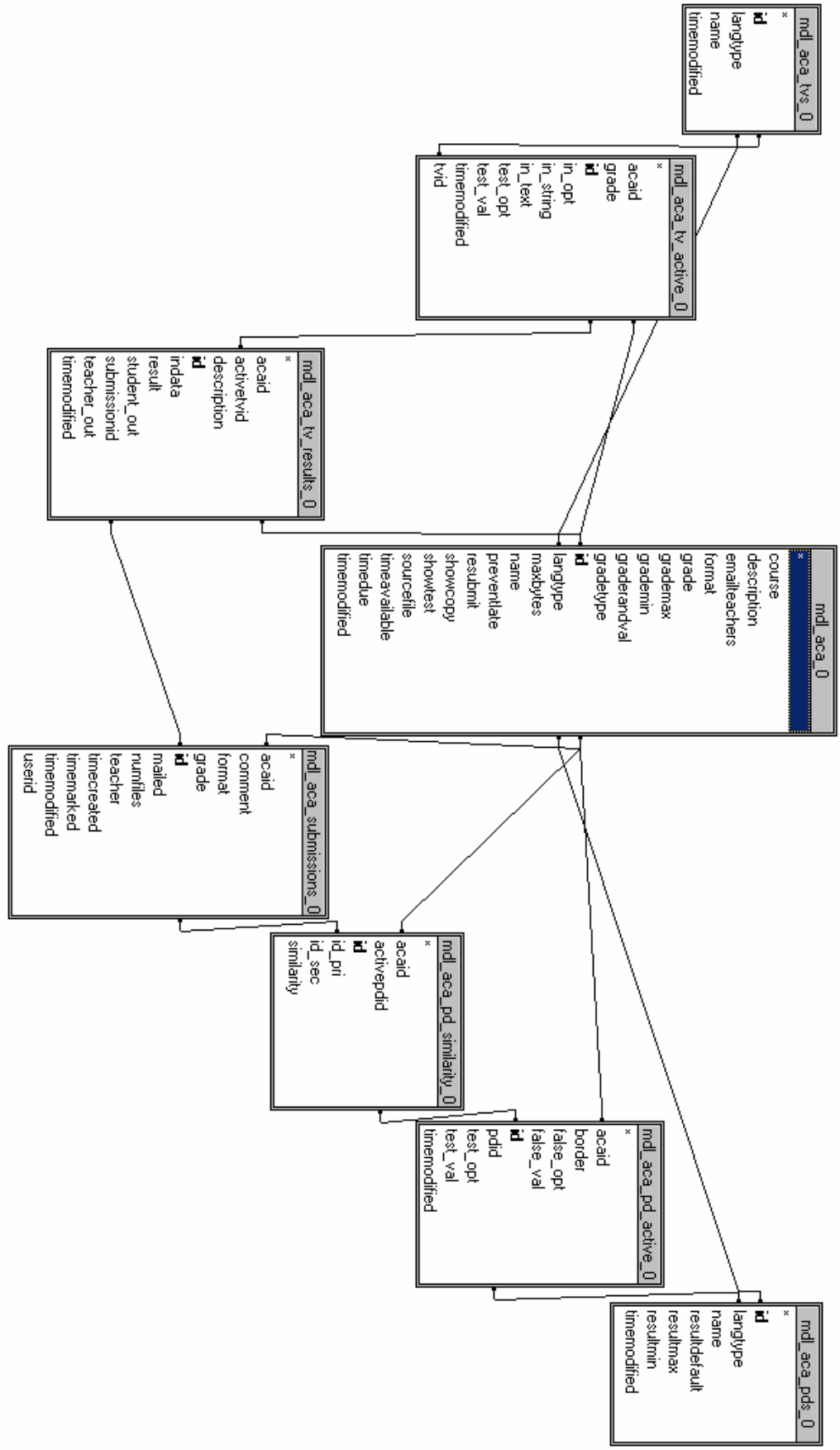
ACA	Automatically Checked Assignment – automaticky hodnocený úkol; název nového modulu pro systém Moodle.
Awk	Utilita pro zpracování textu, jazyk textových manipulací.
bash	Bourne Again Shell – standardní interpret příkazů pro systémy UNIX, Linux.
GNU	GNU's Not Unix – projekt zaměřený na svobodný software, inspirovaný operačními systémy unixového typu.
HTML	Hyper-Text Markup Language – jazyk pro popis dokumentu.
ID	Při použití v databázových tabulkách obvykle označuje unikátní identifikátor řádku tabulky.
LISP	LISt Processing (zpracování seznamů) –funkcionální programovací jazyk.
LMS	Learning Management System (systém pro řízení výuky) – aplikace řešící administrativu a organizaci výuky v rámci eLearningu.
MOSS	Measure Of Software Similarity – služba pro odhalování plagiátů.
MySQL	Volně šiřitelný databázový server.
ODBC	Open DataBase Connectivity – programové rozhraní, pomocí něhož může aplikace přistupovat k datům ve zdroji dat pomocí jazyka SQL.
OS	Operační Systém.
PD4M	Plagiarism Detection For Moodle – modul pro odhalování plagiátů vytvořený pro systém Moodle.
PHP	PHP: Hypertext Preprocessor, nebo Personal Home Page – skriptovací jazyk pro tvorbu interaktivních webových aplikací.
RKR-GST	Running Karp-Rabin and Greedy String Tiling – algoritmus pro výpočet podobnosti dvou řetězců.
sed	Utilita pro dávkové zpracování textu.
SQL	Structured Query Language – jazyk ke komunikaci s databází.
YAP	Yet Another Program Plagiarism Detector – program pro odhalování plagiátů.
ZIP	Formát komprese souborů.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SCHÉMA PROPOJENÍ TABULEK



PŘÍLOHA P II: PODROBNÝ POPIS DATABÁZOVÝCH TABULEK

mdl_aca

Komentář k tabulce: Defines aca module (Základní tabulka modulu)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			identifikátor úkolu
course	int(10)	Ne	0	mdl_course.id	kurz, ke kterému je úkol přiřazen
name	varchar(255)	Ne			název úkolu
description	text	Ne			popis; zadání úkolu
format	tinyint(4)	Ne	0		formát zobrazení popisu
langtype	varchar(50)	Ne			typ prog. jazyka nebo činnosti
resubmit	tinyint(2)	Ne	0		povolení znovuodevzdání
preventlate	tinyint(2)	Ne	0		umožnění pozdního odevzdání
emailteachers	tinyint(2)	Ne	0		upozornění učitelů e-mailem
maxbytes	int(10)	Ne	100000		maximální velikost uploadovaného souboru
timedue	int(10)	Ne	0		datum a čas odevzdání
timeavailable	int(10)	Ne	0		datum a čas zadání
grade	int(10)	Ne	0		hodnocení
gradetype	tinyint(4)	Ne	0		typ hodnocení
grademax	tinyint(4)	Ne	0		typ hodnocení při překročení maxima
grademin	tinyint(4)	Ne	0		typ hodnocení při nedosažení minima
graderandval	int(10)	Ano	0		počet náhodně vybraných testovacích hodnot
showtest	tinyint(2)	Ne	0		povolení zobrazení výsledků testu funkčnosti
showcopy	tinyint(2)	Ne	0		povolení zobrazení výsledků testu plagiátů
sourcefile	varchar(255)	Ne			učitelský soubor
timemodified	int(10)	Ne	0		datum poslední změny

mdl_aca_pd_active

Komentář k tabulce: Table of active plagiat detectors (Tabulka nastavených metod pro hledání plagiátů)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
pdid	int(10)	Ne	0	mdl_aca_pds.id	identifikátor metody
acaid	int(10)	Ne	0	mdl_aca.id	identifikátor úkolu
border	int(11)	Ne	0		nastavená hranice povolené podobnosti
false_opt	tinyint(4)	Ne	0		typ činnosti při překročení hranice podobnosti
false_val	int(11)	Ne	0		hodnota pro typ činnosti
test_opt	tinyint(4)	Ne	0		volba u metody testování
test_val	varchar(100)	Ne			hodnota u metody testování
timemodified	int(10)	Ne	0		poslední změna

mdl_aca_pd_similarity

Komentář k tabulce: Info about similarities between submissions (Tabulka podobností mezi zdrojovými texty odevzdaných úkolů)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
id_pri	int(10)	Ne	0	mdl_user.id	identifikátor autora zdroje
id_sec	int(10)	Ne	0	mdl_user.id	identifikátor autora podobného textu
similarity	float	Ne	0		hodnota podobnosti (%)
acaid	int(10)	Ne	0	mdl_aca.id	identifikátor úkolu
activepdid	int(10)	Ne	0	mdl_aca_pd_active.id	identifikátor nastavené metody

mdl_aca_pds

Komentář k tabulce: Table of all usable aca plagiat detectors (Tabulka možných metod pro hledání plagiátů)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
langtype	varchar(50)	Ne		mdl_aca.langtype	identifikátor prog. jazyka
name	varchar(255)	Ne			název metody
timemodified	int(10)	Ne	0		poslední změna
resultmax	int(11)	Ne	0		maximální hodnota shody
resultmin	int(11)	Ne	0		minimální hodnota shody
resultdefault	int(11)	Ne	0		doporučená hodnota shody

mdl_aca_submissions

Komentář k tabulce: Info about submitted ac assignments (Tabulka odevzdaných úkolů)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
acaid	int(10)	Ne	0	mdl_aca.id	identifikátor úkolu
userid	int(10)	Ne	0	mdl_user.id	identifikátor autora
timecreated	int(10)	Ne	0		datum a čas vytvoření
timemodified	int(10)	Ne	0		datum a čas poslední změny
numfiles	int(10)	Ne	0		počet souborů
grade	int(11)	Ne	0		hodnocení
comment	text	Ne			komentář učitele
format	tinyint(4)	Ne	0		formát komentáře učitele
teacher	int(10)	Ne	0	mdl_user.id	identifikátor učitele
timemarked	int(10)	Ne	0		datum a čas ohodnocení
mailed	tinyint(1)	Ne	0		příznak odeslání e-mailu

mdl_aca_tv_active

Komentář k tabulce: Table of active test vectors (Tabulka nastavených testovacích hodnot)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
tvid	int(10)	Ne	0	mdl_aca_tvs.id	identifikátor metody testování
acaid	int(10)	Ne	0	mdl_aca.id	identifikátor úkolu
test_opt	tinyint(4)	Ne	0		volba typu testování
test_val	varchar(255)	Ne			hodnota volby testování
in_opt	tinyint(4)	Ne	0		volba typu vstupu
in_string	varchar(255)	Ne			hodnota typu vstupu
in_text	text	Ne			hodnota typu vstupu
grade	int(11)	Ne	0		počet bodů za test
timemodified	int(10)	Ne	0		datum a čas poslední změny

mdl_aca_tv_results

Komentář k tabulce: Test vector results for each submission (Tabulka výsledků testování)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
activetvid	int(10)	Ne	0	mdl_aca_tv_acitve.id	identifikátor nastavené testovací hodnoty
acaid	int(10)	Ne	0	mdl_aca.id	identifikátor úkolu
submissionid	int(10)	Ne	0	mdl_aca_submissons.id	identifikátor odevzdaného úkolu
indata	text	Ne			vstupní data testu
teacher_out	text	Ne			výstup programu učitele
student_out	text	Ne			výstup programu studenta
result	tinyint(2)	Ne	0		výsledek testu (0/1)
description	text	Ne			popis výsledků testu (rozdíly)
timemodified	int(10)	Ne	0		poslední změna

mdl_aca_tvs

Komentář k tabulce: Table of all usable aca test vectors (Tabulka metod pro testování funkčnosti programu)

Sloupec	Typ	Nulový	Výchozí	Odkazuje na	Komentáře
id	int(10)	Ne			
langtype	varchar(50)	Ne		mdl_aca.lanngtype	identifikátor prog. jazyka
name	varchar(255)	Ne			název metody
timemodified	int(10)	Ne	0		poslední změna