VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

VYUŽITÍ PROGRAMOVACÍCH TECHNIK PŘI TVORBĚ VÝUKOVÝCH KURZŮ URČENÝCH PRO TESTOVÁNÍ VÝPOČTU DANĚ Z PŘÍJMU FYZICKÝCH OSOB

USING PROGRAMMING TECHNIQUES TO PREPARE COURSES IN TESTING THE CALCULATION OF THE INCOME TAX OF NATURAL PERSONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR
Bc. MONIKA BREJCHOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. PETR DYDOWICZ, Ph.D.

BRNO 2011

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2010/2011
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Brejchová Monika, Bc.

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Využití programovacích technik při tvorbě výukových kurzů určených pro testování výpočtu daně z příjmu fyzických osob

v anglickém jazyce:

Using Programming Techniques to Prepare Courses in Testing the Calculation of the Income Tax of Natural Persons

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza problému
Vlastní návrhy řešení
Závěr
Seznam použité literatury

Seznam odborné literatury:

BORONCZYK, T. PHP 6, MySQL, Apache : vytváříme webové aplikace. 1. vyd. Brno : Computer Press, 2009. 816 s. ISBN 978-80-251-2767-4.
LECKY-THOPMSON, E. PHP 6 : programujeme profesionálně. 1. vyd. Brno : Computer Press, 2010. 718 s. ISBN 978-80-251-3127-5.
MACHÁČEK, I. Daň z příjmů fyzických osob 2010 : praktická pomůcka k daňové optimalizaci. 1. vyd. Praha : C. H. Beck, 2010. 273 s. ISBN 978-80-7400-188-8.
SCHWARTZ, B. MySQL profesionálně : optimalizace pro vysoký výkon. 1. vyd. Brno : Zoner Press, 2009. 712 s. ISBN 978-80-7413-035-9.
VÁŇOVÁ, T. Moodle v síti. 1. vyd. Brno : Tribun EU, 2008. 80 s. ISBN 978-80-7399-447-1.

Vedoucí bakalářské práce: Ing. Petr Dydowicz, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2010/2011.

L.S.

Ing. Jiří Kříž, Ph.D. Ředitel ústavu	doc. RNDr. Anna Putnová, Ph.D., MBA Děkan fakulty

V Brně, dne 25.05.2011

Abstrakt

Bakalářská práce pojednává o problematice využití programovacích technik LMS Moodle při tvorbě výukových kurzů určených pro testování výpočtu daně z příjmu fyzických osob. Popisuje vývojové prostředí Moodlu a funkce předdefinované ve stávajících modulech a ukazuje možnost algoritmizace výpočtů daně a implementace těchto algoritmů v systému. Součástí práce je návrh nového modulu umožňujícího jednoduchou tvorbu testů zaměřených na výpočty daně a jejich automatické vyhodnocování.

Abstract

Bachelor's thesis deals with the problems of using programming techniques in LMS Moodle to prepare courses in testing the calculation of the income tax of natural persons. The thesis describes development environment of Moodle and template functions in predefined modules and indicates the possibility of algoritm development of tax computations and of implemenation of these algorithms into the system. Finally there is a concept of new modul enabling an easy creation of the tests dedicated to tax computations and their authomatic evaluation.

Klíčová slova

LMS Moodle, modul, e-learning, daň z příjmů fyzických osob, PHP, MySQL

Keywords

LMS Moodle, module, e-learning, income tax of natural persons, PHP, MySQL

Bibliografická citace

BREJCHOVÁ, M. *Využití programovacích technik při tvorbě výukových kurzů určených pro testování výpočtu daně z příjmu fyzických osob.* Brno : Vysoké učení technické v Brně, Fakulta podnikatelská, 2011. 76 s. Vedoucí bakalářské práce Ing. Petr Dydowicz, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracovala jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušila autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 3. 6. 2011 .....

Poděkování

Děkuji vedoucímu své bakalářské práce Ing. Petru Dydowiczovi, Ph.D. za věnovaný čas a ochotnou pomoc. Dále děkuji Ing. Danielovi Kábovi, Ph.D. za inspiraci při volbě tématu bakalářské práce a Ing. Davidu Pačesovi z Centra výpočetních a informačních služeb VUT za rady týkající se instalace LMS Moodle a dalšího postupu při vytváření modulu.

Obsah

Úvod

Bakalářská práce bude zaměřena na problematiku využití programovacích technik LMS (Learning Management System) Moodle při tvorbě výukových kurzů určených pro testování výpočtů daní z příjmů fyzických osob. Modul kurzu by měl umožnit zadávání úloh určených pro výpočet daně a automatické vyhodnocování výpočtů provedených účastníky kurzu na základě jejich porovnání se správnými výpočty získanými pomocí příslušného algoritmu vycházejícího z aktuálního zákona o dani z příjmů.

Výsledky práce budou využity v existujícím kurzu zaměřeném na daňovou problematiku, který je určen pro studenty VUT. Jedná se o kurz „Daňová soustava (HDS 09/10L)“ vedený Ing. Danielem Kábou, Ph.D., jenž byl spuštěn v letním semestru 2010 primárně pro studenty Fakulty strojního inženýrství. Kurz je zaměřen na poskytnutí základních znalostí daňových zákonů, správy daní, poplatků a odvodů a problematiky pojistného na sociální a zdravotní pojištění. Stávající kurz má být do budoucna rozšířen a používán firmou zaměřenou na účetnictví a daňové poradenství jako podpůrná aplikace pro testování znalostí zaměstnanců.

Přínos zapojení interaktivních vzdělávacích systémů do výuky studentů či zaměstnanců je při adekvátním použití nesporný. Cílem zapojení vhodných e-learningových nástrojů do vzdělávání by mělo být kromě finanční úspory výrazné zefektivnění výuky z pohledu studenta i učitele.

Pro mnohé přednosti (kterým dominuje především flexibilita) bývá pro účely tvorby výukových kurzů využíván LMS Moodle – volně šiřitelná aplikace s otevřeným kódem, kterou lze jako softwarový balík zdarma nainstalovat přímo na server a následně modifikovat podle potřeb.

Cíle a metody práce

Stěžejním cílem práce je vytvořit v LMS Moodle modul kurzu, který umožní automatické vyhodnocování výpočtů daní z příjmů fyzických osob. Vyhodnocení bude provedeno na základě porovnání výpočtů provedených účastníky kurzu se správnými výpočty získanými pomocí příslušného algoritmu vycházejícího z aktuálního zákona o dani z příjmů.

Dílčí cíle práce jsou:

- popsat vývojové prostředí LMS Moodle a postup při tvorbě nového modulu
- navrhnout algoritmus pro výpočet daně z příjmu fyzických osob a jeho implementaci v modulu
- navrhnout a implementovat možnost automatického vyhodnocování výpočtů provedených účastníky kurzu na základě jejich porovnání se správnými výpočty získanými pomocí příslušného algoritmu vycházejícího z aktuálního zákona o dani z příjmů

Modul bude vyvíjen na systému odděleném od systému, na němž je provozován stávající kurz (VUT neumožňuje vývoj na ostré verzi vlastního LMS). Základním předpokladem tedy bude zajištění softwarových nástrojů potřebných pro instalaci a chod systému. Algoritmy pro výpočet daně potřebné pro implementaci v modulu budou odvozeny z aktuálního zákona o dani z příjmů. Vlastní vývoj modulu bude probíhat pomocí relevantních programovacích nástrojů na základě analýzy stávajících modulů. Na závěr bude zhodnocena ekonomická efektivita inovace systému spočívající v integraci vytvořeného modulu.

Pro získání informací o možnostech programování v systému Moodle a tvorbě zásuvných modulů lze kromě literatury vztahující se k technologiím využívaných v Moodlu vycházet z dokumentace vytvářené komunitou vývojářů tohoto systému.[1]

[1] Dokumentace je dostupná z oficiálních webových stránek http://moodle.org/ na adrese http://docs.moodle.org/overview/.

1 Teoretická východiska práce

1.1 LMS Moodle

1.1.1 Obecná charakteristika

LMS Moodle (Learning Management System Moodle nebo také Modular Object-Oriented Dynamic Learning Environment[2]) je modulární objektově orientovaná dynamická aplikace na bázi PHP určená k tvorbě elektronických online výukových kurzů. Data jsou ukládána v MySQL či jiném databázovém systému (např. PostgreSQL, Oracle, Access, Interbase, ODBC). (VÍCHA, 2004)

Systém je vyvíjen komunitou vývojářů z celého světa. Koordinátorem projektu je Martin Dougiamas z Austrálie. O lokalizaci české verze se stará David Mudrák z Katedry informačních technologií a technické výchovy Pedagogické fakulty Univerzity Karlovy v Praze. (Tamtéž.)

Při adekvátním použití slouží Moodle jako nástroj umožňující realizaci výukových metod navržených v souladu s principy konstruktivisticky orientované výuky. V rámci kurzů vytvořených pomocí tohoto systému lze publikovat studijní materiály v různých formátech, zakládat diskuzní fóra, chat nebo ankety, komunikovat prostřednictvím zpráv, vytvářet a hodnotit workshopy, úkoly a testy, používat různé metody vyhodnocování atd. (VÁŇOVÁ, 2008, s. 6)

Kurzy lze poskládat z vlastních i předdefinovaných modulů (určených pro testování, workshopy, diskuze atp.), v jejichž rámci je možné využít velké množství funkcí a nástrojů pro mnohostrannou analýzu dat získaných od studentů.

Moodle je volně šiřitelný software s otevřeným kódem (Open Source), je poskytován zdarma pod licencí GNU. Svým uživatelům tedy dopřává značnou svobodu, je však chráněn autorskými právy a při jeho šíření a upravování je nutné dodržet stanovená pravidla (např. uplatnění stejných licenčních podmínek u všech odvozených produktů).

[2] Název Moodle lze interpretovat také jako lingvistickou hříčku – anglické slovo „moodle“ označuje líné bloumání od jednoho k druhému či hravost vedoucí k pochopení problému a podporující tvořivost. (HOUSHOLDER, 2006) Kromě výše zmíněného označení LMS je používáno také obdobné označení CMS (Course Management System) či VLE (Virtual Learning Environment). (*MOODLE.CZ*, 2010)

(*Licence – MoodleDocs,* 2009) Editovat lze funkcionalitu i vizuální podobu systému (prostřednictvím HTML a CSS).

V Moodlu lze definovat několik typů účastníků kurzu (tzv. rolí), kteří mají různé oprávnění k využívání jednotlivých funkcí (tzv. pravomocí). Administrátor má na starosti instalaci, aktualizace, nastavuje možnosti zabezpečení, ladění výkonu, přístupových práv atp. Základní úpravy v kurzu (vkládání studijních materiálů, tvorba testů atp.) může provádět „učitel s právy úprav". Role lze uživatelům přiřazovat pro různé kontexty (např. v rámci jednoho kurzu či celého systému). Různé verze Moodlu podporují rozdílné role a i tyto role a jim příslušné pravomoci lze v různých implementacích měnit. (*Rukověť správce – MoodleDocs,* 2010) Tvorbu a přidávání nových modulů lze provádět pouze pod administrátorským účtem, který umožňuje přístup do adresáře se zdrojovými kódy systému.

1.1.2 Instalace systému

Moodle lze provozovat na platformách Unix, Linux, Windows, Mac OS X, Netware i na jakémkoli jiném systému podporujícím PHP. (*MOODLE.CZ,* 2010) Pro úspěšnou instalaci je zapotřebí software pro webový server (doporučený je Apache), PHP bez tzv. safe_mode ve verzi, kterou vyžaduje konkrétní verze systému Moodle, a relační databázový server (doporučený je MySQL). (*Instalace – MoodleDocs*, 2010) Moodle lze tedy instalovat na servery webových poskytovatelů, které tyto technologie podporují, příp. na vlastní počítač, který je předem nutné zmíněnou platformou vybavit.

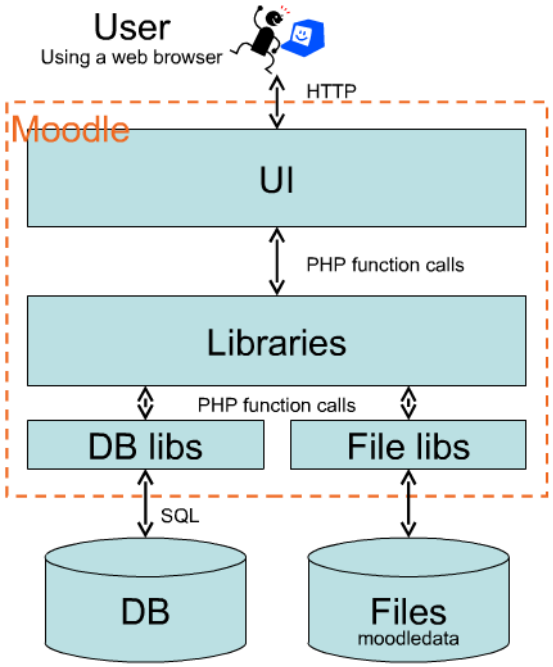
MySQL je multiplatformní Open Source databáze (databázový systém). Komunikace v ní probíhá prostřednictvím jazyka SQL. (*MySQL :: The world's most popular open source database,* 2010) Architektura MySQL se od jiných databázových systémů odlišuje tím, že má pro svou flexibilitu (v hardwarové konfigurovatelnosti, podpoře datových typů atp.) široký záběr a funguje dobře i ve velmi náročných prostředích jako jsou webové aplikace. (SCHWARTZ, 2009)

Apache je Open Source webový server, který umožňuje publikovat webové stránky na Internetu. Jeho úlohou je reagovat na příchozí požadavky od prohlížeče a vracet příslušnou odpověď. (BORONCZYK, 2009) Moodle podporuje Apache verze 1 i 2. (*Apache – MoodleDocs,* 2011)

Moodle lze instalovat jako standardní balík zdrojových kódů pro konkrétní verzi nebo ve formě **CVS (Concurrent Versions System)**. CVS je systém určený ke správě všech historických verzí softwaru. Je důležitou součástí tzv. Source Configuration Management / Source Code Management (SCM). Tyto systémy umožňují více uživatelům současně spravovat skupinu souborů (repozitář) uloženou na souborovém systému serveru, sledovat změny těchto souborů a vytvářet nové vývojové větve. (*CVS – Open Source Version Control*, 2010). V databázi CVS je ukládán kompletní zdrojový kód Moodlu, vývojáři mohou přistupovat k jednotlivým verzím. (*CVS for Administrators – MoodleDocs*, 2010) Poslední alternativou je instalovat distribuci Moodlu určenou pro integraci s jiným softwarem (např. s redakčním systémem Joomla). (*Integrations – MoodleDocs*, 2010).

1.1.3 Architektura systému

Architektura systému je soubor specifikací, spojujících prvků, pravidel a interakcí mezi spolupracujícími vrstvami systému. Každá vrstva má rozhraní, přes které komunikuje s další vrstvou, přičemž každá z nich může být uložena na jiném počítači. (VACH, 2006) Architektura Moodlu je tvořena třemi vrstvami – prezentační, aplikační a datovou. (HUNT, 2010)

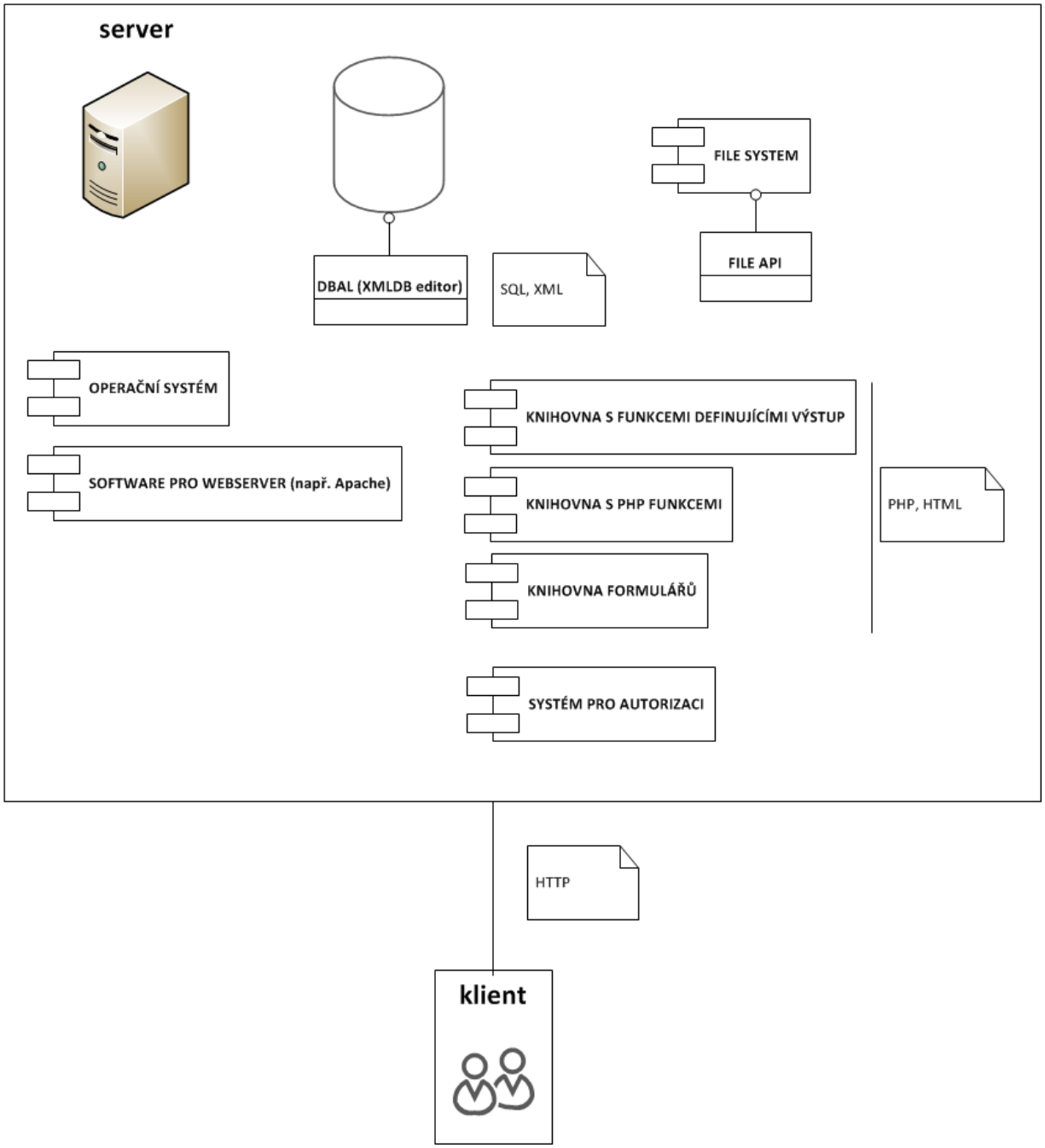


Obr. 1: Třívrstvá architektura Moodlu (Zdroj: HUNT, 2010)

Klíčovými komponenty pro chod systému Moodle jsou:

- **Databáze**. Je tvořena přibližně 200 tabulkami určenými pro jednotlivé moduly. (*Development:Database schema introduction – MoodleDocs*, 2010)[3]
- **Databázová abstraktní vrstva** (**DBAL** – Database Abstraction Layer) – tzv. **XMLDB editor** určený pro správu databázových souborů (např. definici a úpravu tabulek). XMLDB editor definuje databázi v jazyce XML (prostřednictvím souboru *db/install.xml*) – převádí defaultní databázová schémata (např. MySQL) do XMLDB. Tím odpadá nutnost vytvářet při tvorbě modulů více struktur pro různé databázové systémy. (*Development:Using XMLDB – MoodleDocs*, 2008)
- **File API** (File Application programming interface) – rozhraní pro programování aplikací pro správu složek. (*Development:Using the File API – MoodleDocs*, 2010)
- **Systém pro určení rolí a příslušných kompetencí uživatelů**.
- **Knihovna formulářů** *formslib.php* pro vytváření přístupných a bezpečných HTML formulářů. (*Development:lib/formslib.php – MoodleDocs*, 2009)
- **Knihovna s PHP funkcemi** *moodlelib.php*. (*Development:lib/moodlelib.php – MoodleDocs*, 2011)
- **Knihovna s funkcemi definujícími výstup systému** *weblib.php*. (*Development:lib/weblib.php – MoodleDocs*, 2007)
- **Knihovna s funkcemi definujícími přístup k databázi** *datalib.php*. (*Development:Developer documentation – MoodleDocs*, 2011)

[3] Kompletní databázové schéma současné verze systému je dostupné na adrese: http://docs.moodle.org/en/images_en/5/5a/Moodle2erd.png

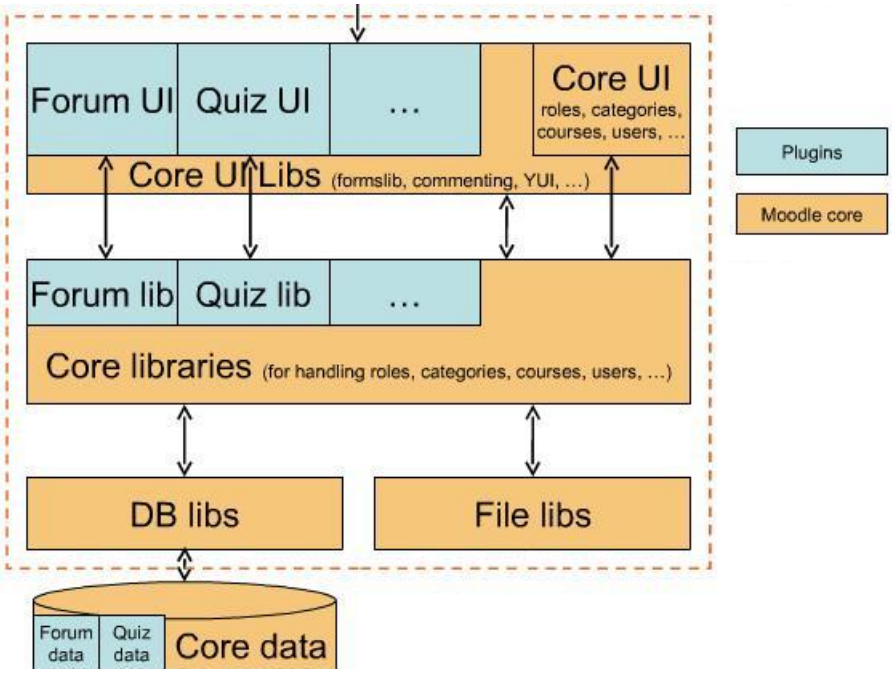


Obr. 2: Komponenty systému (Vlastní zpracování)

Další součástí systému je velké množství knihoven pro specifická užití (např. pro autentifikaci, zpracování e-mailů, síťové funkce Moodlu atp.). (Tamtéž.)

Systém lze modifikovat jak zásahem do vlastního zdrojového kódu (prostřednictvím tzv. patchů), tak obohacením o různé typy zásuvných modulů (pluginů). Tyto moduly se specifickou funkcionalitou využívají prostředky celého systému a rozkládají se ve všech jeho vrstvách (viz Obr. 3). Jednotlivé moduly mohou obsahovat další „podmo-

duly“, přičemž dělení a struktura modulů se v jednotlivých verzích Moodlu liší. Jedním z typů pluginů je tzv. typ úlohy (Question type), který bude podrobněji analyzován a dle potřeby modifikován v této práci. (Tamtéž.)



Obr. 3: Umístění zásuvných modulů v systému (Zdroj: HUNT, 2010)

1.1.4 Počítačové jazyky využívané v systému

V následujících podkapitolách budou stručně popsány základní počítačové jazyky využívané jednotlivými komponenty systému v jeho aplikační i uživatelské vrstvě. Nebude zde uveden detailní výčet a popisy jednotlivých prvků těchto jazyků – ty budou doplněny v praktické části práce podle reálně využitých programovacích prostředků při vývoji modulu.

1.1.4.1 Značkovací jazyky

HTML

HTML (HyperText Markup Language) je značkovací jazyk pro publikaci dokumentů na WWW. Podporuje skriptovací jazyky, kaskádové styly (CSS) i multimediální prvky (konkrétní míra podpory je odvislá od verze HTML). (W3C, 1999) V Moodlu jej

lze užívat takřka neomezeně. (*HTML in Moodle – MoodleDocs*, 2009) Kód lze editovat prostřednictvím uživatelsky přívětivého HTML editoru. (*HTML editor – MoodleDocs*, 2010)

XML

XML (Extensible Markup Language) je obecný flexibilní značkovací jazyk odvozený od jazyka SGML (Standard Generalized Markup Language) určený pro elektronické publikování a výměnu dat. (W3C, 2003) V Moodlu slouží XML pro definování objektů databáze (XMLDB editor – viz výše) a pro import a export úloh používaných v modulu Test. (*Moodle XML format – MoodleDocs*, 2010)

1.1.4.2 Programovací jazyky

Pro vývoj Moodlu jsou určeny tzv. skriptovací jazyky. Jedná se o interpretované programovací jazyky uzpůsobené pro rychlý a snadný vývoj využívané především pro programování webových aplikací (nevyžadují deklaraci proměnných, podporují vyšší datové typy atp.). (VÁCLAVOVIČ, 2001)

PHP

Moodle je vyvíjen primárně v jazyce PHP. PHP je široce rozšířený mnohoúčelový skriptovací programovací jazyk určený především pro programování dynamických webových stránek. (*PHP: Hypertext Preprocesor*, 2010). Pro správný běh systému Moodle je nutná podpora PHP bez tzv. *safe_mode*. (*Instalace – MoodleDocs*, 2010) Vývojáři mohou využít standardní nástroj PHPXref, který ulehčuje práci s PHP kódem pomocí integrovaného přístupu k PHP dokumentaci[4], příp. prohlížet kód na webové aplikaci PHPXref 0.7.[5] (*Development:PHPXref – MoodleDocs*, 2009)

PHP patří mezi objektově orientované programovací (OOP) jazyky. Jimi vytvořená aplikace je tedy modelována jako skupina „spolupracujících objektů, jež nezávisle provádí určité aktivity.“ OOP jazyky jsou charakteristické tím, že obsahují následující

[4] Software je dostupný ke stažení na adrese http://phpxref.sourceforge.net/.

[5] Online PHPXref 0.7 je dostupný na adrese http://xref.moodle.org/nav.html?index.html, dokumentace PHP kódu na adrese http://phpdocs.moodle.org/.

prvky a principy: *třídy* (jednotka kódu skládající se z proměnných a funkcí popisující atributy a metody všech objektů množiny), *objekty* (instance třídy), *dědičnost* („schopnost definovat třídu jednoho typu jakožto podtřídu jiného typu“), *polymorfizmus* („umožňuje definovat třídu jako člena více jak jedné kategorie“), *rozhraní* („způsob, jak specifikovat, že je objekt schopen poskytnout určitou funkcionalitu, aniž by se definovalo, jak této funkcionality dosáhne“) a *zapouzdření* („schopnost objektu chránit před přístupem svá interní data“). (LECKY-THOMSON, NOWICKY, 2010, s. 29–31)

Velká a malá písmena se v tomto jazyce rozlišují pouze u názvů proměnných. Mezery, tabulátory a znaky pro ukončení řádků jsou ignorovány. Kód PHP je v rámci HTML dokumentu vždy nutné uvést a zakončit (existuje více platných způsobů). (KREJČÍ, 2006, s. 9) Po uvození kódu následuje deklarace *proměnných* (pomocí znaku $), *konstant* (pomocí funkce define), *funkcí* a *tříd* s příslušnými *atributy* a *metodami*. (Tamtéž, s. 29–38) Proměnné se nedeklarují předem, ale v okamžiku přiřazení hodnoty do proměnné. V kódu lze využít také předefinované proměnné a konstanty (Tamtéž, s. 87–95) a funkce (matematické funkce, funkce pro práci s proměnnými, konstantami, řetězci, poli, funkcemi, objekty, třídami, soubory, databází MySQL) (Tamtéž, s. 39–85). Definice *datových typů* (např. bool, integer, string,…) není přímou součástí syntaxe jazyka (PHP je tzv. dynamicky typovaný jazyk). (Tamtéž, s. 18–20)

Pro operace s hodnotami konstant a proměnných jsou používány aritmetické a logické *operátory* (např. +, and), operátory inkrementace a dekrementace (např. ++, -), operátory na úrovni bitů (např. &), pro práci s poli (např. []), pro práci s objekty a třídami (např. new), pro řízení chyb (např. @), přiřazení (např. =), operátory typové konverze (např. (bool)), porovnávací operátory (např. ==), prováděcí operátory (//) a operátor zřetězení (.). (Tamtéž, s. 21–28) Pro označení *řídicích struktur* (např. for, if-else-elseif, while,…) se používá standardní notace na začátku a na konci těla struktur, příp. tzv. dvojtečková notace. (Tamtéž, s. 10–13) *Příkazy* (tzv. *jazykové konstrukty*; např. echo, print, return…) se používají podobně jako funkce, jsou však přímou součástí gramatiky jazyka. (Tamtéž, s. 14–18)

Součástí PHP kódu jsou také tzv. *konfigurační direktivy*, které definují vlastnosti a chování některých funkcí týkající se interpretu PHP, souborů a adresářů, databáze MySQL, zpracování dat, zpracování a záznamu chyb a zabezpečení. Standardně jsou uloženy v souboru php.ini. (Tamtéž, s. 97–104)

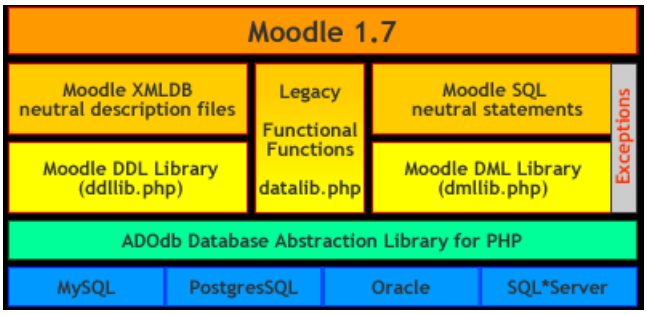
Další skriptovací jazyky

V Moodlu je až do verze 2.0 užití jiných skriptovacích jazyků než je PHP (např. JavaScript či VBScript) zakázáno, části kódu psané v těchto jazycích jsou automaticky odstraňovány. (*HTML in Moodle – MoodleDocs*, 2009) Od verze 2.0 je jejich užití díky novému API podporováno, z důvodu přístupnosti je však nutné používat skriptování (a jiné technologie závislé na uživatelském nastavení – např. flash) obezřetně tak, aby byl Moodle funkční i při jejich deaktivaci v prohlížeči. (*Development:JavaScript guidelines – MoodleDocs*, 2011)

1.1.4.3 Dotazovací jazyky

SQL

SQL (Structured Query Language) je standardizovaný strukturovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. (STEPHENS, PLEW, JONES, 2010, s. 31) Moodle sjednocuje rozdíly SQL v různých databázových implementacích prostřednictvím nadstavby, která zahrnuje **DDL** (Data Definition Language) vrstvu pro vytváření a upgrade databáze a **DML** (Data Manipulation Language) vrstvu pro dotazování a ukládání dat. Pro provádění akcí v databázi používá Moodle kombinaci definovaných SQL příkazů (pro práci s obsahem databáze) a XMLDB popisů (pro práci s objekty databáze). Pro složitější dotazy lze použít výjimky, ty by však měly být minimalizovány a v případě možnosti nahrazeny výše zmíněnými definovanými příkazy. Každý z těchto dvou jazyků používá vlastní knihovnu – DDL Library obsahuje funkce pro práci s objekty, DML Library obsahuje funkce pro práci s obsahem databáze. Knihovna Datalib obsahuje další funkce nezařazené do žádné z těchto knihoven. Všechny tyto funkce zajišťují veškeré akce prováděné vrstvou ADOdb DAL (Database Abstraction Layer) for PHP, která přímo komunikuje s databází a získané výsledky zasílá zpět knihovnám. (*Development:XMLDB_introduction – MoodleDocs,* 2010)



Obr. 4: Komunikace systému s databází
(Zdroj: *Development:XMLDB_introduction – MoodleDocs*, 2010)

1.2 Daň z příjmů fyzických osob

1.2.1 Stručná charakteristika

Daň z příjmů fyzických osob (dále *DPFO*) je přímá daň, jejímž *předmětem* jsou příjmy ze závislé činnosti a funkční požitky, příjmy z podnikání a jiné samostatně výdělečné činnosti, příjmy z kapitálového majetku, příjmy z pronájmu a ostatní příjmy. (Zákon č. 586/1992 Sb., 1992, § 3) Zákon o daních z příjmů dále definuje příjmy, jež nejsou předmětem daně (Tamtéž.) a příjmy, jež jsou od daně osvobozeny (Tamtéž, § 4).

Příjmy, které nejsou předmětem daně nebo jsou od daně osvobozeny, nebudou v kurzech uvažovány. Výpočet bude primárně zaměřen na daně z příjmů plynoucích z pracovněprávního vztahu, jež patří mezi tzv. příjmy ze závislé činnosti (Tamtéž, § 6).

Základním typem příjmů z pracovněprávního vztahu je mzda a plat. V soukromém sektoru se náhrada zaměstnavatele zaměstnanci za vykonanou činnost označuje jako *mzda*, ve státním sektoru jako *plat*. (Zákon č. 262/2006 Sb., § 109) Daň z příjmů je jednou ze srážek ze mzdy[6]. (Tamtéž, § 147) Odečtením daně a dalších srážek od tzv. *hrubé mzdy* (např. sociálního zabezpečení a zdravotního pojištění zaměstnance) se vypočte tzv. *čistá mzda*. (Zákon č. 99/1963 Sb., § 277) Minimální hrubá mzda je 7955 Kč měsíčně (stav k 24. 3. 2011). (Zákon č. 262/2006 Sb., § 111)

Na výpočet daně má vliv také forma *pracovněprávního vztahu* mezi zaměstnancem a zaměstnavatelem. Pracovněprávním vztahem může být pracovní poměr (Zákon č. 262/2006 Sb., § 109) nebo dohody o pracích konaných mimo pracovní poměr – dohoda o provedení práce a dohoda o pracovní činnosti (Tamtéž, § 75 a 76).

Poplatníky daně z příjmů fyzických osob jsou fyzické osoby, „které mají na území České republiky trvalé bydliště nebo se zde obvykle zdržují“ (u těch se daňová povinnost vztahuje na příjmy plynoucí ze zdrojů na území České republiky i na příjmy plynoucí ze zdrojů v zahraničí) a ti, „o nichž to stanoví mezinárodní smlouvy“ (u těch se daňová povinnost vztahuje pouze na příjmy plynoucí ze zdrojů na území České republiky). (Zákon č. 586/1992 Sb., 1992, § 2) Pro účely výpočtů budou uvažovány fyzické osoby žijící nebo obvykle se zdržující v České republice a mající příjmy plynoucí ze zdrojů na území České republiky.

[6] Analogicky u platu.

1.2.2 Algoritmus pro výpočet daně z příjmů fyzických osob

Algoritmus pro výpočet daně z příjmů fyzických osob závisí na typu daně vyplývajícího ze vstupních dat pro výpočet daně:

1. **Srážková daň** se počítá v případě, že jsou splněny všechny následující podmínky:

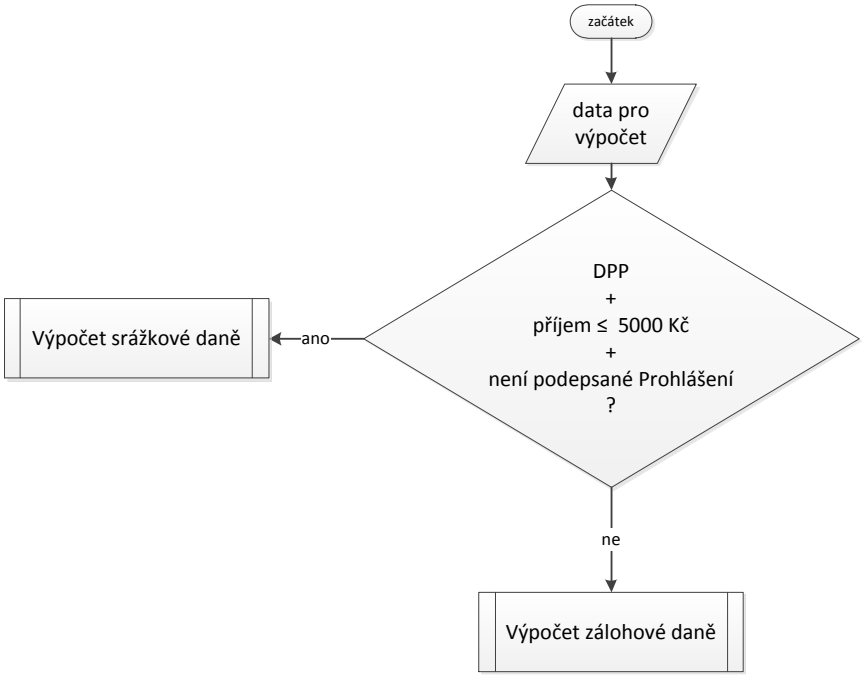
   a) příjem je nižší nebo roven 5000 Kč (včetně),

   b) zaměstnavatel má se zaměstnancem uzavřenou dohodu o provedení práce,

   c) zaměstnanec nemá podepsané Prohlášení poplatníka daně, na základě kterého lze uplatnit slevy a daňové bonusy.

2. **Zálohová daň** se počítá v ostatních případech. (KUČEROVÁ, 2009)

Způsob výběru metody výpočtu zobrazuje následující vývojový diagram.



Obr. 5: Volba metody výpočtu DPFO – vývojový diagram (Vlastní zpracování)

1.2.2.1 Srážková daň

Srážková daň představuje 15 % z hrubé mzdy.

srážková daň = 0,15 × hrubá mzda; základ pro výpočet daně i vypočtená daň se zaokrouhlují na celé koruny dolů

(Tamtéž.)

1.2.2.2 Zálohová daň

Zálohová daň se vypočítá ze zálohy na dani odvíjející se od výše hrubé (resp. superhrubé) mzdy. Od zálohy se odečítají případné slevy na dani a daňové bonusy.

zálohová daň = záloha na daň – (slevy na dani + daňové bonusy)

(Tamtéž.)

Záloha na daň představuje 15 % ze základu daně (tzv. superhrubé mzdy) a do výpočtu se zahrnuje v případě, že nejsou splněny podmínky pro uplatnění srážkové daně.

záloha na daň = 0,15 × superhrubá mzda; základ pro výpočet daně se zaokrouhluje na celé koruny dolů

(Tamtéž.)

> **Superhrubá mzda** je součtem hrubé mzdy, sociálního zabezpečení zaměstnavatele (SZ_1) a zdravotního pojištění zaměstnavatele (ZP_1).[7]
>
> superhrubá mzda = hrubá mzda + SZ_1 + ZP_1
>
> **SZ_1** = 0,25 × hrubá mzda; zaokrouhluje se na celé koruny nahoru
>
> **ZP_1** = 0,09 × hrubá mzda; zaokrouhluje se na celé koruny nahoru
>
> (*Superhrubá mzda základem daně – Finance.cz*, 2011)

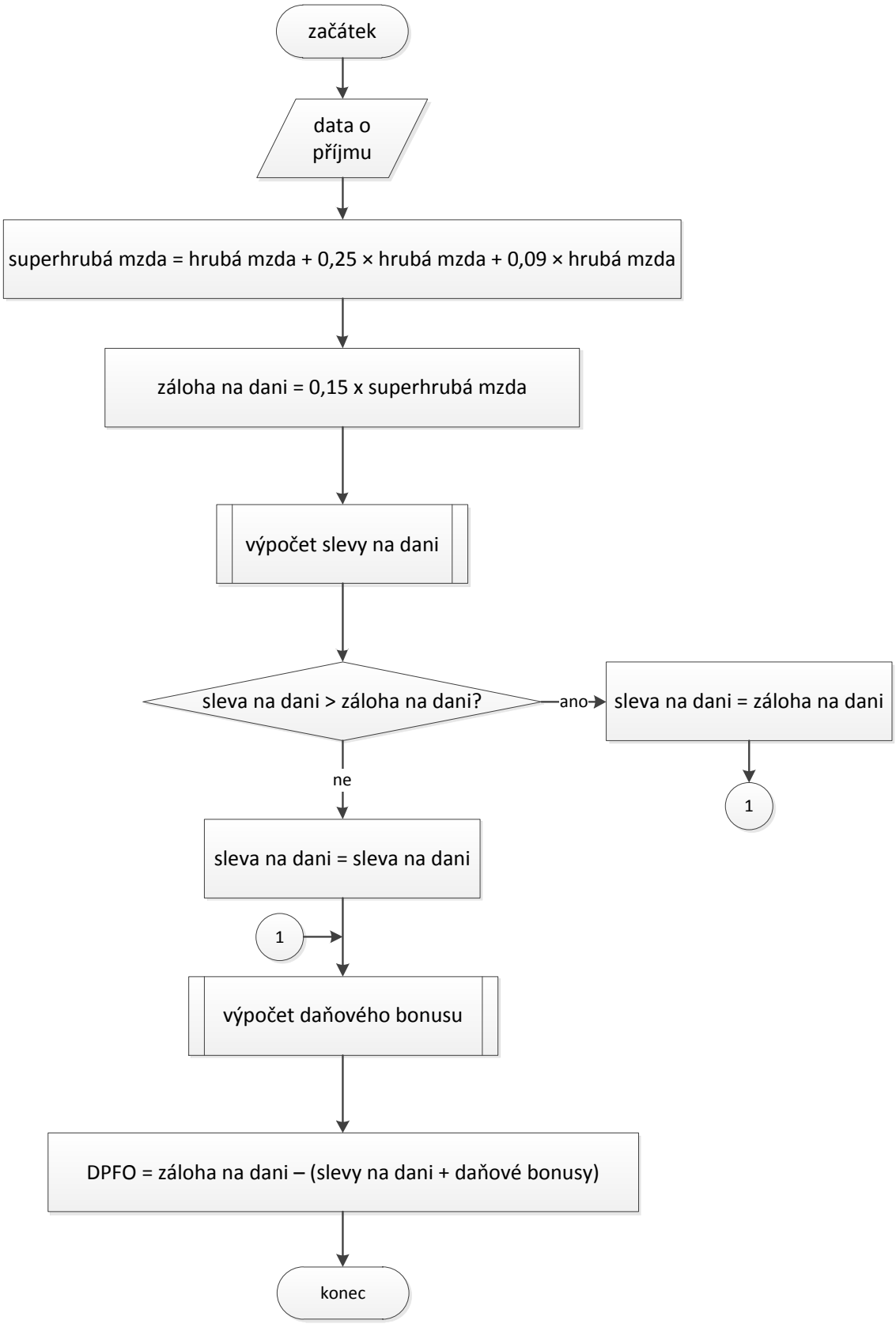
Sleva na dani může snížit zálohu na daň maximálně do výše 0 Kč. Pro zjednodušení budou do výpočtů zahrnuty jen tyto typy slev: poplatník – 1970 Kč měsíčně[8], student prezenčního studia do 26 let (resp. 28 let) – 335 Kč. Ostatní slevy se vztahují za dalších zákonem splněných podmínek na manželku či manžela poplatníka a invaliditu. (Zákon č. 586/1992 Sb., 1992, § 35ba)

[7] Legislativně je superhrubá mzda stanovena jako základ daně z příjmů při výpočtu záloh na daň z příjmů. (Zákon č. 586/1992 Sb., 1992, § 6, odst. 13)

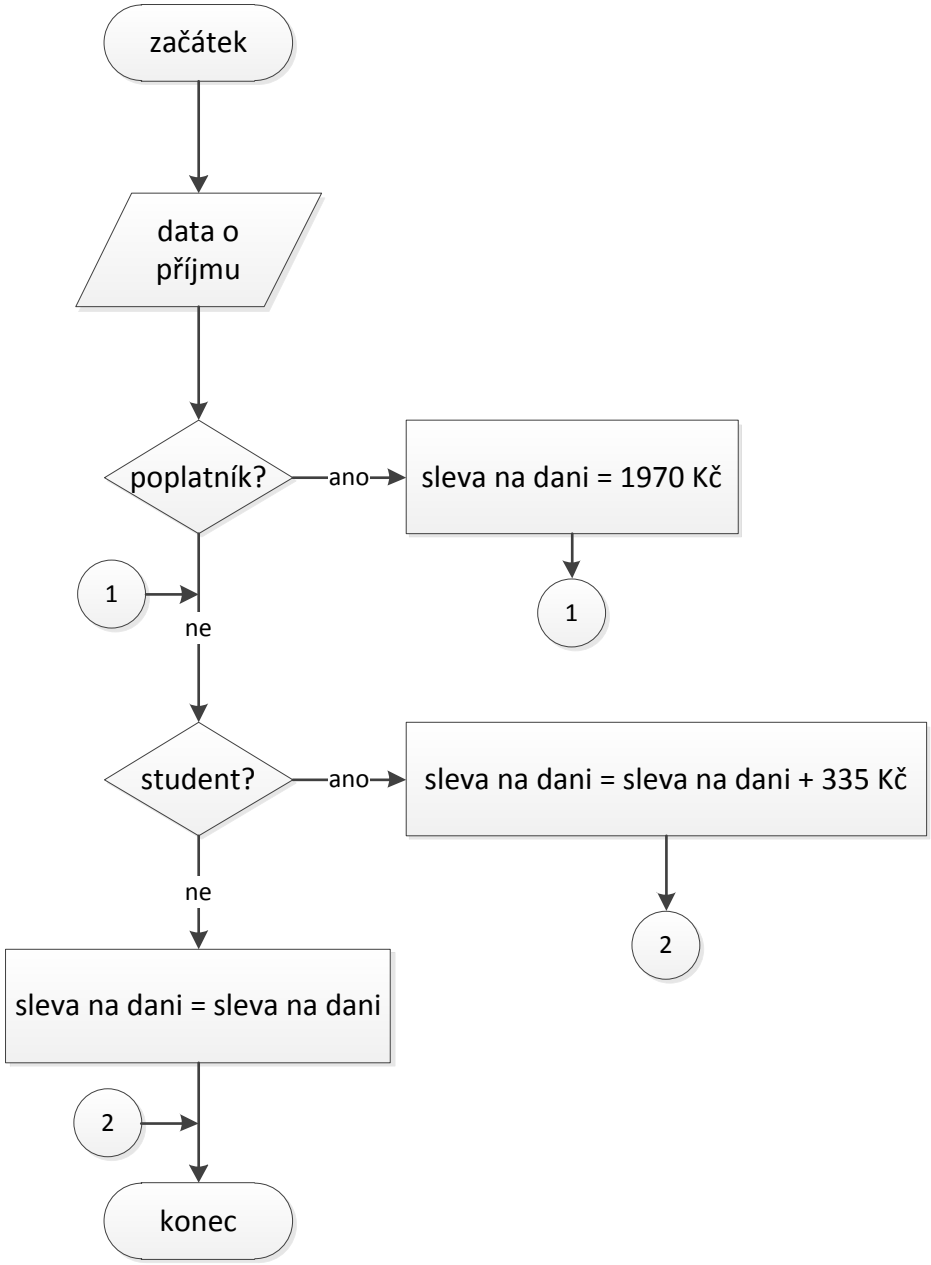
[8] Od roku 2011 se výše slevy na poplatníka z původních 2070 Kč snižuje o 100 Kč měsíčně (tzv. povodňová daň). Všechny zde uvedené částky jsou v měsíční výši, zákon udává částky v roční výši.

Daňový bonus lze uplatnit na vyživované dítě žijící s poplatníkem ve společné domácnosti ve výši 967 Kč na jedno dítě (do 18, resp. do 26 let), maximálně však ve výši 4835 Kč ročně (a za splnění dalších zákonných podmínek, ke kterým nebude v zadávání testů přihlíženo). (Tamtéž, § 35c)

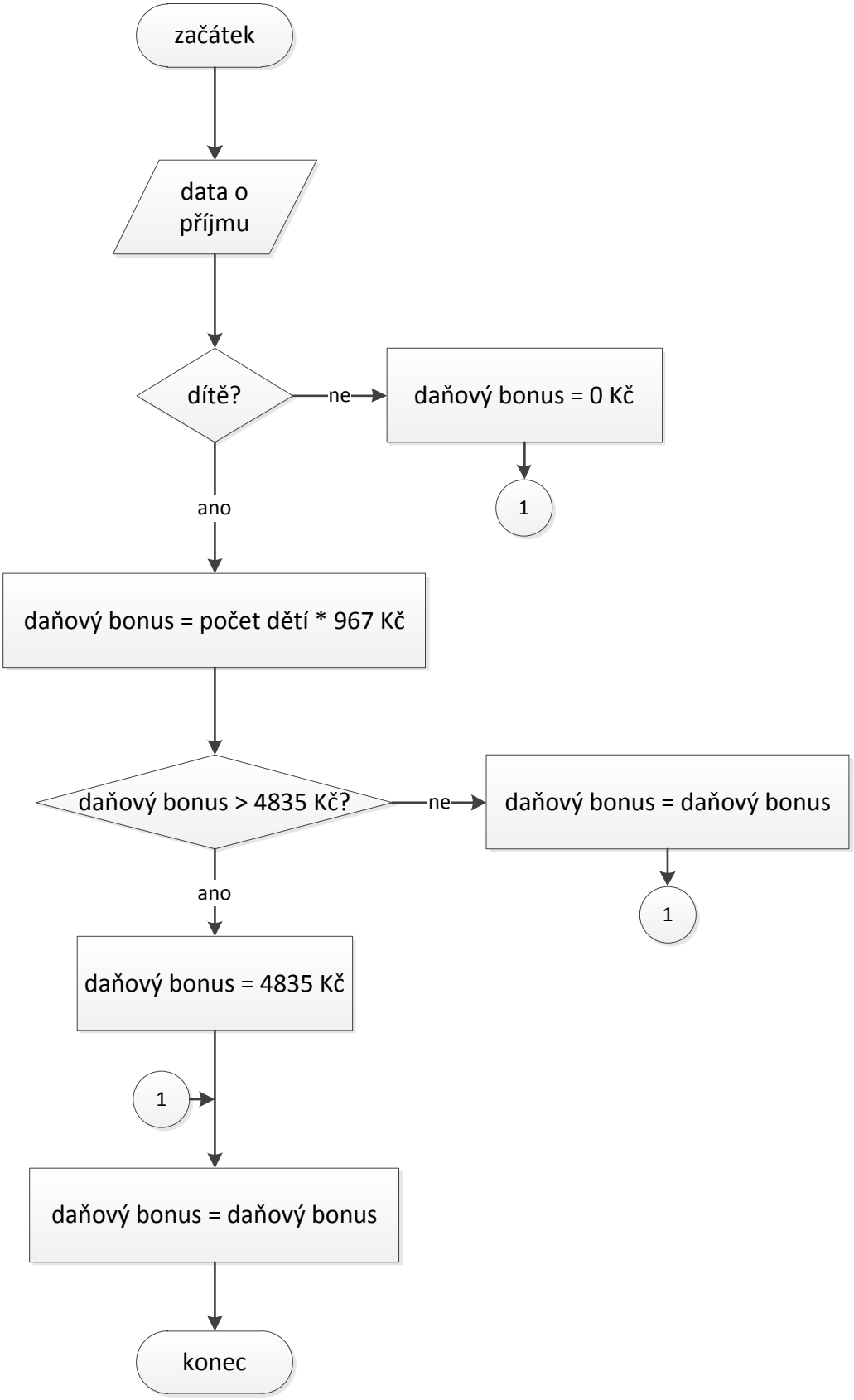
Výpočty zálohové daně, slevy na dani a daňového bonusu jsou zobrazeny na následujících diagramech.



Obr. 6: Výpočet zálohové daně – vývojový diagram (Vlastní zpracování)



Obr. 7: Výpočet slevy na dani – vývojový diagram (Vlastní zpracování)



Obr. 8: Výpočet daňového bonusu – vývojový diagram (Vlastní zpracování)

2 Analýza problému a současné situace

V této kapitole bude provedena podrobná analýza uživatelské, aplikační (programátorské) a datové vrstvy modulu Test a dalších vybraných s problematikou souvisejících modulů. Tato analýza bude sloužit jako východisko pro tvorbu nového modulu.

2.1 Uživatelské rozhraní

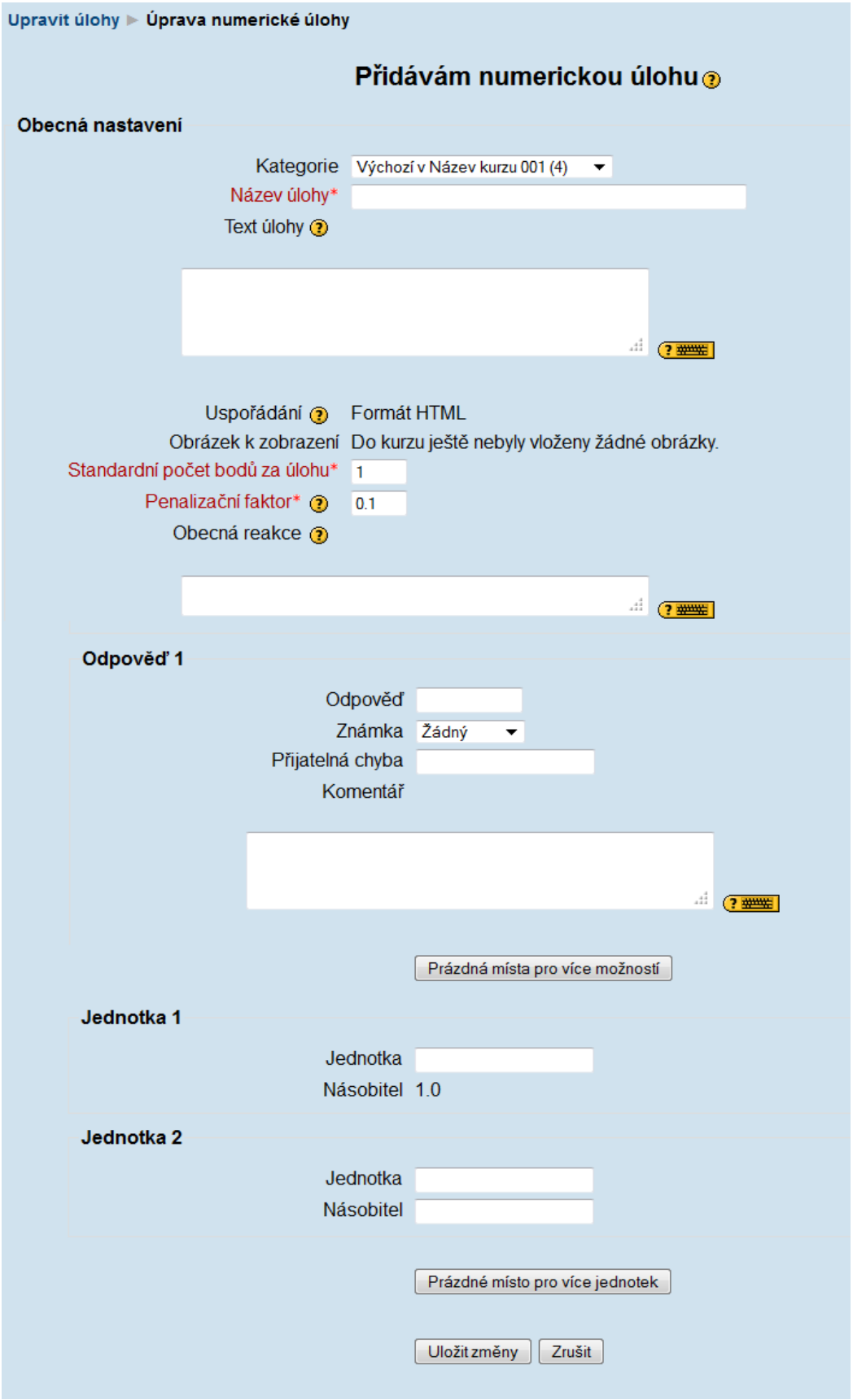
Úkoly s využitím výpočetních operací lze v kurzech zadávat prostřednictvím modulu Test. Tento modul umožňuje vytvářet testovací část kurzu, do níž lze zahrnout několik typů úloh slovního i matematického typu: popis, dlouhá tvořená odpověď, přiřazování, doplňovací úloha (cloze), úloha s výběrem odpovědí, krátká tvořená odpověď; numerická úloha, vypočítávaná úloha. (*Typy úloh – MoodleDocs*, 2008) Vzhledem k charakteru cílového modulu bude vhodné zaměřit se na úlohy matematického typu, tj. numerickou a vypočítávanou úlohu.[9]

2.1.1 Typy matematických úloh

2.1.1.1 Numerická úloha

Jedná se o jednodušší typ předdefinované úlohy matematického typu, která z pohledu studenta vypadá stejně jako klasické slovní úlohy s krátkou tvořenou odpovědí (odpověď spočívá v zadání jednoho výrazu do prázdného políčka). Při vytváření numerické úlohy se pouze zadává otázka a výsledek. V zadání lze také definovat jednotku a stanovit přijatelnou chybu, tj. definovat souvislý interval odpovědí, které jsou považovány za správné. Jako oddělovač desetinných míst je nutné při vytváření úloh i zadávání odpovědí použít tečku.

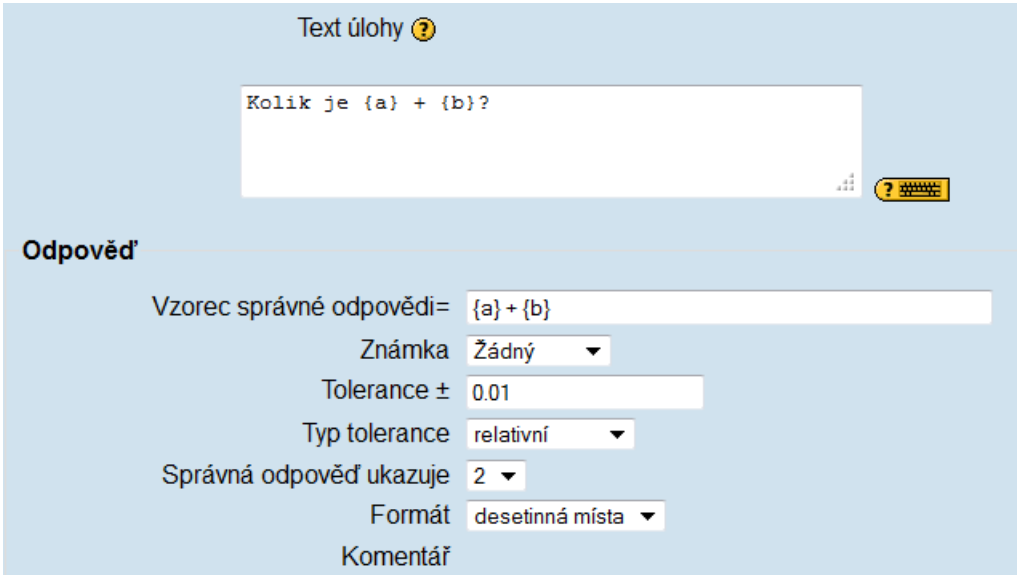
[9] Popisy a náhledy uživatelského prostředí těchto typů úloh odpovídají defaultnímu nastavení systému ve verzi 1.9.11+.



Obr. 9: Náhled rozhraní pro tvorbu numerické úlohy (Vlastní zpracování)

2.1.1.2 Vypočítávaná úloha

Vypočítávané úlohy jsou složitějším typem matematických úloh umožňující použití *masek*, které se při vstupu do testu nahradí konkrétními hodnotami. Do otázky a vzorce správné odpovědi se vkládají proměnné (libovolného pojmenování), které jsou při vstupu do testu nahrazeny konkrétní náhodně vygenerovanou hodnotou v rámci zadaných parametrů datové sady (datasetu). Masky, které je možné v tomto typu úlohy použít, se nastaví nebo vytvoří na další stránce pomocí nástroje pro úpravu vypočítávaných úloh.



Obr. 10: Náhled části rozhraní pro tvorbu vypočítávané úlohy (Vlastní zpracování)

Povolenými operátory v těchto maskách jsou:

- operátory základních početních operací: **+** (součet), **-** (rozdíl), ***** (součin), **/** (podíl), **%** (modulo)
- některé matematické funkce, které se vyskytují v jazyce PHP:
  - 24 funkcí s jedním argumentem: **abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, deg2rad, exp, expm1, floor, log, log10, log1p, rad2deg, round, sin, sinh, sprt, tan, tanh**
  - dvě funkce se dvěma argumenty: **atan2, pow**
  - funkce **min** a **max**, které vyžadují dva nebo více argumentů
  - funkce **pi()** (*Calculated question type – MoodleDocs*, 2010)

U všech funkcí (včetně funkce pi) musí být argumenty uzavřeny v závorkách. V jedné masce je možné použít více funkcí (např.: sin({a}) + cos({b}) * 2) i vkládání jedné funkce do druhé (např.: cos(deg2rad({a} + 90)). (Tamtéž.)

Stejně jako u numerických úloh lze nastavit toleranci, tj. povolit rozpětí hodnot, v jehož rámci jsou všechny odpovědi považovány za správné. Podle stanovení tolerančního intervalu lze rozlišit tři typy tolerance:

- **Relativní tolerance** určuje hranice intervalu pomocí vynásobení hodnoty správné odpovědi nastavenou hodnotou. Např. jestliže je správná odpověď v testu vypočtena na hodnotu 200 a tolerance nastavena na hodnotu 0.5, pak bude hodnota tolerance 100 a za správnou je považována každá odpověď v intervalu 100 až 300 (200 ± 100). Tato metoda je vhodná, pokud se velikost správné odpovědi může výrazně lišit v závislosti na různých hodnotách masek.
- **Nominální tolerance** určuje interval pomocí rozdílu stanovené hodnoty k hodnotě správné odpovědi. Např. při stejném zadání jako u předchozího příkladu musí správná odpověď ležet mezi hodnotou 199,5 a 200,5 (200 ± 0,5). Jedná se o nejjednodušší typ tolerance, který je ovšem vhodný pouze tehdy, pokud jsou rozdíly mezi různými správnými odpověďmi malé.
- **Geometrická tolerance** je tolerance, u níž se každá hranice intervalu zjišťuje odlišným způsobem. Např. při stejném zadání jako u předchozího příkladu se horní hranice intervalu tolerance určí jako 200 + 0,5*200 a je stejná jako v případě tolerance relativní. Dolní hranice se počítá jako 200/(1 + 0,5). Správná odpověď tak musí ležet mezi hodnotou 133,33 a 300. Tato metoda je vhodná pro komplexní výpočty vyžadující velké tolerance. U podobných úloh by bylo možné použít pro výpočet horní hranice intervalu relativní toleranci o hodnotě 1 nebo vyšší, ta však není vhodná pro dolní hranici intervalu, protože by ve všech případech povolovala jako správnou odpověď nulu. (Tamtéž.)

Pole **Platné číslice** určuje pouze to, jakou formou má být správná odpověď uvedena v přehledu nebo v protokolech. Např. pokud je volba nastavena na hodnotu 3, bude správná odpověď 13.333 uvedena jako 13.3; 1236 bude uvedeno jako 1240; 23 bude uvedeno jako 23.0 atp. (Tamtéž.)

2.2 Aplikační vrstva

Každý z modulů v Moodlu je tvořen samostatnou složkou souborů dostupnou v adresáři *mod*, která obsahuje povinné a volitelné elementy pro všechny moduly a dále soubory, které jsou pro jednotlivé moduly unikátní. (*Development:Modules – MoodleDocs,* 2010)

V této části bude proveden podrobnější popis souborů s PHP skripty a dalších souborů tvořících jádro modulu Test a modulů Numerická úloha a Vypočítávaná úloha. Soubory jsou dostupné ve složce *mod → quiz* a *question → type → numerical / calculated*.

Modul Test je tvořen celkem 49 soubory. Většina z nich je ve formátu PHP, ostatní jsou ve formátu HTML, XML a JavaScript (a ikony ve formátu GIF). Povinné soubory, které jsou obsažené v každém modulu, jsou v následujícím výčtu označeny hvězdičkou, volitelné dvěma hvězdičkami. (V modulu nejsou použity volitelné komponenty pro administrátorské nastavení, pro zápis proměnných a některé další volitelné funkce, které lze přidat do *lib.php*.)

- Soubor *lib.php** – jádro modulu obsahující soupis použitých **funkcí**:
  - modul_install() – je volána při instalaci modulu,
  - modul_add_instance() – přidání nové instance modulu[10],
  - modul_update_instance() – update existující instance modulu,
  - modul_delete_instance() – vymazání instance modulu,
  - modul_user_outline() – vrací přehled činností, které provedl konkrétní uživatel s instancí modulu,
  - modul_user_complete() – vrací detailní soupis činností, které provedl konkrétní uživatel s instancí modulu,
  - modul_get_view_actions() / modul_get_post_actions() – třídění akcí v logovacích tabulkách (volané v *course/report/participation/index.php*).

[10] *Instancí modulu* se rozumí konkrétní použití modulu v jednotlivých kurzech (viz objekt jakožto instance třídy v OOP).

- Soubor *locallib.php* – knihovna **specifických funkcí** použitých v tomto v modulu.
- **Databázové soubory** – soubory uložené ve složce *db*:
    - *db/access.php** – definuje možnosti modulu (např. zobrazení informace o existenci modulu, editace testů, vyplňování testů, sledování reportů atp.).
    - *db/install.xml** – definuje strukturu databázových tabulek pro všechny typy databázových systémů. Je použit během instalace modulu.
    - *db/upgrade.php** – definuje změny ve struktuře databáze během upgradu modulu (pro pomoc při řešení případných problémů s upgradem).
    - soubor *mysql.php* – definuje databázi MySQL pro zastaralé verze Moodlu.
    - soubor *postgres7.php* – definuje databázi PostgreSQL pro zastaralé verze Moodlu.
- **Zálohovací a obnovovací soubory** *backuplib.php***, *restorelib.php***, *restorelibpre15.php.*
- **Reportovací soubory:**
    - *report.php* – obsahuje pluginy pro zobrazování reportů o testech.
        - */analysis/report.php* – pro tvorbu tabulek s reporty o otázkách,
        - */grading/report.php* – report s odpověďmi napomáhající při manuálním známkování testů,
        - */overview/report.php* – report o pokusech studentů při plnění testů (dále jen „pokusech“),
        - */overview/overviewgraph.php* – zobrazení grafů vztahujících se k výsledkům testů,
        - */overview/overviewsettings_form.php* – zobrazení formuláře s přehledem výsledků,
        - */regrade/report.php* – skript pro změnu známkování u všech pokusů,

        - *default.php* – skript pro vytváření reportů o kvízech (je volán z *quiz/report.php*),
        - *reportlib.php* – obsahuje funkce pro různé typy reportů.
- Soubor *review.php* a *reviewquestion.php* – zobrazují přehledy o jednotlivých pokusech.
- Soubor *reviewoptions.html* ve složce */report* – formulář pro nastavení možností přehledů.
- Soubor *version.php** – definice metadat o verzi modulu.
- Konfigurační soubor *mod_form.php** – formulář pro konfiguraci (např. nastavení názvů a rozmístění polí) modulu.
- Soubor *defaults.php** – umožňuje definovat defaultní hodnoty proměnných. Soubor je obsažen v upgrade_activity_modules v *lib/adminlib.php**. Definuje pole $defaults. Tyto hodnoty jsou poté načteny do konfigurační tabulky (config table). (Při nastavení $defaults['_use_config_plugins'] na hodnotu true jsou hodnoty místo toho načítány do tabulky config_plugins.)
- Soubor *index.php** – stránka s výčtem všech instancí modulu.
- Soubor *view.php** – zobrazuje jednotlivé instance modulu.
- Soubor *attempt.php* – zobrazuje informace o pokusech u jednotlivých instancí modulu.
- Soubor *attempt_close_js.php*
- Soubor *comment.php* – komentáře související s činností při pokusech.
- Soubor *edit.php* – definuje stránku pro editaci testů (přidávání otázek atp.).
- Soubor *editlib.php* – obsahuje funkce použité při editaci testů (jsou volány v souboru *edit.php*).
- Soubor *jstimer.php* – generuje plovoucí hodiny zobrazující, kolik času zbývá k vyplnění testu.
- Soubor *quiz.js* – speciální skripty pro ukládání uživatelských dat, příspěvků na fórech atp.
- Soubor *tabs.php* – nastavení záložek na jednotlivých stránkách testu v závislosti na možnostech uživatele.
- Další soubory s podpůrnými funkcemi: *grade.php*, *pagelib.php*, *protect_js.php*.

- Ikony: *icon.gif** (ikona modulu o rozměrech 16 × 16 px) a dalších 9 ikon použitých v modulu ve složce *pix.*

Jednotlivé typy úloh (Question types) používají společné i specifické funkce. Modul (Question type) Vypočítávaná úloha (*moodle → question → type → calculated*) má následující strukturu:

- složka *db*
  - soubor *install.xml* – definice databázové struktury modulu
  - soubor *mysql.php* **–** definuje databázi MySQL pro zastaralé verze Moodlu
  - soubor *postgres7.php* **–** definuje databázi PostgreSQL pro zastaralé verze Moodlu
  - soubor *upgrade.php* – definuje změny ve struktuře databáze během upgradu modulu
- soubor *edit_calculated_form.php* – definice formuláře pro editaci úlohy
- soubor *questiontype.php* – definice PHP tříd modulu
- soubor *version.php* – metadata o verzi modulu
- soubor *icon.gif* – ikona modulu

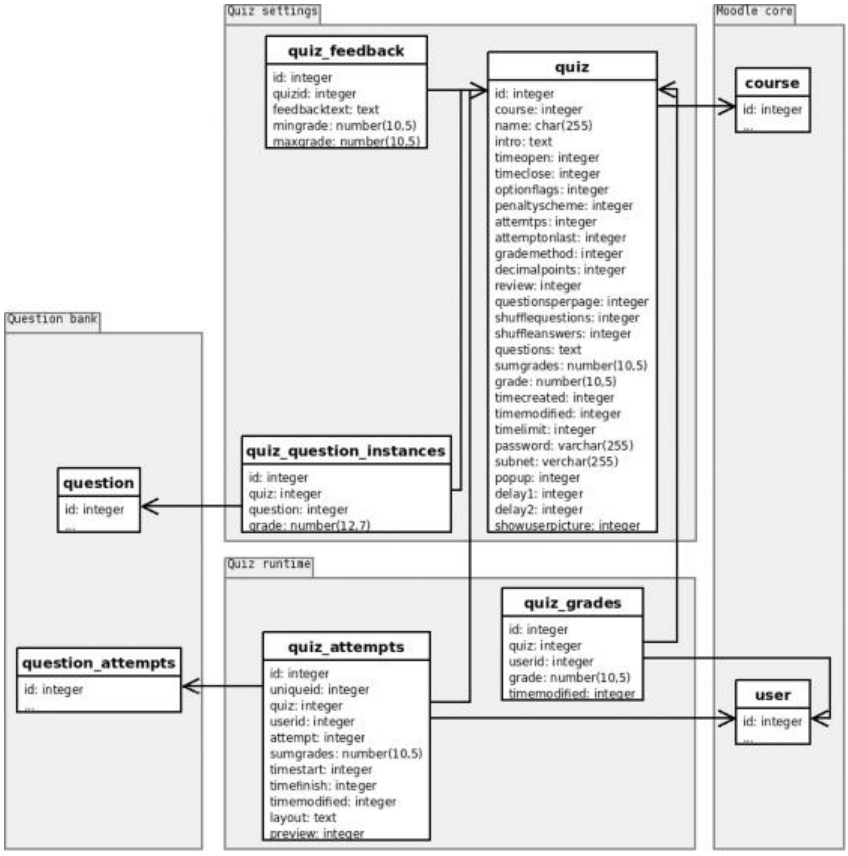
Struktura ostatních Question types je obdobná, přičemž moduly pro jednotlivé typy úloh se mezi sebou liší specifickými funkcemi, stavbou formulářů atp. Ve složce *type* jsou kromě těchto modulů další společné soubory:

- *question.html* – definice celkového layoutu otázky
- *edit_question_form.php* – třídy definující základ formulářů pro editaci úloh
- *questiontype.php* – defaultní třídy pro libovolný modul s typem úlohy

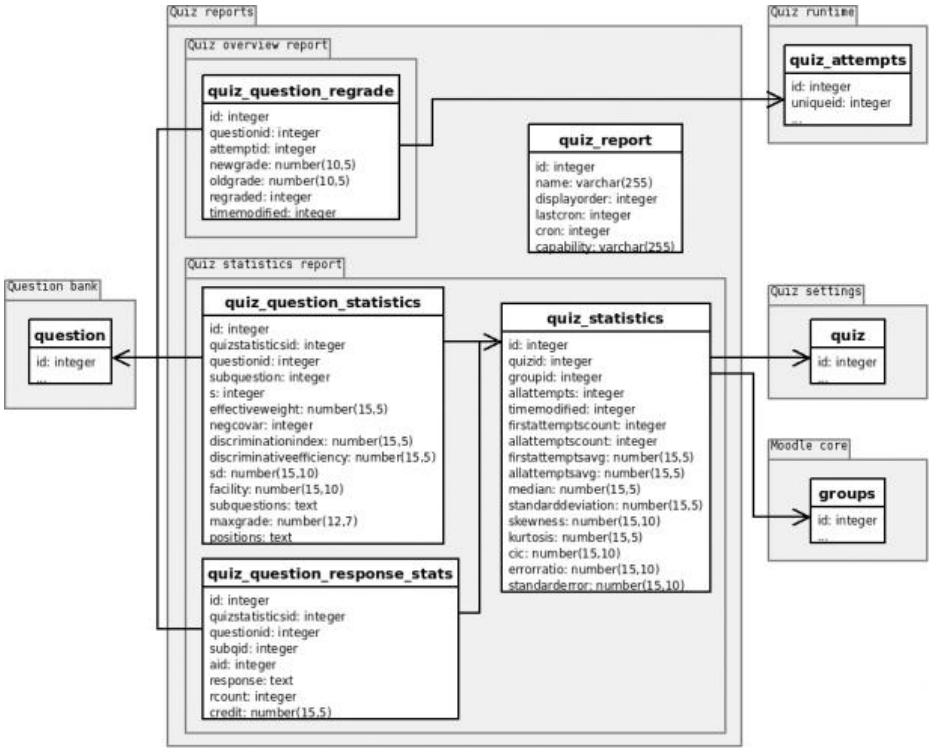
2.3 Databázové schéma

Databázové tabulky používané v modulu Test jsou rozděleny do několika hlavních skupin: nastavení (Quiz settings), provoz (Quiz runtime), banka úloh (Question bank), reporty (Quiz reports) a jádro Moodlu (Moodle core). V Quiz settings se uchovávají data o nastavení modulu a v Quiz runtime data o jednotlivých činnostech při plnění testů studenty. Otázky a informace o odpovědích jsou shromažďovány v bance úloh. S Moodle core jsou tabulky propojeny daty o uživatelích a kurzech. (*Development:Quiz database structure – MoodleDocs*, 2009) Toto schéma je platné pro defaultní typ úlohy (Question type), konkrétní typy úloh mohou využívat další specifické tabulky.

Podrobná databázová schémata jsou zobrazena na následujících obrázcích.



Obr. 11: Databázové schéma modulu Test – 1. část
(Zdroj: *Development:Quiz database structure – MoodleDocs*, 2009)



Obr. 12: Databázové schéma modulu Test – 2. část
(Zdroj: *Development:Quiz database structure – MoodleDocs*, 2009)

Numerická úloha využívá tabulky question_numerical a question_numerical_units. Tabulka question_numerical doplňuje tabulku question_answers, zahrnuje hodnotu tolerance pro odpovědi na jednotlivé otázky. Tabulka question_numerical_units doplňuje tabulku quiz_question_answers a obsahuje jednotku vztahující se k číselné odpovědi. Obdobně je to u vypočítávané úlohy, která využívá tabulku question_calculated ukládající hodnotu tolerance a další položky specifické pro tento typ úlohy. Pro ukládání ostatních hodnot (např. odpovědí) postačuje u těchto typů úloh defaultní databázové schéma. (*Development:Numerical question type – MoodleDocs*, 2007)

2.4 Zhodnocení možností systému a význam tvorby nového modulu

Na základě předchozí analýzy lze usoudit, že uživatelské rozhraní a funkce implementované v modulech matematických typů úloh nejsou dostatečné pro efektivní tvorbu testů zaměřených na výpočet daně z příjmů.

Nejvhodnějším nástrojem pro tvorbu testů tohoto typu je tzv. vypočítávaná úloha. U ní lze zadat příklad pomocí masek s proměnnými a vyhodnotit ho na základě automatického porovnání vypočítaného výsledku s výsledkem získaným od studenta. Tento typ zadávání úloh je však vzhledem ke způsobu výpočtu daně z příjmu, který je založený na rozhodovacích algoritmech, nevhodný – neumožňuje do masky pro výpočet tyto algoritmy zadat a případné ruční zadávání masek výpočtu pro různé typy úloh je příliš zdlouhavé. Systém je souhrnně zhodnocen v následující SWOT analýze (silné a slabé stránky modulu jsou hodnoceny především s ohledem na Vypočítávanou úlohu).

Tab. 1: SWOT analýza využití systému při testování výpočtu daně (Vlastní zpracování)

Silné stránky (**S**trenghts)	Slabé stránky (**W**eaknesses)
• v zadání lze použít libovolné masky s proměnnými (adaptibilita na legislativní změny výpočtu daně) • lze nastavit toleranci pro výpočet (u výpočtů daní vhodné z důvodu možnosti tolerance chyb vzniklých zaokrouhlováním) • výsledek vypočítaný podle zadané masky je automaticky porovnán s výsledkem získaným od studenta	• zdlouhavé a složité zadávání příkladu • nemožnost zadávat univerzální masku pro výpočet (různé způsoby výpočtu daně)
Příležitosti (Opportunities)	Hrozby (**T**hreats)
• do systému lze začlenit nový modul (Question type) umožňující efektivnější zadávání úlohy	• problémy s vývojem, příp. instalací nového modulu (různé verze Moodlu) • legislativní změny související s výpočtem daně vyžadují zásahy do kódu

Pro účely testování daňových výpočtů bude vyvinut nový modul (typ úlohy – tzv. Question type plugin). Hierarchie tohoto modulu v systému je následující: *question → type → konkrétní Question type*. Jeho funkcionalita bude vycházet z některých programovacích technik využitých v defaultních typech úloh v Moodlu a uživatelské rozhraní (editor pro tvorbu úlohy) bude uzpůsobeno pro přímé zadávání hodnot relevantních pro výpočet daně.

3 Návrh řešení a vypracování

Vzhledem k nedostatečnosti základních přednastavených modulů v Moodlu bude navržen a realizován nový modul (tzv. Question type plugin) s upravenými vlastnostmi, který umožní splnit všechny zadané požadavky. Výsledek vypracování bude stručně představen také z pohledu uživatelů (učitele a studenta).

3.1 Instalační a aplikační část

3.1.1 Instalace systému a umístění zdrojových souborů modulu

K instalaci systému byl zvolen server webového poskytovatele, který podporuje PHP a MySQL.[11] Po zřízení domény je na serveru potřeba vytvořit databázi, a poté přistoupit k instalaci Moodlu. Na server byl nahrán standardní Moodle balíček.[12] Po nahrání souborů je nutné spustit instalační skript, a poté nastavit webovou adresu, adresář Moodle, datový adresář (ten je nutné nastavit ručně) a připojení k již vytvořené databázi. V následujícím kroku proběhne automatická kontrola programového prostředí serveru.[13] Závěrečnými instalačními kroky jsou nahrání konfiguračního souboru do kořenového adresáře instalace Moodlu, potvrzení licenčních podmínek, nastavení účtu správce a údajů o kurzu.

[11] Jedná se o freehosting 000webhost.com, který podporuje PHP 5 s vypnutým safe_mode a disponuje dvěma databázemi MySQL.
Vývoj modulu lze provádět také na offline instalaci systému, ke které lze použít např. softwarový balíček XAMPP zahrnující předkonfigurované instalace Apache, MySQL databáze, PHP a dalších technologií.

[12] Instalační balíčky jsou dostupné na adrese http://moodle.org/download/. Moodle na VUT je současně (stav k 24. 3. 2011) používán ve verzi 1.9.5+. Pro předejití případných problémů se zpětnou kompatibilitou je vhodné zvolit instalaci Moodle ve verzi ze stejné řady (1.9.11+, stav k 24. 3. 2011), neboť od verze 2.0 se mírně mění hierarchie modulů v systému, knihovny funkcí atp. Instalační balíčky pro Question type plugin lze sice podle dokumentace Moodle použít pro všechny verze počínaje verzí 1.7, ve verzi 2.x však při kontrole zásuvných modulů došlo k chybovému hlášení, ze kterého vyplynulo, že v modulu jsou volány funkce, které již nejsou v nové verzi implementovány.

[13] Zvolený hosting nepodporuje doporučené komponenty OpenSSL a XML-RPC, ty jsou však užitečné pro síťové funkce Moodlu, a nejsou tedy pro účely vypracování nezbytné.

Pro účely tvorby nového modulu je nutné stáhnout balíček obsahující základní soubory, do kterých budou vpisovány zdrojové kódy definující modul.[14] (*Development:Question type plugin how to – MoodleDocs*, 2009) Ne všechny soubory jsou pro modul potřebné, některé z nich mohou být odstraněny (pro vývoj Question type modulu nebude využit např. soubor *styles.css*). Struktura modulu je následující (soubory, které se významově shodují se soubory tvořícími strukturu modulu Vypočítávaná úloha, jsou označeny hvězdičkou):

- složka *db*
  - soubor *install.xml** – prázdný soubor určený pro definici struktury databázových tabulek
- složka *lang*
  - složka *cs_utf8*[15]
    - složka *help*
      - složka *modul*
        - soubor *modul.html* – zápis použitých proměnných
    - soubor *qtype_modul.php* – zápis názvů zobrazujících se v uživatelském rozhraní pro jednotlivé proměnné
- soubor *questiontype.php** – definice PHP tříd modulu
- soubor *edit_modul_form.php** – definice formuláře pro editaci úlohy
- soubor *display.html* – šablona definující zobrazení úloh studentům
- složka *simpletest*
  - soubor *testquestiontype.php* – testování knihoven funkcí modulu
- soubor *version.php** – metadata o verzi modulu
- soubor *script.js* – prázdný soubor pro zápis přídavných JavaScript kódů
- soubor *styles.css* – prázdný soubor pro zápis přídavných kaskádových stylů
- soubor *icon.gif** – ikona modulu
- soubor *README.txt* – informace pro vývojáře

[14] Balíček souborů je dostupný na adrese http://moodle.org/mod/data/view.php?d=13&rid=443.

[15] V původním balíčku souborů je složka *en_utf8*, pro českou jazykovou lokalizaci je nutné ji přejmenovat.

Po stažení balíčku kódů následuje série kroků potřebná pro přípravu souborů na zápis kódu:

- Návrh identifikátoru a jména modulu a jejich zaznamenání do souborů (identifikátor nesmí obsahovat číslice a velká písmena; jako identifikátor byl zvolen řetězec *´dpfo´*, jako název *´DPFO´*). Dále vepsání jména a e-mailové adresy vývojáře do souborů.
- Výběr ikony reprezentující modul (*icon.gif*). 
  Přesunutí složky obsahující soubory tvořící nový modul do adresáře *question/type* ve zdrojových kódech systému (*moodle* → *question* → *type* → *dpfo*). Ve vyšších verzích (2.x) může administrátor manipulovat s modulem prostřednictvím webové sekce Správa typů testových úloh. (Tamtéž.)

3.1.2 Vývoj modulu

Vývoj modulu se skládá z těchto kroků:

- Definice potřebných databázových tabulek (*db/install.xml*).
- Definice formuláře pro editaci úlohy (*edit_dpfo_form.php*).
- Tvorba šablony definující zobrazení úloh studentům (*display.html*).
- Implementace specifických PHP tříd (*questiontype.php*), které se týkají:
  - Ukládání, načítání a mazání dat z databáze při tvorbě úlohy.
  - Ukládání, načítání a mazání dat z databáze a doby, po kterou je úloha zobrazena v testu.
  - Zobrazení úlohy studentovi.
  - Nakládání s odpověďmi: hodnocení odpovědí, zajišťování správných odpovědí atd.
  - Zálohování a ukládání jednotlivých úloh.
  - Import a export jednotlivých úloh.
- Zápis použitých proměnných do souboru *qtype_dpfo.php*.[16]
- Tvorba nápovědy (*lang/cs_utf8/help/dpfo/dpfo.html*). (Tamtéž.)

Důležité části kódy jsou obsaženy v příslušných kapitolách. Úplné zdrojové kódy souborů jsou uvedeny v přílohách.

[16] Soubor je možné uložit také do složky *moodledata/lang/cs_utf8*.

3.1.2.1 Databázové tabulky (*install.xml*)

Pro modul je možné vytvořit speciální databázové tabulky nebo adresář *db* smazat a nechat ukládat data vztahující se k úlohám do předdefinované databáze. Pro modul DPFO bude potřeba vytvořit tabulku, do níž se uloží hodnoty specifické pro výpočet daně zadávané ve formuláři pro editaci úlohy.

Kód pro vytvoření tabulky se umístí do souboru *install.xml*. Po instalaci modulu (viz dále) je nutné tabulku nahrát a uložit do databáze (v editoru XMLDB dostupném z menu pro správu stránek). Úspěšnou instalaci tabulek lze ověřit navštívením stránky s chybovými hlášeními (Notifications page – admin/index.php).

Tabulka vytvořená pro modul DPFO bude obsahovat položky načtené z formuláře pro editaci úlohy (příjem, typ pracovněprávního vztahu atp.) a položky potřebné pro vazbu s ostatními tabulkami.

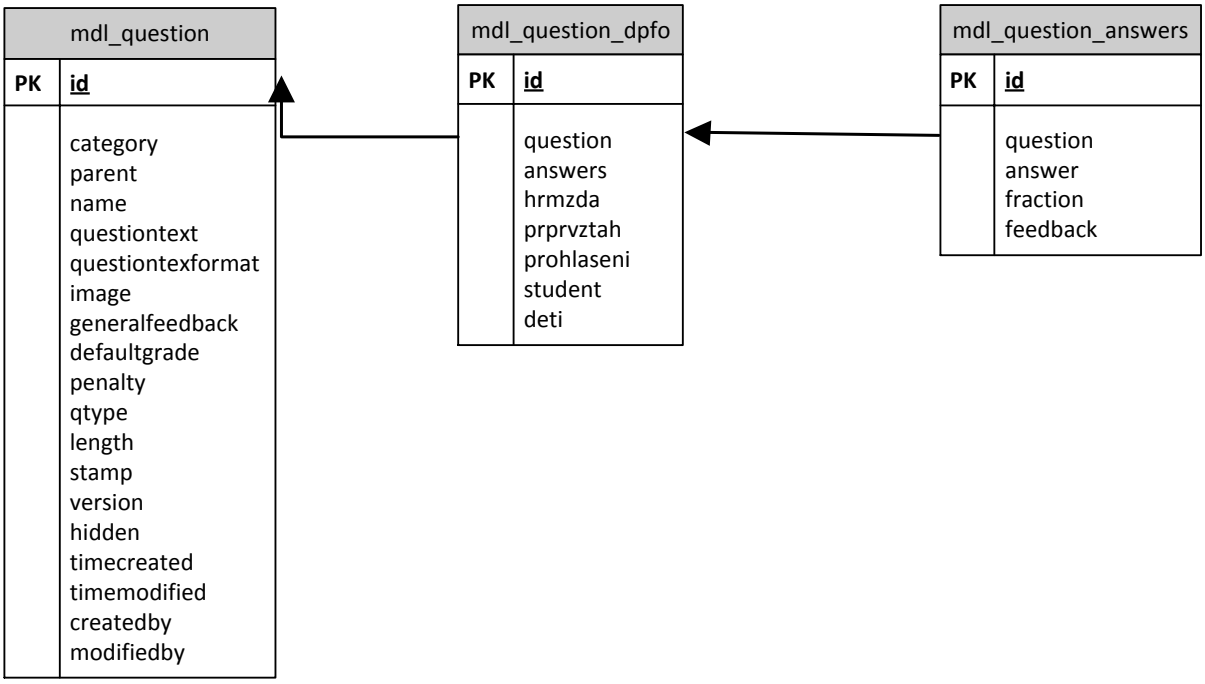
```
<TABLE NAME="question_dpfo" COMMENT="položky tabulky pro DPFO">
    <FIELDS>
     <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true"
      SEQUENCE="true" ENUM="false" NEXT="question"/>
     <FIELD NAME="question" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true"
      DEFAULT="0" SEQUENCE="false" ENUM="false" PREVIOUS="id" NEXT="answer"/>
     <FIELD NAME="answer" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true"
      DEFAULT="0" SEQUENCE="false" ENUM="false" PREVIOUS="question" NEXT="hrmzda"/>
     <FIELD NAME="hrmzda" TYPE="int" LENGTH="6" NOTNULL="true" UNSIGNED="true"
      SEQUENCE="false" ENUM="false" PREVIOUS="answer" NEXT="prprvztah"/>
     <FIELD NAME="prprvztah" TYPE="char" LENGTH="4" NOTNULL="true" UNSIGNED="true"
      SEQUENCE="false" ENUM="false" PREVIOUS="hrmzda" NEXT="prohlaseni"/>
     <FIELD NAME="prohlaseni" TYPE="bool" LENGTH="1" NOTNULL="true" UNSIGNED="true"
      SEQUENCE="false" ENUM="false" PREVIOUS="prprvztah" NEXT="student"/>
     <FIELD NAME="student" TYPE="bool" LENGTH="1" NOTNULL="true" UNSIGNED="true"
      SEQUENCE="false" ENUM="false" PREVIOUS="prohlaseni" NEXT="deti"/>
     <FIELD NAME="deti" TYPE="int" LENGTH="2" NOTNULL="true" UNSIGNED="true"
      SEQUENCE="false" ENUM="false" PREVIOUS="student"/>

     //unsigned = bez znaménka - pouze kladné hodnoty
     //sequence = zachování pořadí (1,2,3)
```

```
    //enum = výčtový typ - množina konstant (enumerátorů) - proměnná nabývá hodnoty jedné z nich
    //previous / next = funkce, která posouvá interní ukazatel o jeden prvek dopředu / dozadu (pro prázdné prvky nebo prvky s indexem 0 vrací "false" => zajišťuje provázanost prvků - správný průchod polem jen při vyplnění všech prvků)
   </FIELDS>
   <KEYS>
    <KEY NAME="primary" TYPE="primary" FIELDS="id" NEXT="question"/>
    // primární klíč
    <KEY NAME="question" TYPE="foreign" FIELDS="question" REFTABLE="questions"
     REFFIELDS="id" PREVIOUS="primary"/>
    // cizí klíč - odkaz do tabulky questions
   </KEYS>
   <INDEXES>
    <INDEX NAME="question" UNIQUE="false" FIELDS="question"/>
    // index slouží pro rychlé hledání v tabulce
   </INDEXES>
  </TABLE>
 </TABLES>
</XMLDB>
```

Druhou možností je vytvořit tabulku přímo ve vlastní databázi (zde v jazyce SQL):

```
CREATE TABLE `a9695686_data`.`mdl_question_dpfo` (
`id` INT( 10 ) UNSIGNED NOT NULL AUTO_INCREMENT ,
`question` INT( 10 ) UNSIGNED NOT NULL DEFAULT '0',
`answer` INT( 10 ) UNSIGNED NOT NULL DEFAULT '0',
`hrmzda` INT( 6 ) UNSIGNED NOT NULL ,
`prprvztah` CHAR( 4 ) NOT NULL ,
`prohlaseni` BOOL NOT NULL ,
`student` BOOL NOT NULL ,
`deti` INT( 2 ) UNSIGNED NOT NULL ,
PRIMARY KEY ( `id` ) ,
INDEX ( `answer` )
)
ENGINE = MYISAM
```

Umístění tabulky mdl_question_dpfo v databázové struktuře systému je zobrazeno na následujícím schématu:



Obr. 13: Tabulka mdl_question_dpfo v rámci databázové struktury systému (Vlastní zpracování)

3.1.2.2 Formulář pro editaci úlohy (*edit_dpfo_form.php*)

Definice formuláře pro editaci úlohy se provede v souboru *edit_dpfo_form.php* rozšířením defaultní třídy formulářů pro editaci úloh (**class** question_edit_dpfo_form **extends** question_edit_form; třída question_edit_dpfo_form je definována v souboru */question/type/edit_question_form.php*). Zahrnuje definici funkcí pro přidání formulářových polí, načtení dat do formuláře a ověření vstupních dat. V případě, že by defaultní formulář obsahoval zbytečná pole nebo by bylo potřeba definovat formulář jiným způsobem, je možné definovat formulář jako zcela novou třídu bez vazby na defaultní formulář.

Defaultní formulář obsahuje tato **pole**:

- výběr kategorie úlohy
- název úlohy (povinná položka)
- zadání textu úlohy ve formě HTML editoru
- standardní počet bodů za úlohu
- penalizační faktor
- zadání obecné reakce ve formě HTML editoru

Všechna tato pole lze ve formuláři ponechat. Pro účely efektivního zadávání úloh budou přidána nová pole určená pro vstup hodnot relevantních pro výpočet daně, tj.:

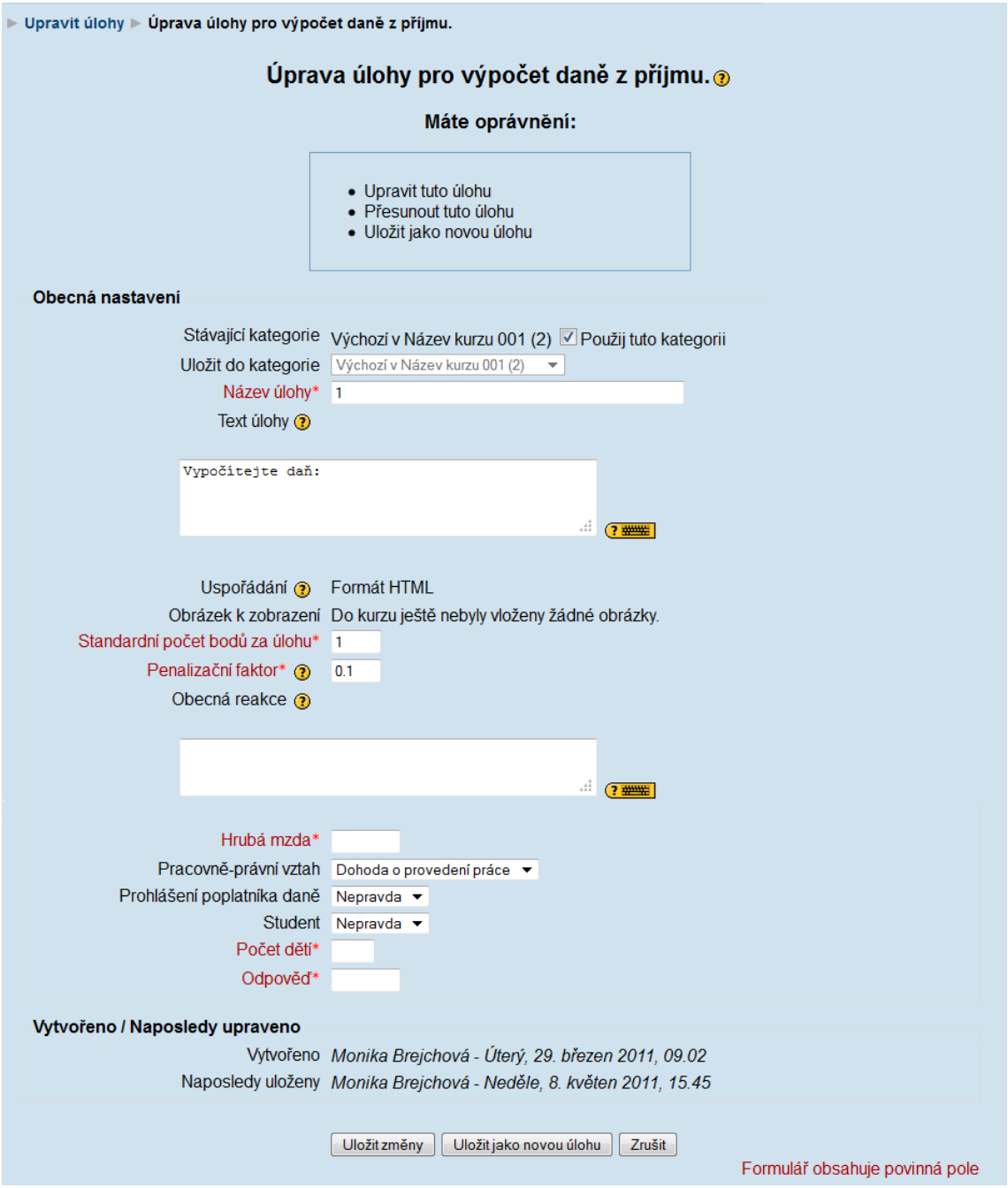
- hrubá mzda (v Kč)
- pracovněprávní vztah (možnosti: dohoda o provedení práce, dohoda o pracovní činnosti, pracovní poměr)
- Prohlášení poplatníka daně (možnosti: ano, ne)
- student (možnosti: ano, ne)
- počet dětí
- správná odpověď

Pole jsou definována pomocí metody $mform->addElement. Následuje nastavení datových typů ($mform->setType), defaultních hodnot ($mform->setDefault) a označení povinných parametrů ($mform->addRule). [17]

[17] $ označuje proměnnou, -> je operátor, který vrací kontext objektu. (KREJČÍ, 2006, s. 25, 29)

- datové typy:
  - PARAM_INT – celá čísla
  - PARAM_NUMBER – reálná čísla
  - PARAM_BOOL – logický datový typ
  - PARAM_RAW – text
- příkaz pro vytvoření pole s určitými parametry: array('klíč'=> hodnota)
- funkce pro získání řetězce (označení proměnné u políčka ve formuláři): get_string('označení', 'soubor s proměnnými')
- metody:
  - přidání elementu: $mform->addElement('typ pole', 'označení', get_string(), array())
  - nastavení datového typu proměnné: $mform->setType('označení', datový typ);
  - nastavení defaultní hodnoty proměnné: $mform->setDefault('označení', 'defaultní hodnota');
  - nastavení povinného vyplnění pole: $mform->addRule('označení', null, 'required', null, 'client');

```
function definition_inner(&$mform) {
// definice dalsich formularovych poli

  // hruba mzda
    $mform->addElement('text', 'hrmzda', get_string('hrmzda', 'qtype_dpfo'), array('size' => 6));
    $mform->setType('hrmzda', PARAM_NUMBER);
    $mform->addRule('hrmzda', null, 'required', null, 'client');

  // pracovne-pravni vztah
    $menu = array(get_string('dopp', 'qtype_dpfo'), get_string('dopc', 'qtype_dpfo'), get_string('pp',
    'qtype_dpfo'));
    $mform->addElement('select', 'prprvztah', get_string('prprvztah', 'qtype_dpfo'), $menu);

  // Prohlaseni poplatnika dane
    $mform->addElement('select', 'prohlaseni', get_string('prohlaseni', 'qtype_dpfo'),
    array(0 => get_string('false', 'quiz'), 1 => get_string('true', 'quiz')));

  // student
    $mform->addElement('select', 'student', get_string('student', 'qtype_dpfo'),
    array(0 => get_string('false', 'quiz'), 1 => get_string('true', 'quiz')));

  // pocet deti
    $mform->addElement('text', 'deti', get_string('deti', 'qtype_dpfo'), array('size' => 2));
    $mform->setType('deti', PARAM_INT);
    $mform->addRule('deti', null, 'required', null, 'client');

  // spravna odpoved
    $mform->addElement('text', 'odpoved', get_string('odpoved', 'qtype_dpfo'), array('size' => 45));
    $mform->setType('odpoved', PARAM_NUMBER);
    $mform->setDefault('odpoved', 'Spravna odpoved bude vypocitana automaticky.');
  }
```



Obr. 14: Náhled formuláře pro editaci úlohy typu DPFO (Vlastní zpracování)

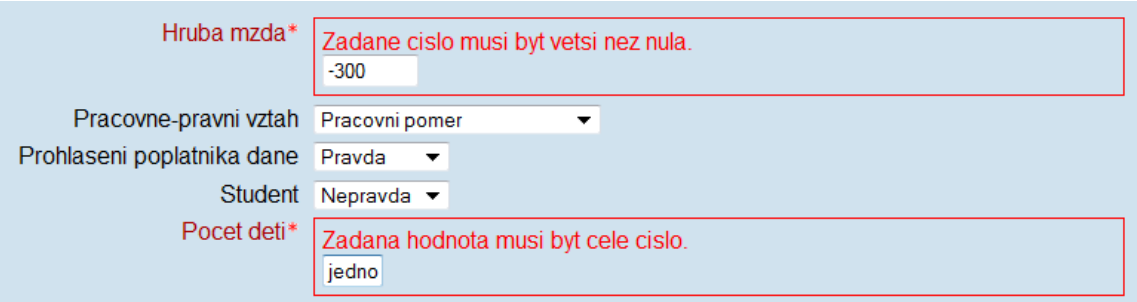
Načtení dat (těch, u kterých to není definováno v core kódu) **do formuláře** po jeho otevření se provede následovně:

```
function set_data($question) {
 // nacteni dat do formulare
        if (!empty($question->options)) {
            $answers = $question->options->answers;
            foreach ($answers as $answer){
                    $question->hrmzda = $answer->hrmzda;
                    $question->prprvztah = $answer->prprvztah;
                    $question->prohlaseni = $answer->prohlaseni;
                    $question->student = $answer->student;
                    $question->deti = $answer->deti;
                    $question->odpoved = $answer->answer;
                }
            }

            parent::set_data($question);
        }
```

U polí s otevřenou možností odpovědi je nutné **ošetřit vstupní hodnoty**:

- hrubá mzda (v Kč) – pouze celá čísla větší než nula
- počet dětí – pouze celá čísla počínaje hodnotou 0

```
function validation($data) {
// osetreni vstupnich hodnot
      $errors = array();

      //hruba mzda musi byt vetsi nez nula
       $hrmzda = $data['hrmzda'];
       if ($hrmzda <= 0) {
          $errors['hrmzda'] = get_string('vetsineznula','qtype_dpfo');
      }

      //hruba mzda musi byt cele cislo
      if (!(is_int($hrmzda))) {
          $errors['hrmzda'] = get_string('celecislo','qtype_dpfo');
      }

      //pocet deti musi byt vetsi nebo roven nule
      $deti = $data['deti'];
       if ($deti < 0) {
          $errors['deti'] = get_string('asponnula','qtype_dpfo');
      }

      //pocet deti musi byt cele cislo
      if (!(is_int($deti))) {
          $errors['deti'] = get_string('celecislo','qtype_dpfo');
      }

      if ($errors) {
         return $errors;
      } else {
         return true;
      }
   }
```



Obr. 15: Ukázka validace dat ve formuláři (Vlastní zpracování)

3.1.2.3 Zobrazení úlohy v testu (*display.html*)

V souboru *display.html* je prostřednictvím HTML a PHP kódu definováno, jak se zobrazí spuštěná úloha studentovi v průběhu testu. Zobrazení elementů v core kódu je definované v *question.html*, odkud je voláno metodou print_question v souboru *preview.php*.

Defaultně jsou zobrazeny tyto elementy:

- číslo úlohy
- text zadání
- maximální počet bodů
- tlačítka pro odeslání úlohy a další činnosti související s průběhem testu
- odkaz na titulní stránku kurzu systému a na dokumentaci
- odkaz na profil uživatele a možnost jeho odhlášení

Obr. 16: Zobrazení úlohy v testu před úpravou kódu (Vlastní zpracování)

K těmto prvkům je třeba přidat:

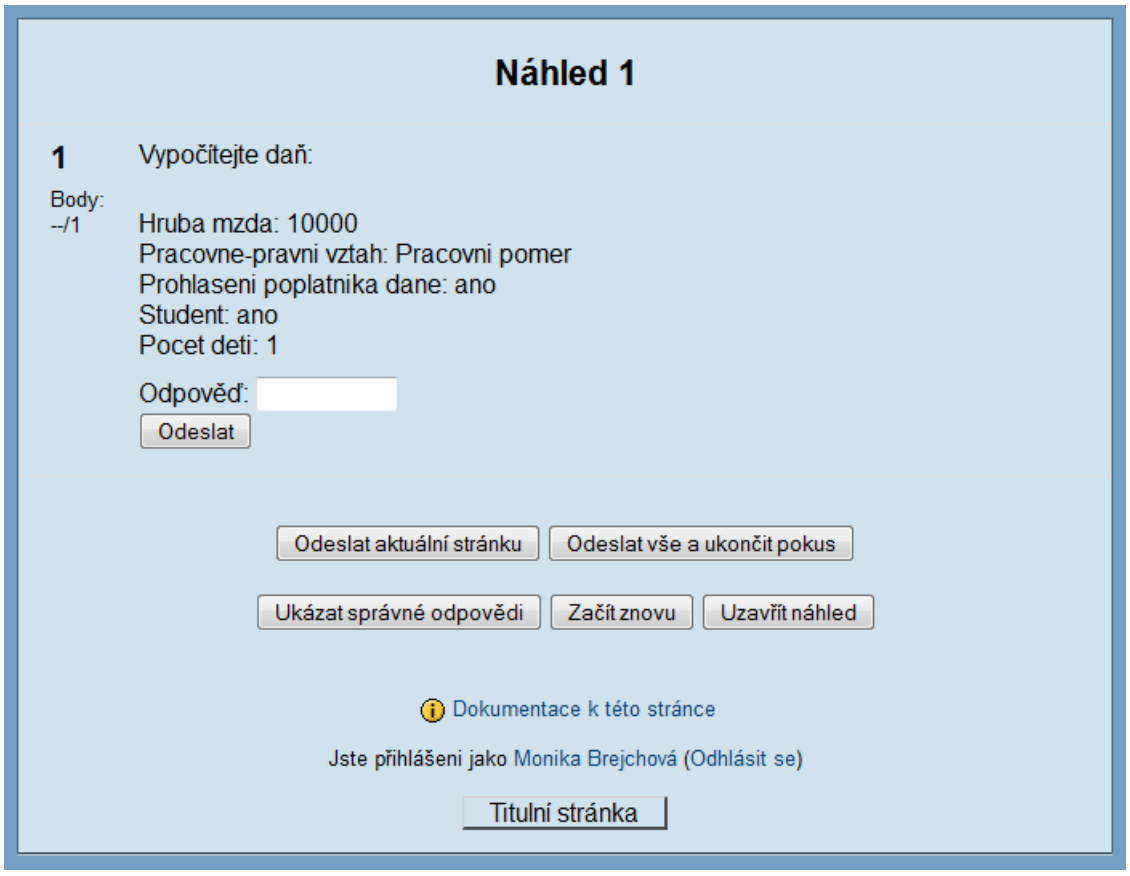
- zadání otázky s hodnotami jednotlivých položek, které vstupují do výpočtu daně (tj. hrubá mzda, typ pracovněprávního vztahu, informace o Prohlášení poplatníka daně, studiu a počtu dětí poplatníka)
- pole pro vyplnění odpovědi

Zadání otázky je reprezentováno proměnnou $questiontext (příkaz <?php echo $questiontext; ?>), která je definovaná v souboru *questiontype.php* ve funkci function print_question_formulation_and_controls. V této funkci je nutné definovat také proměnné s hodnotami, které tvoří součást zadání, a vstupují do výpočtu daně (tato úprava funkce je součástí další kapitoly). Zadané hodnoty jsou zobrazeny prostřednictvím následujícího kódu.

```
<div class="hrmzda">
                <?php
                   echo get_string('hrmzda', 'qtype_dpfo').': ';
                   echo $zadanihrmzda;
                 ?>
           </div>

           <div class="prprvztah">
                <?php
                   echo get_string('prprvztah', 'qtype_dpfo').': ';
                   echo $zadaniprprvztah;
                 ?>
           </div>

           <div class="prohlaseni">
                <?php
                   echo get_string('prohlaseni', 'qtype_dpfo').': ';
                   echo $zadaniprohlaseni;
                 ?>
           </div>

           <div class="student">
                <?php
                   echo get_string('student', 'qtype_dpfo').': ';
                   echo $zadanistudent;
                 ?>

           </div>

           <div class="deti">
                <?php
                    echo get_string('deti', 'qtype_dpfo').': ';
                   echo $zadanideti;
                 ?>
           </div>
```

Pole pro zadání odpovědi lze definovat v prvku takto:

```
<div class="odpoved">

    <?php
     echo get_string('answer', 'quiz').': ';
    ?>

    //pole pro vepsání odpovědi
    <input type="text" class=""   name=""   value=""  size="10"/>

</div>
```



Obr. 17: Zobrazení úlohy v testu po úpravě kódu (Vlastní zpracování)

3.1.2.4 Funkce modulu (*questiontype.php*)

Tento soubor obsahuje všechny funkce specifické pro modul DPFO:

- načtení dat potřebných pro zjištění odpovědi z databáze
  (function get_question_options(&$question))
- funkce pro výpočet daně
  (function qtype_dpfo_calculate_answer)
- uložení dat z formuláře pro editaci úlohy
  (do tabulek mdl_question_answer a mdl_question_dpfo)
- načtení zadaných hodnot do testu
  (function print_question_formulation_and_controls(&$question, &$state, $cmoptions, $options))

Funkce pro **načtení dat z databáze**:

```
function get_question_options(&$question) {
// ziskani dat a defaultnich hodnot z databaze
      global $CFG;
      // global = globalni promenna (zajisti viditelnost promenne uvnitr funkce)
      //$CFG - CFG (context-free grammar) = formalni gramatika, ve ktere maji vsechny pravidla tvar
"neterminal -> retezec terminalu nebo neterminalu"

       if (!$question->options->answers = get_records_sql(
                    "SELECT a.*, d.hrmzda, d.prprvztah, d.prohlaseni, d.student, d.deti " .
                    "FROM {$CFG->prefix}question_answers a, " .
                    "    {$CFG->prefix}question_dpfo d " .
                    "WHERE a.question = $question->id " .
                    "    AND   a.id = d.answer " .
                    "ORDER BY a.id ASC")) {
         notify('Chyba: Chybejici data pro dpfo ulohu ' . $question ->id . '!');
         return false;
      }
      }
```

Funkce pro **výpočet správné odpovědi** (daně z příjmu):

```
function qtype_dpfo_calculate_answer($hrmzda, $prprvztah, $prohlaseni, $student, $deti) {
 //vypocet spravne odpovedi

 $dpfo = 0;
 if ($prprvztah == 'dpp' && $hrmzda <= 5000 && $prohlaseni == false)
 //vypocet srazkove dane
 {dpfo == 0.15 * $hrmzda;}

 else
  //vypocet zalohove dane
 {
 //vypocet superhrube mzdy
 $supermzda = $hrmzda + (0.25 * $hrmzda) + (0.09 * $hrmzda);
 //vypocet zalohy na dani
 $zaloha = 0.15 * $supermzda;
 //vypocet slevy na dani
 $sleva = 0;
 if ($prohlaseni == true)
 {$sleva = 1970;}
 if ($student == true)
 {$sleva = $sleva + 335;}
 if ($sleva > $zaloha)
 {$sleva = $zaloha;}
 //vypocet danoveho bonusu
 if ($deti > 0)
 {$bonus = $deti * 967;}
 if ($bonus > 4835)
 {$bonus = 4835;}
  //vypocet dane
 $dpfo = $zaloha - $sleva - $bonus;
 }
 // vysledek, ktery vraci funkce
 return $dpfo;
}
```

Funkce pro **ukládání dat** z formuláře a vypočítané odpovědi:

```
function save_question_options($question) {
        // nacteni jiz ulozenych dat
            if (!$oldanswers = get_records('question_answers', 'question', $question->id, 'id ASC')) {
                $oldanswers = array();
            }

            if (!$oldoptions = get_records('question_dpfo', 'question', $question->id, 'answer ASC'))
{
                $oldoptions = array();
            }

            // ulozeni odpovedi do mdl_question_answers
             $answer = new stdClass;
             $vysledek = 0;
             $vysledek = qtype_dpfo_calculate_answer($question->hrmzda, $question->prprvztah,
             $question->prohlaseni, $question->student, $question->deti);
             $answer->answer = $vysledek;
             $answer->question = $question->id;

             if ($oldanswer = array_shift($oldanswers)) {  // aktualizace odpovedi
                 $answer->id = $oldanswer->id;
                 if (! update_record("question_answers", $answer)) {
                     $result->error = "Nelze aktualizovat odpoved! (id=$answer->id)";
                     return $result;
                 }
               } else { // vlozeni odpovedi
                 if (! $answer->id = insert_record("question_answers", $answer)) {
                     $result->error = "Nelze ulozit odpoved!";
                     return $result;
                 }
               }

             // ulozeni hodnot do mdl_question_dpfo
             if (!$options = array_shift($oldoptions)) {
```

```
            $options = new stdClass;
        }

         $options->question = $question->id;
         $options->answer = $answer->id;
         $options->hrmzda = $question->hrmzda;
         $options->prprvztah = $question->prprvztah;
         $options->prohlaseni = $question->prohlaseni;
         $options->student = $question->student;
         $options->deti = $question->deti;

        if (isset($options->id)) { // aktualizace
            if (! update_record('question_dpfo', $options)) {
                $result->error = "Could not update dpfo options! (id=$options->id)";
                return $result;
            }
        }
        else { // vlozeni
            if (! insert_record('question_dpfo', $options)) {
                $result->error = "Could not insert quiz numerical options!";
               return $result;
            }
       }
        }
```

Část funkce pro **načtení zadaných hodnot do testu**:

```
function print_question_formulation_and_controls(&$question, &$state, $cmoptions, $options) {
        global $CFG;
       //nacteni zadanych hodnot
       if (!$moznosti = get_records('question_dpfo', 'question', $question->id, 'answer ASC')) {
            $moznosti = array();
        }
         if (!$zadani = array_shift($moznosti)) {
             $zadani = new stdClass;
            }
```

```
        $zadanihrmzda = $zadani->hrmzda;
        $zadaniprprvztah = $zadani->prprvztah;
         if ($zadaniprprvztah = '0') {
            $zadaniprprvztah = get_string('dopp', 'qtype_dpfo');
         }
          if ($zadaniprprvztah = '1') {
            $zadaniprprvztah = get_string('dopc', 'qtype_dpfo');
         }
          if ($zadaniprprvztah = '2') {
            $zadaniprprvztah = get_string('pp', 'qtype_dpfo');
         }
        $zadaniprohlaseni = $zadani->prohlaseni;
         if ($zadaniprohlaseni = '0') {
            $zadaniprohlaseni = 'ne';
         }
         if ($zadaniprohlaseni = '1') {
            $zadaniprohlaseni = 'ano';
         }
        $zadanistudent = $zadani->student;
         if ($zadanistudent = '0') {
            $zadanistudent = 'ne';
         }
         if ($zadanistudent = '1') {
            $zadanistudent = 'ano';
         }
        $zadanideti = $zadani->deti;

      include("$CFG->dirroot/question/type/dpfo/display.html");
    }
```

Všechny funkce jsou zahrnuty ve třídě, která je rozšířením třídy s funkcemi pro všechny typy úloh (class dpfo_qtype extends default_questiontype). Po uzavření této třídy je potřeba iniciovat typ úlohy v systému (question_register_questiontype(new dpfo_qtype());).

3.1.2.5 Zápis proměnných (*qtype_dpfo.php*)

Pro korektní zobrazování proměnných ve formulářích je nutné do souboru *qtype_dpfo.php* uloženého v jazykové složce (pro českou lokalizaci) zapsat všechny proměnné, které dosud nebyly v jazykových složkách (nebo v Moodle „core“ kódu) deklarovány. Proměnné se zapisují ve tvaru: $string[‘označení proměnné] = ‘název proměnné’. Jsou volány z ostatních souborů prostřednictvím funkce get_string().

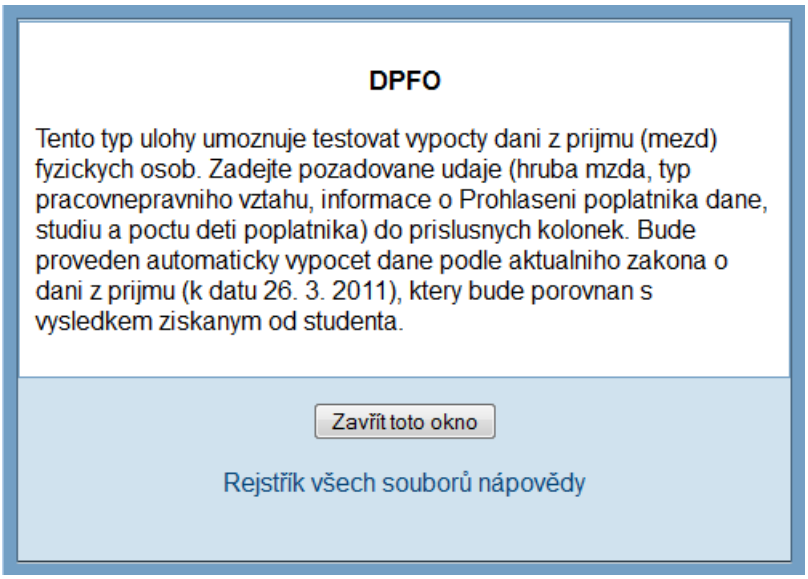
```
$string['addingdpfo'] = 'Přidání úlohy pro výpočet daně z příjmu.';
$string['editingdpfo'] = 'Úprava úlohy pro výpočet daně z příjmu.';
$string['dpfo'] = 'DPFO';
$string['dpfosummary'] = 'Typ úlohy pro výpočet daně z příjmu fyzických osob.';
$string['hrmzda'] = 'Hrubá mzda';
$string['prprvztah'] = 'Pracovně-právní vztah';
$string['dopp'] = 'Dohoda o provedení práce';
$string['dopc'] = 'Dohoda o pracovní činnosti';
$string['pp'] = 'Pracovní poměr';
$string['prohlaseni'] = 'Prohlášení poplatníka daně';
$string['student'] = 'Student';
$string['deti'] = 'Počet dětí';
$string['odpoved'] = 'Odpověď';
$string['vetsineznula'] = 'Zadané číslo musí být větší než nula.';
$string['celecislo'] = 'Zadaná hodnota musí být celé číslo.';
$string['asponnula'] = 'Zadané číslo musí být větší nebo rovno nule.';
```

Kontrolu a editaci proměnných a textových řetězců lze provést také přes administrátorské webové rozhraní (Správa stránek → Jazyk → Úprava překladu).

3.1.2.6 Nápověda (*dpfo.html*)

V nápovědě má být obsažen souhrn základních informací o modulu, návod, jak postupovat při tvorbě úloh atp. Tato nápověda je zobrazena po kliknutí na symbol nápovědy zobrazené v záhlaví formuláře pro tvorbu úlohy.

Nápovědu lze obdobně jako textové řetězce editovat také z administrátorského webového rozhraní (Správa stránek → Jazyk → Kontrola překladu).



Obr. 18: Nápověda k použití modulu (Vlastní zpracování)

3.2 Testování a uživatelská část

3.2.1 Testování modulu

Pro ověření správné funkčnosti modulu je nutné provést alespoň základní testování, které by mělo obsahovat následující kroky:

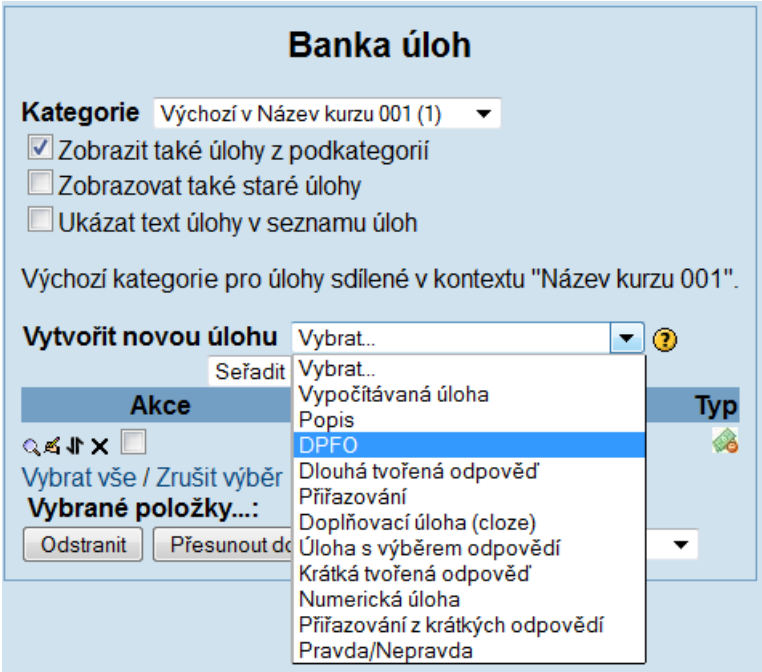
- vytvoření několika otázek daného typu (s kombinováním výrazně rozdílných zadání)
- kontrola uložení zadaných údajů z formuláře (i při změně)
- zapisování dat, která nejsou validní, do formuláře
- zobrazení náhledů vytvořených otázek
- kontrola reakcí na správné a nesprávné odpovědi (v různých prohlížečích)

- vložení vytvořených otázek do tří typů testů: test s adaptivními otázkami (při nesprávné odpovědi je znovu nabídnuta možnost odpovědi i v rámci jednoho pokusu), test s neadaptivními otázkami (možnost jediného pokusu odpovědi) a test, který uchová odpovědi studenta a zobrazí je při příštím pokusu
- vyplnění testu v roli studenta
- kontrola reportů
- záloha kurzu a jeho obnovení ve formě nového kurzu – kontrola, zda se otázky zkopírovaly korektně (včetně aktualizace případných odkazů)
- export otázek do všech typů formátů a následný import
- kontrola přístupnosti (*Development:Question type plugin how to – MoodleDocs*, 2011)

3.2.2 Rozhraní pro učitele

Instalace modulu je provedena administrátorem (zdrojové kódy modulu jsou přidány k ostatním zdrojovým kódům systému).

Po instalaci modulu je již možné modul využívat běžným způsobem jako ostatní „typy úloh“ při tvorbě nových úloh. Datový typ se zobrazí při tvorbě nové otázky v roletovém menu typů úloh (na stránce kurzu volba Úlohy v menu Správa → Vytvořit novou úlohu). Do zobrazeného formuláře zadá učitel (příp. jiný uživatel s relevantními oprávněními) potřebná data a potvrdí uložení úlohy kliknutím na tlačítko Uložit změny.



Obr. 19: Přidání úlohy typu DPFO do banky úloh (Vlastní zpracování)

Test se v kurzu vytvoří v režimu úprav kliknutím na možnost Přidat činnost → Test v hlavních blocích uprostřed stránky. Po zadání základních údajů se test uloží a zobrazuje se v patřičném bloku. Chce-li učitel vložit vytvořenou úlohu do testu, klikne z hlavní stránky kurzu na název testu, na záložce Upravit zaškrtne vybrané otázky ze seznamu zobrazeného v pravé části stránky a zvolí možnost Vložit do testu.

Výsledky testů lze sledovat v modulu Známky (dostupný z menu správa na hlavní stránce kurzu), příp. na záložce výsledky (na stránce zobrazené po kliknutí na název testu). Výsledky lze filtrovat, zobrazovat pro jednotlivé otázky i pro celý test, příp. kurz, a analyzovat dle různých kritérií (první nebo poslední pokus, nejvyšší známka atp.). Testy lze rovněž znovu oznámkovat, příp. známkovat ručně.

3.2.3 Rozhraní pro studenty

Test spustí student kliknutím na název testu a následně na tlačítko Pokusit se o zvládnutí testu. Po zodpovězení otázek test uloží (pro možnost dalšího navázání) nebo odešle. Po odeslání testu se mu zobrazí prohlídka příslušného pokusu, kde jsou zobrazeny základní údaje o pokusu (časové údaje, dosažené body a známka) a výsledky jednotlivých odpovědí. Výsledky může také sledovat v modulu Známky (dostupný z hlavní stránky kurzu v menu Správa), příp. opětovným kliknutím na název kurzu.

4 Ekonomické zhodnocení a přínosy

Zhodnocení efektivity zpracování a následného užívání vytvořeného kurzu je z důvodu existence finančních i nefinančních přínosů obtížně uchopitelné. Například ve firmě má zavedení e-learningu s relevantními výukovými nástroji pozitivní vliv na rozvoj znalostí jejích pracovníků, které jsou klíčovým zdrojem k podnikání a faktorem úspěchu firmy.

Ekonomické zhodnocení lze kromě výčtu nákladů a výnosů rozšířit o další charakteristiky. Využívá se například tzv. **principu ekvivalentního přínosu**, který nesrovnává náklady a výnosy, ale porovnává rozdíl mezi náklady na zavedení e-learningu a náklady dosud užívaných vzdělávacích metod. (KUBEŠ, s. 28)

Náklady vynaložené na inovaci vzdělávacího systému představují investici, kterou lze hodnotit standardními metodami (ROI, ROE a ROK). **ROI** (**Return of Investment**) představuje poměr zisku k vynaloženým nákladům. Kvantifikace některých efektů je však natolik obtížná, že není doporučeno tento ukazatel u hodnocení e-learningových systémů používat. **ROE** (**Return of Expectation**) hodnotí míru naplnění předem stanovených očekávání (např. o kolik se zvýší produktivita zaměstnanců po absolvování kurzu). **ROK** (**Return of Knowledge**) sleduje zlepšení indikátorů souvisejících s úrovní vzdělání zaměstnanců (např. názory zákazníků). (Tamtéž, s. 21–22)

Specifikem elektronického vzdělávání je fakt, že přináší vysoké počáteční náklady, ale snižuje náklady na vzdělání jednotlivce a celkové náklady klesají s rostoucím počtem studentů. Rozhodnutí, zda je ekonomicky výhodnější použití klasické nebo elektronické formy vzdělávání lze pak učinit na základě **analýzy bodu zvratu**, tj. počtu studentů, od jehož hodnoty se elektronické vzdělávání začne finančně vyplácet. (Tamtéž, s. 30–31)

Kromě výnosů z užívání lze posuzovat i výnosy z prodeje tohoto systému (resp. kurzu vytvořeného v systému s využitím rozšiřujícího modulu). Veškerý software vytvořený v LMS Moodle lze vzhledem k zaštítění systému licencí GNU užívat ke komerčním účelům, je však nutné aplikovat podmínky licence i na tyto odvozené produkty (šířit i tento software jako volně šiřitelný Open Source software, tj. zpřístupnit kupujícímu přístup ke zdrojovým kódům). (*Licence – MoodleDocs*, 2009) Toto omezení je kompenzováno odbouráním nákladů nutných na vývoj vlastního, příp. nákup komer-

čního LMS. Náklady jsou tak redukovány na zdroje vynaložené na instalaci systému, tvorbu a administraci kurzu, příp. vytváření doplňkových modulů a úprav kódu.

V následujících podkapitolách bude uveden výčet faktorů, které se podílejí na tvorbě nákladů a výnosů pro konkrétní kurz v LMS Moodle (včetně integrovaného modulu pro výpočet daně). U nákladů bude uvedeno rovněž vyčíslení. Vyčíslení výnosů (resp. porovnání nákladů na e-learning a na klasickou formu vzdělání – viz princip ekvivalentního přínosu) závisí na tom, v které konkrétní organizaci je kurz implementován, a do jaké míry pokryje její vzdělávací potřeby.

4.1 Náklady

Mezi obecné faktory vstupující do tvorby e-learningového kurzu lze zařadit následující (KUBEŠ, s. 26):

Přímé (náklady zaměstnavatele)

- **příprava kurzu**
  - náklady na přípravu obsahu (jednorázové)
  - nákup placených informačních zdrojů (jednorázové)
  - příprava elektronické podoby (jednorázové)
    - převedení materiálů do elektronické podoby
    - nasazení kurzu na server
  - náklady na pravidelnou aktualizaci kurzu
- **provoz kurzu**
  - provoz serveru
    - správa serveru
    - licence na používaný software
  - plat administrátora / lektora (hodnocení materiálů, konzultace…) / moderátora v diskuzních místnostech atp.
- **náklady na organizaci kurzu**
  - administrativa
    - spojená s přípravou kurzu (jednorázové)
    - spojená s výběrem a přihlašováním účastníků do kurzu

Nepřímé (náklady zaměstnance)

- ušlá produktivita / zisk (resp. vyplácená mzda) za dobu výuky
- ušlá produktivita pracovního místa („neproduktivní“ činnost)

Pro kurz vytvořený v LMS Moodle (včetně integrovaného modulu) lze navrhnout následující rozpis přímých nákladů:

Tab. 2: Náklady na vývoj modulu a kurz vytvořený v LMS Moodle (Vlastní zpracování)

činnost	**poznámka**	**částka**
vývoj modulu	100 hodin (j[18]), 200 Kč/hod.[19]	20 000 Kč
náklady na přípravu obsahu	100 hodin (j), 140 Kč/hod.[20]	14 000 Kč
převedení materiálů do el. podoby	50 hodin (j), 150 Kč/hod.[21]	7 500 Kč
nasazení kurzu na server	5 hodin (j), 150 Kč/hod.	750 Kč
náklady na aktualizace kurzu	5 hodin (m[22]), 150 Kč/hod.	750 Kč
správa serveru	placený webhosting splňující potřebné technické parametry (měsíčně)	100 Kč[23]
plat administrátora	4 hodiny (m), 200 Kč/hod.[24]	800 Kč
administrativa spojená s přípravou kurzu	20 hodin (j), 120 Kč/hod.[25]	2400 Kč
jednorázové náklady		44 650 Kč
měsíční náklady		1650 Kč

[18] jednorázově
[19] Průměrný plat programátora je cca 217 Kč/hod. (Zdroj: http://www.platy.cz/platy/informacni-technologie)
[20] Průměrný plat lektora je cca 139 Kč/hod. (Zdroj: http://www.platy.cz/platy/skolstvi-vzdelavani-veda-vyzkum)
[21] Průměrný plat webmastera je cca 149 Kč/hod. (Zdroj: http://www.platy.cz/platy/informacni-technologie)
[22] měsíčně
[23] Např. webhosting s kapacitou 1 GB za 1068 Kč/rok. (Zdroj: http://www.hostingy.cz/)
[24] Průměrný plat systémového administrátora je cca 196 Kč/hod.
(Zdroj: http://www.platy.cz/platy/informacni-technologie)
[25] Průměrný plat administrativního pracovníka je cca 116 Kč/hod.
(Zdroj: http://www.platy.cz/platy/administrativa)

Náklady na vývoj nového modulu budou kompenzovány úbytkem času vynaloženého na zadávání testovacích úloh pro výpočet daně.

Při použití komerční LMS aplikace by bylo nutné zahrnout do výpočtu licenční poplatek, příp. další náklady závisející na konkrétním typu LMS (např. plat lektora při použití systému nepodporujícího automatické vyhodnocování testů). Při použití vlastní LMS aplikace by byly náklady vyšší o náklady vynaložené na vývoj systému.

4.2 Výnosy

Při efektivním užívání e-learningu jako řešení nebo jako podpůrného prostředku vzdělávacích potřeb v podniku přináší jeho implementace úspory nákladů např. v těchto oblastech (KUBEŠ, s. 27):

- výdaje na cestování, ubytování a stravné
- snížení nepřítomnosti pracovníků na pracovišti
- výdaje na lektory, učebny a technické prostředky
- rizika spojená s odložením vzdělávání z kapacitních důvodů
- odbourání nákladů kvůli nevzdělávání

Určení přínosů plynoucích ze vzdělání zaměstnanců je obtížnější, protože některé z nich se mohou projevit pouze dlouhodobě nebo zprostředkovaně. Některé reálné výsledky vzdělávání lze měřit po absolvování kurzu různými metodami a na různých úrovních. (Tamtéž, s. 22–24). V úvahu je však nutné vzít i objektivně nesnadno postižitelné aspekty jako je např. upevňování firemní kultury a postojů či předávání praktických zkušeností.

Závěr

V práci byl zdokumentován postup vedoucí ke vzniku nového modulu (datového typu) integrovatelného do systému Moodle a využitelného v kurzu určeném pro testování daňových výpočtů. Východiskem byla analýza předdefinovaných modulů datových typů, na jejímž základě byl vyvíjen nový modul. Vývoj spočíval v návrhu databázových tabulek, formulářů a funkcí obsahujících algoritmy pro výpočet daně z příjmu uzpůsobeném tak, aby nový modul umožnil efektivní zadávání a vyhodnocování úloh testujících výpočet daně.

V závěru práce bylo provedeno zhodnocení ekonomických aspektů celého řešení, ve kterém byly vyčísleny náklady na vývoj nového modulu a tvorbu kurzu a které poukázalo na obecné přínosy integrace e-learningových kurzů do podnikových informačních systémů.

Hlavním přínosem navrženého modulu je zrychlení zadávání testovacích úloh a možnost jejich automatického vyhodnocování. Modul může být na základě uvedené metodiky snadno rozšířen na výpočet daní jiného typu příjmů, případně lze metodiku tvorby modulu využít pro usnadnění vývoje dalších modulů obdobného typu.

Seznam použitých zdrojů

1. *Apache – MoodleDocs* [online]. c2011 [cit. 2011-03-08]. Dostupné z URL: <http://docs.moodle.org/en/Apache>.
2. BORONCZYK, T. et al. *PHP 6, MySQL, Apache : vytváříme webové aplikace*. 1. vyd. Brno : Computer Press, 2009. 816 s. ISBN 978-80-251-2767-4.
3. *Calculated question type – MoodleDocs* [online]. c2010 [cit. 2011-03-11]. Dostupné z URL: <http://docs.moodle.org/en/Calculated_question_type#Tolerance>.
4. *CVS for Administrators – MoodleDocs* [online]. c2010 [cit. 2010-11-24]. Dostupné z URL: <http://docs.moodle.org/en/CVS_for_Administrators>.
5. *CVS – Open Source Version Control* [online]. c2010 [cit. 2010-11-24]. Dostupné z URL: <http://www.nongnu.org/cvs/>.
6. *Development:Database schema introduction – MoodleDocs* [online]. c2010 [cit. 2011-03-02]. Dostupné z URL: <http://docs.moodle.org/en/Development:Database_schema_introduction>.
7. *Development:Developer documentation – MoodleDocs* [online]. c2011 [cit. 2011-02-15]. Dostupné z URL: <http://docs.moodle.org/en/Development:Developer_documentation>.
8. *Development:JavaScript guidelines – MoodleDocs* [online]. c2011 [cit. 2011-03-05]. Dostupné z URL: <http://docs.moodle.org/en/Development:JavaScript_guidelines>.
9. *Development:lib/formslib.php – MoodleDocs* [online]. c2009 [cit. 2011-03-01]. Dostupné z URL: <http://docs.moodle.org/en/Development:lib/formslib.php>.
10. *Development:lib/moodle.php – MoodleDocs* [online]. c2011 [cit. 2011-03-02]. Dostupné z URL: <http://docs.moodle.org/en/Development:lib/moodlelib.php>.
11. *Development:lib/weblib.php – MoodleDocs* [online]. c2007 [cit. 2011-03-02]. Dostupné z URL: <http://docs.moodle.org/en/Development:lib/weblib.php>.
12. *Development:Modules – MoodleDocs* [online]. c2010 [cit. 2011-03-06]. Dostupné z URL: <http://docs.moodle.org/en/Development:Modules>.

13. *Development:Numerical question type – MoodleDocs* [online]. c2007 [cit. 2011-03-06]. Dostupné z URL: <http://docs.moodle.org/en/Development:Numerical_question_type#Database_tables>.
14. *Development:PHPXref – MoodleDocs* [online]. c2009 [cit. 2011-03-09]. Dostupné z URL: <http://docs.moodle.org/en/Development:PHPXref >.
15. *Development:Question type plugin how to – MoodleDocs* [online]. c2009 [cit. 2011-03-04]. Dostupné z URL: <http://docs.moodle.org/en/How_to_write_a_question_type_plugin>.
16. *Development:Quiz database structure – MoodleDocs* [online]. c2009 [cit. 2011-03-06]. Dostupné z URL: <http://docs.moodle.org/en/Development:Quiz_database_structure>.
17. *Development:Using the File API – MoodleDocs* [online]. c2010 [cit. 2011-03-01]. Dostupné z URL: <http://docs.moodle.org/en/Development:Using_the_file_API>.
18. *Development:Using XMLDB – MoodleDocs* [online]. c2008 [cit. 2011-02-28]. Dostupné z URL: <http://docs.moodle.org/en/Development:Using_XMLDB>.
19. *Development:XMLDB introduction – MoodleDocs* [online]. c2010 [cit. 2011-03-15]. Dostupné z URL: <http://docs.moodle.org/en/Development:XMLDB_introduction>.
20. HOUSHOLDER, C. *Moodle: the story behind the name* [online]. c2006 [cit. 2010-12-3]. Dostupné z URL: <http://media.www.smithsophian.com/media/storage/paper587/news/2006/09/21/News/Moodle.The.Story.Behind.The.Name-2303995.shtml>.
21. *HTML editor – MoodleDocs* [online]. c2010 [cit. 2011-03-08]. Dostupné z URL: <http://docs.moodle.org/en/HTML_editor>.
22. *HTML in Moodle – MoodleDocs* [online]. c2009 [cit. 2011-03-08]. Dostupné z URL: <http://docs.moodle.org/en/HTML_in_Moodle>.
23. HUNT, T. *A basic introduction to the Moodle architecture* [online]. c2010 [cit. 2011-04-14]. Dostupné z URL: <http://www.slideshare.net/tjh1000/a-basic-introduciton-to-the-moodle-architecture-5442122>.
24. *Instalace – MoodleDocs* [online]. c2010 [cit. 2011-02-03]. Dostupné z URL: <http://docs.moodle.org/cs/Instalace>.

25. *Integrations – MoodleDocs* [online]. c2010 [cit. 2011-02-15]. Dostupné z URL: <http://docs.moodle.org/en/Integrations>.
26. KREJČÍ, L. *PHP – Kapesní přehled*. 1. vyd. Brno : Computer Press, 2006. 108 s. ISBN 80-251-0808-2.
27. KUBEŠ, T. *Posouzení vybraných technických, ekonomických a personálních aspektů e-vzdělávání v Learning Solution systému SAP v prostředí Československé obchodní banky, a. s.* Praha : Vysoká škola ekonomická v Praze, Fakulta podnikohospodářská, 2009. 73 s. Vedoucí diplomové práce PhDr. Petr Beroušek.
28. KUČEROVÁ, D. *Podepsat nebo nepodepsat Prohlášení při vedlejší činnosti?* [online]. c2008 [cit. 2011-03-14]. Dostupné z URL: <http://www.podnikatel.cz/clanky/zjednoduseni-podminek-u-dohody-o-provedeni-prace/>.
29. LECKY-THOMSON, E., NOWICKY, S. D. *PHP 6 : programujeme profesionálně*. 1. vyd. Brno : Computer Press, 2010. 718 s. ISBN 978-80-251-3127-5.
30. *Licence – MoodleDocs* [online]. c2009 [cit. 2011-03-01]. Dostupné z URL: <http://docs.moodle.org/en/License>.
31. *Moodle XML format – MoodleDocs* [online]. c2010 [cit. 2010-03-05]. Dostupné z URL: <http://docs.moodle.org/en/Moodle_XML>.
32. *MOODLE.CZ* [online]. c2010 [cit. 2010-11-24]. Dostupné z URL: <http://moodle.cz/>.
33. *Moodle.org: open-source community-based tools for learning* [online]. c2011 [cit. 2011-03-15]. Dostupné z URL: <http://moodle.org/>.
34. *MySQL :: The world's most popular open source database* [online]. c2010 [cit. 2010-11-22]. Dostupné z URL: <http://www.mysql.com>.
35. *PHP: Hypertext Preprocessor* [online]. c2010 [cit. 2010-11-22]. Dostupné z URL: <http://www.php.net/>.
36. *Rukověť správce – MoodleDocs* [online]. c2010 [cit. 2010-11-24]. Dostupné z URL: <http://docs.moodle.org/cs/Rukověť_správce>.
37. SCHWARTZ, B. et al. *MySQL profesionálně – optimalizace pro vysoký výkon*. 1. vyd. Brno : Zoner Press, 2009. 712 s. ISBN 978-80-7413-035-9.
38. STEPHENS, B., PLEW, R., JONES, A. D. *Naučte se SQL za 28 dní*. 1. vyd. Brno : Computer Press, 2010. 728 s. ISBN 978-80-251-2700-1.

39. *Superhrubá mzda základem daně – Finance.cz*. [online]. c2011 [cit. 2011-03-14]. Dostupné z URL: <http://www.finance.cz/dane-a-mzda/informace/reforma-2008/superhruba-mzda/>.
40. *Typy úloh – MoodleDocs* [online]. c2008 [cit. 2011-03-11]. Dostupné z URL: <http://docs.moodle.org/cs/Typy_%C3%BAloh>.
41. VACH, D. *Třívrstvá architektura | ITexpert.cz* [online]. c2006 [cit. 2011-05-07]. Dostupné z URL: <http://www.itexpert.cz/trivrstva-architektura/>.
42. VÁCLAVOVIČ, J. *Skriptovací programovací jazyky* [online]. c2001 [cit. 2010-03-15]. Dostupné z URL: <http://reboot.cz/howto/programovani/skriptovaci-programovaci-jazyky/articles.html?id=153>.
43. VÁŇOVÁ, T. *Moodle v síti*. 1. vyd. Brno : Tribun EU, 2008. 80 s. ISBN 978-80-7399-447-1.
44. VÍCHA, K. *Virtuální studium – Moodle.cz* [online]. c2004 [cit. 2010-06-20]. Dostupné z URL: <http://interval.cz/clanky/virtualni-studium-moodlecz/>.
45. W3C. *HTML 4.01 Specification* [online]. c1999 [cit. 2011-03-14]. Dostupné z URL: <http://www.w3.org/TR/html4/>.
46. W3C. *Extensible Markup Language (XML)* [online]. c2003 [cit. 2011-03-14]. Dostupné z URL: <http://www.w3.org/XML/>.
47. Zákon č. 262/2006 Sb., zákoník práce. Ve znění zákona č. zákona č. 347/2010 Sb. (účinnost od 1. ledna 2011).
48. Zákon č. 586/1992 Sb., o daních z příjmů. Ve znění zákona č. 348/2010 Sb. (účinnost od 1. ledna 2011).
49. Zákon č. 99/1963 Sb., občanský soudní řád. Ve znění zákona č. zákona č. 409/2010 Sb. (účinnost od 1. ledna 2011).

Seznam obrázků a tabulek

Seznam příloh

Příloha č. 1 – db/install.xml

Příloha č. 2 – lang/cs_utf8/help/dpfo/

Příloha č. 3 – lang/cs_utf8/qtype/dpfo

Příloha č. 4 – simpletest/testquestiontype

Příloha č. 5 – display.html

Příloha č. 6 – edit_dpfo_form.php

Příloha č. 7 – questiontype.php

K tištěné verzi bakalářské práce je přiloženo CD obsahující text práce a zdrojové kódy modulu.

Přílohy

Příloha č. 1 – db/install.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<XMLDB PATH="question/type/dpfo/db" VERSION="20060825" COMMENT="XMLDB
pro question/type/dpfo">
  <TABLES>
    <TABLE NAME="question_dpfo" COMMENT="polozky tabulky pro DPFO">
      <FIELDS>

        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIG-
NED="true" SEQUENCE="true" ENUM="false" NEXT="question"/>
        <FIELD NAME="question" TYPE="int" LENGTH="10" NOTNULL="true"
UNSIGNED="true" DEFAULT="0" SEQUENCE="false" ENUM="false" PREVI-
OUS="id" NEXT="answer"/>
        <FIELD NAME="answer" TYPE="int" LENGTH="10" NOTNULL="true"
UNSIGNED="true" DEFAULT="0" SEQUENCE="false" ENUM="false" PREVI-
OUS="question" NEXT="hrmzda"/>

        <FIELD NAME="hrmzda" TYPE="int" LENGTH="6" NOTNULL="true"
UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="answer"
NEXT="prprvztah"/>
        <FIELD NAME="prprvztah" TYPE="char" LENGTH="4" NOTNULL="true"
UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="hrmzda"
NEXT="prohlaseni"/>
        <FIELD NAME="prohlaseni" TYPE="bool" LENGTH="1" NOTNULL="true"
UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="prprvztah"
NEXT="student"/>
        <FIELD NAME="student" TYPE="bool" LENGTH="1" NOTNULL="true"
UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="prohlaseni"
NEXT="deti"/>
        <FIELD NAME="deti" TYPE="int" LENGTH="2" NOTNULL="true" UNSIG-
NED="true" SEQUENCE="false" ENUM="false" PREVIOUS="student"/>


//unsigned = bez znamenka - pouze kladne hodnoty
//sequence = zachovani poradi (1,2,3)
//enum = vyctovy typ - mnozina konstant (enumeratoru) - promenna naby-
va hodnoty jedne z nich
//previous / next = funkce, ktera posouva interni ukazatel o jeden pr-
vek dopredu / dozadu  (pro prazdne prvky nebo prvky s indexem 0 vraci
"false" => zajistuje provazanost prvku - spravny pruchod polem jen pri
vyplneni vsech prvku)
      </FIELDS>
      <KEYS>
        <KEY NAME="primary" TYPE="primary" FIELDS="id"
NEXT="question"/>
        // primarni klic
        <KEY NAME="question" TYPE="foreign" FIELDS="question" REFTA-
BLE="questions" REFFIELDS="id" PREVIOUS="primary"/>
        // cizi klic - odkaz do tabulky questions
      </KEYS>
```

```
      <INDEXES>
        <INDEX NAME="answer" UNIQUE="false" FIELDS="answer"/>
        // index slouiz pro rychle hledani v tabulce
      </INDEXES>
    </TABLE>
  </TABLES>
</XMLDB>
```

Příloha č. 2 – lang/cs_utf8/help/dpfo/

```
<p align="center"><b>DPFO</b></p>

<p>Tento typ ulohy umoznuje testovat vypocty dani z prijmu (mezd) fy-
zickych osob. Zadejte pozadovane udaje (hruba mzda, typ pracovne-
pravniho vztahu, informace o Prohlaseni poplatnika dane, studiu a po-
ctu deti poplatnika) do prislusnych kolonek. Bude proveden automaticky
vypocet dane podle aktualniho zakona o dani z prijmu (k datu 26. 3.
2011), ktery bude porovnan s vysledkem ziskanym do studenta.</p>
```

Příloha č. 3 – lang/cs_utf8/qtype/dpfo

```
<?php // $Id: qtype_TEMPLATE.php,v 1.3 2009/02/26 14:39:44 mjollnir_
Exp $
/**
 * The language strings for the DPFO question type.
 *
 * @copyright &copy; 2011 Monika Brejchová
 * @author monika.brejchova@gmail.com
 * @license http://www.gnu.org/copyleft/gpl.html GNU Public License
 * @package DPFO
 *//** */

$string['addingdpfo'] = 'Pridani ulohy pro vypocet dane z prijmu.';
$string['editingdpfo'] = 'Uprava ulohy pro vypocet dane z prijmu.';
$string['dpfo'] = 'DPFO';
$string['dpfosummary'] = 'Typ ulohy pro vypocet dane z prijmu fyzic-
kych osob.';
$string['hrmzda'] = 'Hruba mzda';
$string['prprvztah'] = 'Pracovne-pravni vztah';
$string['dopp'] = 'Dohoda o provedeni prace';
$string['dopc'] = 'Dohoda o pracovni cinnosti';
$string['pp'] = 'Pracovni pomer';
$string['prohlaseni'] = 'Prohlaseni poplatnika dane';
$string['student'] = 'Student';
$string['deti'] = 'Pocet deti';
$string['odpoved'] = 'Odpoved';
$string['vetsineznula'] = 'Zadane cislo musi byt vetsi nez nula.';
$string['celecislo'] = 'Zadana hodnota musi byt cele cislo.';
$string['asponnula'] = 'Zadane cislo musi byt vetsi nebo rovno nule.';

?>
```

Příloha č. 4 – simpletest/testquestiontype

```
<?php
/**
 * Unit tests for this question type.
 *
 * @copyright &copy; 2011 Monika Brejchová
 * @author monika.brejchova@gmail.com
 * @license http://www.gnu.org/copyleft/gpl.html GNU Public License
 * @package DPFO
 *//** */

require_once(dirname(__FILE__) . '/../../../../config.php');

global $CFG;
require_once($CFG->libdir . '/simpletestlib.php');
require_once($CFG->dirroot . '/question/type/dpfo/questiontype.php');

class dpfo_qtype_test extends UnitTestCase {
    var $qtype;

    function setUp() {
        $this->qtype = new dpfo_qtype();
    }

    function tearDown() {
        $this->qtype = null;
    }

    function test_name() {
        $this->assertEqual($this->qtype->name(), 'dpfo');
    }

    // TODO write unit tests for the other methods of the question ty-
pe class.
}

?>
```

Příloha č. 5 – display.html

```
<div class="qtext">
    <?php echo $questiontext; ?>
</div>

<div class="hrmzda">
      <?php
          echo get_string('hrmzda', 'qtype_dpfo').': ';
          echo $zadanihrmzda;
        ?>

    </div>

    <div class="prprvztah">
      <?php
```

```
          echo get_string('prprvztah', 'qtype_dpfo').': ';
          echo $zadaniprprvztah;
        ?>

    </div>

    <div class="prohlaseni">
      <?php
          echo get_string('prohlaseni', 'qtype_dpfo').': ';
          echo $zadaniprohlaseni;
        ?>

    </div>

    <div class="student">
      <?php
          echo get_string('student', 'qtype_dpfo').': ';
          echo $zadanistudent;
        ?>

    </div>

    <div class="deti">
      <?php
           echo get_string('deti', 'qtype_dpfo').': ';
          echo $zadanideti;
        ?>

    </div>

<?php if ($image) { ?>
    <img class="qimage" src="<?php echo $image; ?>" alt="" />
<?php } ?>

<div class="ablock clearfix">
    <!-- TODO add the form controls that the student will use to enter
their response. -->

    <?php if ($feedback) { ?>
        <div class="feedback">
            <?php echo $feedback ?>
        </div>
    <?php } ?>



   <div class="odpoved">

      <?php
        echo get_string('answer', 'quiz').': ';
      ?>

      //pole pro vepsání odpovìdi
      <input type="text" class=""   name=""   value=""  size="10"/>

    </div>


    <!-- tlacitko "Odeslat" -->
```

```
    <?php
         $this->print_question_submit_buttons($question, $state, $cmo-
ptions, $options);
    ?>



</div>
```

Příloha č. 6 – edit_dpfo_form.php

```
<?php
/**
 * The editing form code for this question type.
 *
 * @copyright &copy; 2011 Monika Brejchová
 * @author monika.brejchova@gmail.com
 * @license http://www.gnu.org/copyleft/gpl.html GNU Public License
 * @package DPFO
 *//** */

require_once($CFG->dirroot.'/question/type/edit_question_form.php');

/**
 * DPFO editing form definition.
 *
 * See http://docs.moodle.org/en/Development:lib/formslib.php for in-
formation
 * about the Moodle forms library, which is based on the HTML Quick-
form PEAR library.
 */
class question_edit_dpfo_form extends question_edit_form {


      function definition_inner(&$mform) {
        // definice dalsich formularovych poli

    // hruba mzda
        $mform->addElement('text', 'hrmzda', get_string('hrmzda',
'qtype_dpfo'),
                array('size' => 6));
        $mform->setType('hrmzda', PARAM_NUMBER);
        $mform->addRule('hrmzda', null, 'required', null, 'client');

    // pracovne-pravni vztah
        $menu = array(get_string('dopp', 'qtype_dpfo'),
get_string('dopc', 'qtype_dpfo'), get_string('pp', 'qtype_dpfo'));
        $mform->addElement('select', 'prprvztah',
get_string('prprvztah', 'qtype_dpfo'), $menu);

    // Prohlaseni poplatnika dane
        $mform->addElement('select', 'prohlaseni',
get_string('prohlaseni', 'qtype_dpfo'),
        array(0 => get_string('false', 'quiz'), 1 =>
get_string('true', 'quiz')));
```

```
    // student
        $mform->addElement('select', 'student', get_string('student',
'qtype_dpfo'),
        array(0 => get_string('false', 'quiz'), 1 =>
get_string('true', 'quiz')));

    // pocet deti
        $mform->addElement('text', 'deti', get_string('deti', 'qty-
pe_dpfo'), array('size' => 2));
        $mform->setType('deti', PARAM_INT);
        $mform->addRule('deti', null, 'required', null, 'client');

    // spravna odpoved
        $mform->addElement('text', 'odpoved', get_string('odpoved',
'qtype_dpfo'),
                array('size' => 45));
        $mform->setType('odpoved', PARAM_NUMBER);
        $mform->setDefault('odpoved', 'Spravna odpoved bude vypocitana
automaticky.');


    }


    function set_data($question) {
    // nacteni dat do formulare

        if (!empty($question->options)) {
        $answers = $question->options->answers;
        foreach ($answers as $answer){
                    $question->hrmzda = $answer->hrmzda;
                    $question->prprvztah = $answer->prprvztah;
                    $question->prohlaseni = $answer->prohlaseni;
                    $question->student = $answer->student;
                    $question->deti = $answer->deti;
                    $question->odpoved = $answer->answer;
                }
        }

        parent::set_data($question);
    }



    function validation($data) {
        // osetreni vstupnich hodnot
        $errors = array();

        //hruba mzda musi byt vetsi nez nula
         $hrmzda = $data['hrmzda'];
         if ($hrmzda <= 0) {
             $errors['hrmzda'] =
get_string('vetsineznula','qtype_dpfo');
        }

        //hruba mzda musi byt cele cislo
        if (!(is_int($hrmzda))) {
             $errors['hrmzda'] = get_string('celecislo','qtype_dpfo');
        }
```

```
        //pocet deti musi byt vetsi nebo roven nule
        $deti = $data['deti'];
         if ($deti < 0) {
             $errors['deti'] = get_string('asponnula','qtype_dpfo');
        }

        //pocet deti musi byt cele cislo
        if (!(is_int($deti))) {
             $errors['deti'] = get_string('celecislo','qtype_dpfo');
        }

        if ($errors) {
            return $errors;
        } else {
            return true;
        }
    }

    function qtype() {
        return 'dpfo';
    }


}
?>
```

Příloha č. 7 – questiontype.php

```
<?php
/**
 * The question type class for the DPFO question type.
 *
 * @copyright &copy; 2011 Monika Brejchová
 * @author monika.brejchova@gmail.com
 * @license http://www.gnu.org/copyleft/gpl.html GNU Public License
 * @package dpfo
 *//** */

/**
 * The DPFO question class
 *
 * Obsahuje funkce pro praci s daty a vypocet odpovedi.
 */
class dpfo_qtype extends default_questiontype {

    function name() {
        return 'dpfo';
    }

/**
     * @return boolean to indicate success of failure.
     */

     function get_question_options(&$question) {
        // ziskani dat a defaultnich hodnot z databaze
```

```
         global $CFG;
         // global = globalni promenna (zajisti viditelnost promenne
uvnitr funkce)
         //$CFG - CFG (context-free grammar) = formalni gramatika, ve
ktere maji vsechny pravidla tvar "neterminal -> retezec terminalu nebo
neterminalu"

          if (!$question->options->answers = get_records_sql(
                                    "SELECT a.*, d.hrmzda, d.prprvztah,
d.prohlaseni, d.student, d.deti " .
                                    "FROM {$CFG->prefix}question_answers
a, " .
                                    "     {$CFG->prefix}question_dpfo d "
.
                                    "WHERE a.question = $question->id " .
                                    "    AND   a.id = d.answer " .
                                    "ORDER BY a.id ASC")) {
            notify('Chyba: Chybejici data pro dpfo ulohu ' . $question
->id . '!');
            return false;
         }
         }

           /**
     * Save the units and the answers associated with this question.
     * @return boolean to indicate success of failure.
     */

    /// Exchange formula variables with the correct values...
    //global $QTYPES;
    //$answer = $QTYPES['dpfo']->substitute_variables($formula, $indi-
vidualdata);

    function save_question_options($question) {

       function qtype_dpfo_calculate_answer($hrmzda, $prprvztah, $pro-
hlaseni, $student, $deti) {
       //vypocet spravne odpovedi

      $dpfo = 0;
      if ($prprvztah == 'dpp' && $hrmzda <= 5000 && $prohlaseni ==
false)
      //vypocet srazkove dane
      {dpfo == 0.15 * $hrmzda;}

      else
        //vypocet zalohove dane
      {
      //vypocet superhrube mzdy
      $supermzda = $hrmzda + (0.25 * $hrmzda) + (0.09 * $hrmzda);
      //vypocet zalohy na dani
      $zaloha = 0.15 * $supermzda;
      //vypocet slevy na dani
      $sleva = 0;
      if ($prohlaseni == true)
      {$sleva = 1970;}
      if ($student == true)
      {$sleva = $sleva + 335;}
      if ($sleva > $zaloha)
```

```
        {$sleva = $zaloha;}
        //vypocet danoveho bonusu
        if ($deti > 0)
        {$bonus = $deti * 967;}
        if ($bonus > 4835)
        {$bonus = 4835;}
         //vypocet dane
        $dpfo = $zaloha - $sleva - $bonus;
        }


        // vysledek, ktery vraci funkce
        return $dpfo;
    }

          // ulozeni dat z objektu $question do databaze
          //($question has all the post data from editquestion.html)

          // nacteni jiz ulozenych dat

          if (!$oldanswers = get_records('question_answers', 'question',
$question->id, 'id ASC')) {
              $oldanswers = array();
          }

          if (!$oldoptions = get_records('question_dpfo', 'question',
$question->id, 'answer ASC')) {
              $oldoptions = array();
          }

          // ulozeni odpovedi do mdl_question_answers
            $answer = new stdClass;
            $vysledek = 0;
            $vysledek = qtype_dpfo_calculate_answer($question->hrmzda,
$question->prprvztah, $question->prohlaseni, $question->student, $que-
stion->deti);
                $answer->answer   = $vysledek;
                $answer->question = $question->id;

           if ($oldanswer = array_shift($oldanswers)) {  // aktualizace
odpovedi
                  $answer->id = $oldanswer->id;
                  if (! update_record("question_answers", $answer)) {
                      $result->error = "Nelze aktualizovat odpoved!
(id=$answer->id)";
                      return $result;
                  }
              } else { // vlozeni odpovedi
                  if (! $answer->id = insert_record("question_answers",
$answer)) {
                      $result->error = "Nelze ulozit odpoved!";
                      return $result;
                  }
              }

           // ulozeni hodnot do mdl_question_dpfo
           if (!$options = array_shift($oldoptions)) {
                  $options = new stdClass;
              }
```

```
                    $options->question = $question->id;
                    $options->answer = $answer->id;
                    $options->hrmzda = $question->hrmzda;
                    $options->prprvztah = $question->prprvztah;
                    $options->prohlaseni = $question->prohlaseni;
                    $options->student = $question->student;
                    $options->deti = $question->deti;


            if (isset($options->id)) { // aktualizace
                if (! update_record('question_dpfo', $options)) {
                    $result->error = "Could not update dpfo options!
(id=$options->id)";
                    return $result;
                }
            }
            else { // vlozeni
                if (! insert_record('question_dpfo', $options)) {
                    $result->error = "Could not insert quiz numerical
options!";
                   return $result;
                }


          }


            }


    /**
     * Deletes question from the question-type specific tables
     *
     * @param integer $questionid The question being deleted
     * @return boolean to indicate success of failure.
     */
    function delete_question($questionid) {
        // TODO delete any
        return true;
    }

    function create_session_and_responses(&$question, &$state, $cmop-
tions, $attempt) {
        // TODO create a blank repsonse in the $state->responses
array, which
        // represents the situation before the student has made a re-
sponse.

        return true;
    }

    function restore_session_and_responses(&$question, &$state) {
        // TODO unpack $state->responses[''], which has just been lo-
aded from the
        // database field question_states.answer into the $state-
>responses array.
        return true;
    }
```

```
    function save_session_and_responses(&$question, &$state) {
        // TODO package up the students response from the $state-
>responses
        // array into a string and save it in the questi-
on_states.answer field.

        $responses = '';

        return set_field('question_states', 'answer', $responses,
'id', $state->id);
    }

    function print_question_formulation_and_controls(&$question,
&$state, $cmoptions, $options) {
        global $CFG;

        $readonly = empty($options->readonly) ? '' : 'disa-
bled="disabled"';

        // Print formulation
        $questiontext = $this->format_text($question->questiontext,
                $question->questiontextformat, $cmoptions);
        $image = get_question_image($question, $cmoptions->course);

        // TODO prepare any other data necessary. For instance
        $feedback = '';
        if ($options->feedback) {

        }

       //nacteni zadanych hodnot
       if (!$moznosti = get_records('question_dpfo', 'question', $que-
stion->id, 'answer ASC')) {
            $moznosti = array();
        }
         if (!$zadani = array_shift($moznosti)) {
                $zadani = new stdClass;
           }

          $zadanihrmzda = $zadani->hrmzda;
          $zadaniprprvztah = $zadani->prprvztah;
            if ($zadaniprprvztah = '0') {
                $zadaniprprvztah = get_string('dopp', 'qtype_dpfo');
            }
             if ($zadaniprprvztah = '1') {
                $zadaniprprvztah = get_string('dopc', 'qtype_dpfo');
            }
             if ($zadaniprprvztah = '2') {
                $zadaniprprvztah = get_string('pp', 'qtype_dpfo');
            }
          $zadaniprohlaseni = $zadani->prohlaseni;
            if ($zadaniprohlaseni = '0') {
                $zadaniprohlaseni = 'ne';
            }
            if ($zadaniprohlaseni = '1') {
                $zadaniprohlaseni = 'ano';
            }
          $zadanistudent = $zadani->student;
```

```
            if ($zadanistudent = '0') {
                $zadanistudent = 'ne';
            }
            if ($zadanistudent = '1') {
                $zadanistudent = 'ano';
            }
          $zadanideti = $zadani->deti;



        include("$CFG->dirroot/question/type/dpfo/display.html");
    }

    function grade_responses(&$question, &$state, $cmoptions) {
        // TODO assign a grade to the response in state.
    }

    function compare_responses($question, $state, $teststate) {
        // TODO write the code to return two different student respon-
ses, and
        // return two if the should be considered the same.
        return false;
    }

    /**
     * Checks whether a response matches a given answer, taking the
tolerance
     * and units into account. Returns a true for if a response mat-
ches the
     * answer, false if it doesn't.
     */
    function test_response(&$question, &$state, $answer) {
        // TODO if your code uses the question_answer table, write a
method to
        // determine whether the student's response in $state matches
the
        // answer in $answer.
        return false;
    }

    function check_response(&$question, &$state){
        // TODO
        return false;
    }

    function get_correct_responses(&$question, &$state) {
        // TODO
        return false;
    }

    function get_all_responses(&$question, &$state) {
        $result = new stdClass;
        // TODO
        return $result;
    }

    function get_actual_response($question, $state) {
        // TODO
        $responses = '';
```

```
        return $responses;
    }

    /**
     * Backup the data in the question
     *
     * This is used in question/backuplib.php
     */
    function backup($bf,$preferences,$question,$level=6) {
        $status = true;

        // TODO write code to backup an instance of your question ty-
pe.

        return $status;
    }

    /**
     * Restores the data in the question
     *
     * This is used in question/restorelib.php
     */
    function restore($old_question_id,$new_question_id,$info,$restore)
{
        $status = true;

        // TODO write code to restore an instance of your question ty-
pe.

        return $status;
    }




}

// iniciace typu ulohy v systemu
question_register_questiontype(new dpfo_qtype());


?>
```