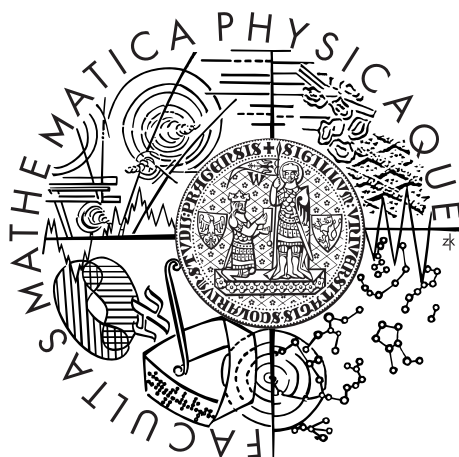


Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Petr Hudeček

Rozšíření systému XML Check

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. Irena Holubová, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2015

Děkuji vedoucí mé bakalářské práce, RNDr. Ireně Holubové, Ph.D., za čas a pozornost, kterou věnovala této práci, a za nadšení, s jakým přednáší v předmětu *Technologie XML*. Děkuji také rodičům za několikanásobnou pravopisnou, gramatickou a stylistickou kontrolu.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne

Petr Hudeček

Název práce: Rozšíření systému XML Check

Autor: Petr Hudeček

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. Irena Holubová, Ph.D., Katedra softwarového inženýrství

Abstrakt: V této práci rozšiřujeme systém pro správu domácích úkolů z roku 2011 nazvaný XML Check. Jedná se o webovou aplikaci, která se používá v předmětu Technologie XML na MFF UK a FEL ČVUT. Studenti a pedagogové od doby zavedení v systému identifikovali množství chyb, které opravujeme, a navrhli také nové funkce, které do systému přidáváme. Nejdůležitější novou funkcí je kontrola plagiátorství, která se nyní spouští na každém odevzdaném úkolu. Pro kontrolu plagiátorství jsme vyzkoušeli několik různých algoritmů (Levenshteinovu vzdálenost, algoritmus Greedy-String-Tiling a editační vzdálenost Zhang-Shasha), které v práci srovnáváme. Jejich účinnost určujeme na základě pokusů na úkolech odevzdaných za poslední čtyři roky. Rozšířený systém se nyní aktivně používá.

Klíčová slova: správa domácích úkolů, kontrola správnosti domácích úkolů, kontrola plagiátorství, správa studijních skupin

Title: Extension of the XML Check system

Author: Petr Hudeček

Department: Department of Software Engineering

Supervisor: doc. RNDr. Irena Holubová, Ph.D., Department of Software Engineering

Abstract: In this thesis, we improve the 2011 homework manager called XML Check. It is a web application used in the course XML Technologies at MFF UK and FEL ČVUT. Students and teachers have identified a number of deficiencies in the system that we are fixing. They have also suggested new functionality that we are implementing. Checking submissions for plagiarism is the most important new feature. This check is now applied to each new submission. We tried several different algorithms for this (Levenshtein distance, Greedy-String-Tiling algorithm and Zhang-Shasha tree edit distance) and we compare these algorithms. We determine their performance based on experiments on student submissions from the past four years. The improved system is now in active use.

Keywords: management of homework, checking homework correctness, checking for plagiarism, management of study groups

Obsah

Úvod	1
1 Základní informace	3
1.1 Používané pojmy	3
1.2 Předmět Technologie XML	6
1.3 Domácí úkoly v předmětu Technologie XML	8
2 Analýza	10
2.1 Systémy pro odevzdávání domácích úkolů	10
2.2 Programy pro detekci plagiátů	22
2.3 Analýza metod pro určování podobnosti	24
3 Popis provedené práce	30
3.1 Oprava chyb	30
3.2 Přidání nové funkcionality	36
3.3 Kontrola podobnosti domácích úkolů	50
4 Časová osa projektu	57
5 Uživatelská dokumentace	59
5.1 Běžné úkony	59
5.2 Úkony studenta	67
5.3 Úkony cvičícího	71
5.4 Úkony přednášejícího	75
5.5 Úkony administrátora	77
5.6 Řešení problémů	83
5.7 Instalace	84
6 Programátorská dokumentace	88
6.1 Základní dokumentace	88
6.2 Adresářová struktura projektu	88
6.3 Přístup k databázi	90

6.4	Komunikace Javascriptu s PHP	90
6.5	Soothsilver DTD Parser	91
6.6	Odstraňování objektů z databáze	93
6.7	Vícejazyčnost	94
6.8	Hlášení chyb	95
6.9	Modul <i>similarity</i> (kontrola podobnosti)	96
6.10	Dokumentace některých dalších změn	104
7	Pohled zpět a do budoucna	105
7.1	Zpětný pohled	105
7.2	Možné pokračování	106
	Závěr	110
	Seznam použité literatury	111
	Seznam tabulek	114
	Seznam obrázků	115
	Seznam použitých zkratk	116
	Přílohy	118
A	Obsah příloženého CD	118
B	Podrobné zadání domácích úkolů	118
C	Seznam opravených chyb	121
D	Seznam přidanych nebo změněných funkcí	123
E	Seznam refaktorizací	125
F	Popis technologií, které se zkouší v předmětu Technologie XML	127

Úvod

Na vysokých školách se používá mnoho různých systémů pro odevzdávání domácích úkolů. Tyto systémy studentům umožňují, obvykle pomocí webového rozhraní, posílat učitelům řešení domácích úkolů a učitelům usnadňují práci s organizací odevzdaných řešení a jejich hodnocením. Některé z těchto systémů dokonce řešení hodnotí samy, takže učitel nemusí svůj čas věnovat hodnocení vůbec. Hodnocení mohou usnadňovat také tím, že je automaticky ohodnocena alespoň část řešení.

Mezi takové systémy patří i XML Check, který byl vytvořen Janem Konopáskem v roce 2011 [1]. Tento systém byl vytvořen pro účely předmětu *Technologie XML* na Matematicko-fyzikální fakultě Univerzity Karlovy (MFF)¹ a na Fakultě elektrotechnické Českého vysokého učení technického (FEL)². Systém hodnotí řešení studentů jen částečně: zkontroluje vybrané rysy jako např. přítomnost povinných prvků nebo dostatečnou složitost, samotný počet bodů ovšem musí přidělit až cvičící. Student tak nemůže automatické hodnocení obejít trikem, a navíc může dostat zpětnou vazbu od cvičícího.

XML Check se používá nepřetržitě od roku 2011, od té doby však nebyl upraven a byly v něm nalezeny chyby. Tyto chyby v určitých případech způsobovaly, že studentovo řešení nebylo přijato, ač bylo správné, a naopak. Několik dalších chyb mělo za následek, že studenti museli někdy posílat svoje řešení cvičícímu e-mailem, protože systém je ani nedokázal vložit do databáze.

V této práci se věnujeme opravě těchto chyb, současně přitom ale také systém rozšiřujeme. Studenti a cvičící navrhli novou funkcionalitu, kterou by v systému rádi viděli, a těmito návrhy se zabýváme. Největší přidanou funkcí je pak kontrola podobnosti domácích úkolů, která má za cíl odhalit opisování mezi studenty. Porovnáváme několik metod na zjišťování podobnosti mezi soubory a tři takové metody implementujeme.

Do roku 2014 studenti odevzdali přes 2500 řešení, na kterých tyto metody testujeme. Uvedeme statistická data zjištěná z této analýzy a pokusíme se určit

¹<http://www.mff.cuni.cz/>

²<http://www.fel.cvut.cz/>

nejčastější způsoby opisování.

Vylepšený systém XML Check s nově zabudovanou kontrolou podobnosti bude v předmětu *Technologie XML* nadále využíván.

V první kapitole této práce definujeme používané pojmy, uvádíme stručnou historii předmětu *Technologie XML* a popisujeme, jak vypadají zadání domácích úkolů.

V kapitole 2 srovnáváme systém XML Check s jinými systémy pro odevzdávání domácích úkolů a pro kontrolu plagiátorství a také popisujeme vybrané metody pro porovnávání textů a XML dokumentů.

V kapitole 3 popisujeme nejzávažnější opravené chyby a nejdůležitější nové funkce a uvádíme výsledky analýzy odevzdaných řešení z hlediska podobnosti.

V kapitole 4 jen pro přehlednost uvádíme časovou osu projektu.

Kapitola 5 obsahuje uživatelskou dokumentaci, která se nachází také v odděleném souboru, ke kterému mají přístup studenti.

Kapitola 6 obsahuje programátorskou dokumentaci, která se věnuje hlavně částem systému, které byly přidány nebo změněny. Pro plné pochopení kódu musí čtenář přečíst také programátorskou dokumentaci k původní verzi systému v [1].

V poslední kapitole se pokoušíme o zpětné hodnocení a navrhuje, jak by mohl být systém dále rozšířen v budoucnu.

1. Základní informace

V této kapitole nejprve definujeme nejdůležitější používané pojmy, např. co myslíme systémem pro správu domácích úkolů, a definujeme také, co myslíme např. slovy *úkol*, *problém* nebo *řešení* v rámci systému XML Check.

Poté popisujeme krátkou historii předmětu Technologie XML a jeho současnou podobu včetně stručného popisu všech domácích úkolů, které se v předmětu zadávají.

1.1 Používané pojmy

V textu této práce jsou použity především následující pojmy¹:

1.1.1 Nejdůležitější pojmy

- **Systém pro správu (odevzdávání) domácích úkolů** je aplikace, která zjednodušuje studentovi a vyučujícímu vzájemnou komunikaci týkající se řešení a odevzdávání domácích úkolů; typicky takový systém umožňuje studentovi přihlásit se do systému a vložit do něj soubory či text řešení, vyučující pak má přístup k těmto souborům, může je v systému ohodnotit a studentům se pak zobrazuje jejich známka (popř. počet bodů) a možná také slovní komentář. Některé systémy také mají poloautomatickou nebo úplně automatickou kontrolu úkolů, tj. zkontrolují správnost studentova řešení bez toho, aby se na něj podíval vyučující.

1.1.2 Organizace

- **Konsorcium W3²** (anglicky *World Wide Web Consortium*, zkracováno **W3C**) je mezinárodní organizace, která vydává standardy pro protokoly a jazyky používané ve spojitosti s internetem. Konsorcium W3 například vydalo specifikaci jazyka XML, jazyka XPath, jazyka XQuery, technologie XSLT, technologie XSD a modelu DOM. Všechny tyto technologie a jazyky

¹Další zkratky jsou rozvedeny na straně 116.

²<http://www.w3.org/>

se používají v domácích úkolech, které se odevzdávají do systému XML Check.

- **Matematicko-fyzikální fakulta Univerzity Karlovy v Praze** (zkráceně **MFF**) je fakulta, v níž se ve studijním programu Informatika vyučuje předmět Technologie XML.
- **Fakulta elektrotechnická Českého vysokého učení technického v Praze** (zkráceně **FEL**) je fakulta, v níž se v oboru Softwarové inženýrství vyučuje předmět Technologie XML.

1.1.3 Technologie

- **Extensible Markup Language** (zkracováno **XML**) je značkovací jazyk, jehož specifikaci udržuje konsorcium W3. Používá se v různých prostředích pro různé účely, například pro spravování konfigurace nebo posílání zpráv. Jazyk popisujeme podrobněji v příloze F.1.
- **Soubor XML** je textový soubor, který je správně zformovaný podle specifikace W3C. Soubor s koncovkou *xml*, který by nebyl správně zformovaný, by tedy ani nebyl soubor XML. Pro účely této práce ovšem budeme takový soubor za soubor XML považovat, byť za soubor, který není správně zformovaný, a tedy bude systémem odmítnut. Můžeme tak mluvit o tom, že student odevzdá soubor XML, ač může odevzdat i soubor chybný (v takovém případě se ohlásí chyba). Synonymum pro soubor XML je **dokument XML** (někdy také nazýván *XML dokument*).
- **PHP: Hypertext Preprocessor**³ (zkracováno **PHP**) je programovací jazyk pro webové stránky, v němž je naprogramován systém XML Check.

1.1.4 Systém XML Check

- **Systém XML Check** (zkracováno **XML Check**, **systém** nebo **aplikace**, pokud je z kontextu zřejmé, o co se jedná) je webová aplikace pro odevzdávání domácích úkolů využívaná v předmětu Technologie XML na MFF a FEL.

³<http://www.php.net/>

Tématem této práce je její rozšiřování. Dříve se jmenovala **Assignment manager**.

- **Původní verze systému XML Check** je verze 0.68 systému XML Check, tedy verze, v jaké byl systém odevzdán jako bakalářská práce Janem Konopáskem a zároveň verze, v jaké byl využíván až do začátku letního semestru 2014/2015.
- **Přednáška** (lecture) v systému XML Check je objekt vlastněný uživatelem (přednášejícím), který sestává z několika cvičení, otázek, testů, příloh a problémů. Přednáška reprezentuje školní předmět.
- **Skupina** (group) v systému XML Check je objekt vlastněný uživatelem (cvičícím), do kterého se mohou přihlašovat studenti. Skupina reprezentuje okruh lidí navštěvujících cvičení daného předmětu vždy ve stejný termín (rozvrhový lístek).
- **Problém** (problem) v systému XML Check je objekt spojený s nějakou přednáškou, který popisuje, co student musí učinit, aby daný problém vyřešil. Problém je často asociovaný s nějakým kontrolujícím pluginem.
- **Úkol** (assignment) v systému XML Check je objekt spojený s nějakou skupinou a problémem. Úkol zadává cvičící tak, že vybere *problém*, který má jeho skupina řešit a určí pro něj termín odevzdání. Problémy typicky přežívají z roku na rok, zatímco pro každý semestr se úkoly vytváří znovu.
- **Řešení** (submission) v systému XML Check je ZIP archiv nahraný studentem k nějakému úkolu, který podle studenta daný úkol řeší.
- **Kontrolující plugin** (také **hodnotící plugin** nebo **opravující plugin**) je část systému XML Check, která je zodpovědná za automatickou kontrolu jednoho z domácích úkolů. Plugin má studentovo řešení na vstupu a jako výstup má procentuální úspěšnost a popř. chybové hlášky.

1.2 Předmět Technologie XML

Tato sekce popisuje historii předmětu Technologie XML na MFF a FEL a také jeho podobu v době sepisování této práce (tj. v akademickém roce 2014/2015).

Aplikace XML Check vznikla v roce 2011 pro použití v předmětu Technologie XML. Tento jednosemestrální předmět se vyučuje jak na MFF (pod kódem NPRG036), tak i na FEL (pod kódem A7B36XML).

Na MFF byl předmět poprvé vyučován v roce 2005, od roku 2008 ho pak samostatně vyučuje doc. RNDr. Irena Holubová, Ph.D., která předmět vyučuje i na FEL.

Na obou vysokých školách se předmět vyučuje velmi podobně. Jeho týdenní dotace je 2+2, tedy každý týden se koná přednáška trvající devadesát minut a jedno cvičení trvající devadesát minut. Předmět od studentů vyžaduje (neklasifikovaný) zápočet a složení zkoušky.

Během přednášky přednášející vysvětluje studentům principy vybraných technologií souvisejících s formátem XML. Přednášející přitom také učí syntaxi a sémantiku vybraných souvisejících programovacích jazyků.

Během cvičení, která se na obou školách odehrávají v počítačové laboratoři, cvičící v krátkosti ukáže další praktické informace týkající se právě probírané technologie a poté dá studentům vypracovat sérii krátkých příkladů na její použití. Pokud student vyřeší takový případ v časovém limitu, odešle ho e-mailem cvičícímu, který mu udělí jeden bod.

Na konci každého druhého cvičení cvičící zadá studentům větší domácí úkol, celkem takto studenti vypracují šest domácích úkolů během semestru. Těchto šest domácích úkolů má v daném roce stejné zadání napříč cvičeními, a dokonce napříč školami. Navíc se zadání úkolů nijak zásadně nezměnilo od doby, kdy byly úkoly zadány poprvé. V zadání se pouze opravovaly chyby nebo zpřesňovaly formulace.

Za každý domácí úkol může student získat až 20 bodů při perfektním vypracování. Celkem tedy může získat během semestru 120 bodů za domácí úkoly a přibližně 50 bodů za vypracovávání příkladů během cvičení. Student musí získat alespoň 100 bodů, aby mu byl přiznán zápočet. Body přesahující hranici 100 bodů se studentovi přičtou k jeho výsledku ze zkoušky.

Zkouška na konci semestru má podobu písemného testu, který kontroluje

znalosti studenta ze všech probíraných technologií. Za perfektní odpovědi na všechny otázky v písemném testu získá student 100 bodů.

Známky jsou na MFF přidělovány studentům podle počtů získaných bodů:

- Kdo má alespoň 90 bodů, dostane jedničku.
- Kdo má alespoň 75 bodů, dostane dvojku.
- Kdo má alespoň 60 bodů, dostane trojku.
- Kdo má méně, u zkoušky neuspěl.

Před vznikem původní verze systému XML Check v roce 2011 studenti posílali řešení domácích úkolů cvičícímu na e-mail. Ten řešení ohodnotil a body vepsal do tabulky, kterou na svých stránkách udržuje přednášející. Po zavedení systému XML Check cvičící místo toho stahuje řešení studentů ze systému a po ohodnocení vepíše body zpět do systému. Systém udržuje tabulku studentů s přiznanými body. Cvičící nicméně stále přepisují body za domácí úkoly do zvláštní tabulky na webových stránkách předmětu, protože tabulka obsahuje více informací než jen body za domácí úkoly, konkrétně obsahuje také body přenesené z minulého ročníku, body za příklady ze cvičení, body ze zkoušky a informace o prodělaných pokusech o zkoušku.

Anotace předmětu na MFF⁴ zní:

„Cílem přednášky je seznámit posluchače se základními principy, formáty a nástroji založenými na technologii XML. Probereme klíčové aspekty od principů formátu samotného, přes popis přípustné struktury XML dat, rozhraní pro práci s XML dokumenty, až po jazyky pro dotazování, aktualizace a transformace XML dat. Na závěr se seznámíme [s] problematikou dobrého návrhu XML dat, testováním XML aplikací a s nejběžnějšími XML formáty. Hlavní důraz přednášky bude kladen na praktickou stránku problematiky.“

Anotace předmětu na FEL⁵ zní:

⁴Sylabus na MFF: <https://is.cuni.cz/studium/predmety/index.php?do=predmet&kod=NPRG036>

⁵Sylabus na FEL: http://www.fel.cvut.cz/cz/education/bk_peo/predmety/13/95/p1395506.html

„Přehled základních principů, formátů a nástrojů založených na technologii XML. Formát XML, definice struktury pomocí schématu zapsaného v jazyce DTD nebo XML Schema. Reprezentace XML dat a dokumentů, rozhraní [sic] DOM a SAX. Jazyk XPath, dotazovací jazyk XQuery. XML databáze a jejich vztah k jiným databázovým systémům.“

1.3 Domácí úkoly v předmětu Technologie XML

Předmět Technologie XML předkládá studentům šest domácích úkolů. Na vypracování každého z nich má student dva týdny.

Na začátku semestru má student zvolit téma, které bude modelovat pomocí souborů XML a souvisejících technologií. Může například zvolit téma „knihovna“ a soubor XML bude databází knih a výpůjček. Nebo může zvolit téma „hudební skladba“ a soubor XML bude zápisem notové osnovy. Mohl by také zvolit téma „úřad“ a v souboru XML popisovat různé kanceláře a zaměstnance.

V prvním úkolu pak vytvoří soubor XML, se kterým bude pracovat po zbytek semestru.

Podrobné zadání všech šesti domácích úkolů se nachází v příloze B. Podrobnější popis technologií, jejichž znalost zkoušejí tyto domácí úkoly, se nachází v příloze F.

1.3.1 Úkol 1: XML a DTD

Student má vytvořit soubor XML popisující vybrané téma, tento soubor má splňovat určité minimální nároky na složitost definované v zadání. Navíc má student vytvořit soubor DTD⁶, který bude popisovat strukturu vytvořeného souboru XML.

1.3.2 Úkol 2: DOM/SAX

Existuje větší množství způsobů, jak parsovat soubory XML. Mezi nejběžněji používané patří standard pro parsování Document Object Model a knihovna Simple API for XML. Pro splnění tohoto úkolu musí student napsat zdrojový kód dvou tříd v jazyce Java, který provede dostatečně složité transformace na souboru XML

⁶Jazyk **Document Type Definition** popisuje strukturu souboru XML.

z prvního úkolu, a který z tohoto souboru XML zjistí nějaké informace. K tomu využijte tyto dvě zmíněné technologie.

1.3.3 Úkol 3: XPath

XPath je dotazovací jazyk pro dokumenty XML. Student má v tomto úkolu vytvořit pět dotazů XPath nad svým dokumentem z prvního úkolu. Tyto dotazy musí splňovat určité nároky na složitost.

1.3.4 Úkol 4: Schéma XSD

Dokumentům DTD chybí některé funkce užitečné pro validaci dokumentů, W3C proto vytvořilo standard XSD⁷. Schéma XML je soubor s příponou *xsd*, pomocí kterého lze validovat soubor XML. V tomto úkolu má student napsat vlastní soubor XSD, jehož validací by prošel soubor XML z prvního úkolu. Schéma XSD musí splňovat určité nároky na složitost a soubor z prvního úkolu může student upravit.

1.3.5 Úkol 5: XQuery

XQuery je pokročilý dotazovací jazyk pro dokumenty XML, který je vystavěn na základech XPath. Student má v tomto úkolu vytvořit pět dotazů XQuery nad svým dokumentem z prvního úkolu a nad alespoň jedním dalším dokumentem (dotazy XQuery mohou pracovat s více dokumenty současně). Tyto dotazy musí splňovat určité nároky na složitost.

1.3.6 Úkol 6: Transformace XSLT

XSLT⁸ je programovací jazyk pro transformaci XML dokumentů. Skripty XSLT mají podobu správně zformovaných XML dokumentů. Student má v tomto úkolu vytvořit jeden XSLT skript, který provede dostatečně složitou transformaci dokumentu XML z prvního úkolu (složitost je definována podrobně v zadání úkolu).

⁷XML Schema Definition Language

⁸Extensible Stylesheet Language Transformations

2. Analýza

2.1 Systémy pro odevzdávání domácích úkolů

Ačkoliv byl systém XML Check vyvinut pro použití v předmětu Technologie XML, je velmi modulární a dal by se používat i v jiných předmětech a prostředích. Vývoji původní verze systému, vytvořené v roce 2011, předcházela analýza [1, str. 9-10], která ho srovnávala s jedním jiným systémem pro správu domácích úkolů, konkrétně se systémem CodEx[27]. Analýza tehdy ukázala, že oba systémy jsou si podobné a mají mnoho stejných funkcí, existoval ovšem zásadní rozdíl:

- Systém XML Check provádí určitou automatickou kontrolu řešení studenta, stále se ovšem spoléhá na cvičícího, který musí kontrolu systému potvrdit a přidělit body. Pokud systém řešení označí jako neúplné, je na cvičícím, kolik bodů studentovi přidělí.
- Naproti tomu CodEx provádí úplnou kontrolu řešení studenta. Cvičící sice má možnost přidělit studentovi *bonusové body* nebo mu naopak body strhnout (např. když bude řešení opsané), typicky tak ovšem nečiní. Celý systém je navržen tak, že přidělování bonusových bodů je spíše výjimkou než normou.

Z tohoto důvodu se autor původní verze systému rozhodl implementovat nový systém místo využití CodExu.

Systém se po následující tři roky osvědčil, objevilo se v něm ovšem několik problémů, popsaných níže v kapitole 3.1, stálo tedy za úvahu, zda by nebylo vhodné systém přestat využívat a nahradit jej některým z jiných existujících systémů pro opravu domácích úkolů. A naopak, je možné, že systém XML Check by mohl být výhodně zaveden pro použití v jiném předmětu než Technologie XML. Z těchto důvodů byla provedena následující analýza existujících systémů pro správu domácích úkolů.

2.1.1 Metodologie analýzy

Systém XML Check má za sebou tři roky používání a je již otestovaný. Není vhodné ho tedy nahradit systémem, který byl právě vyvinut a ještě nebyl nikdy

v živém provozu. Tato analýza proto studuje hlavně systémy, které se již někde používaly nebo se právě používají. Pro každý takový odevzdávací systém se ptá na tyto otázky:

- **Rozsah a rozšiřitelnost:** Jaký je rozsah systému? Jaké druhy domácích úkolů dokáže přijímat? Dá se snadno rozšířit, aby přijímal i jiné druhy domácích úkolů?
- **Uživatel:** Kdo systém nyní používá a k jakému účelu?
- **Důvod existence:** Proč si uživatelé tohoto systému vybrali právě tento systém? Pokud je uživatel systému zároveň autorem systému, nebo alespoň nařídil jeho tvorbu, proč to považoval za nutné, a nevyužil jiný systém?
- **Historie a vývoj:** Kdy systém vznikl a kdo jej vytvořil? Je systém stále v aktivním vývoji, tj. opravují se nalezené chyby a přidávají nové funkce?
- **Popis:** Jak systém funguje z uživatelského hlediska z pohledu studenta? Jak systém funguje z uživatelského hlediska z pohledu učitele, cvičícího nebo přednášejícího?
- **Technologie:** Jaké technologie systém používá?
- **Upravitelnost a licence:** Dá se systém upravit? Je zdrojový kód veřejně k dispozici, a pokud ne, jakým způsobem se dá sehnat? Jak snadné je upravit tento zdrojový kód, aby systém vyhovoval potřebám předmětu *Technologie XML*? Jakou licenci je chráněn zdrojový kód?
- **Kladné a záporné stránky:** Jaké zvláštní funkce systému jeho současnému uživateli nejvíce pomáhají? Jaké problémy má systém, a v jakých ohledech nevyhovuje současnému uživateli?
- **Záměnnost:** Je možné tímto systémem nahradit XML Check pro použití v předmětu *Technologie XML*? Zastal by systém XML Check roli tohoto systému lépe než tento systém pro jeho současného uživatele?

2.1.2 Odevzdávání e-mailem učiteli

Velmi běžné je odevzdávat domácí úkoly bez použití zvláštního systému prostě tak, že student pošle svůj domácí úkol jako přílohu e-mailu svému učiteli. Učitel projde e-maily, které dostal, ohodnotí je a zvláště si do samostatného souboru zapisuje výsledky studentů.

- **Rozsah:** Přes e-mail se dají posílat libovolné soubory, ovšem pouze do určité velikosti (některé e-mailové servery, klienti nebo firewally blokují větší e-maily). Některé e-mailové servery blokují určité přílohy, např. *gmail.com* neumožňuje posílání souborů EXE, a to ani v archivu. Samozřejmě nejsou úkoly nijak kontrolovány automaticky, celkem jednoduše je lze ovšem automaticky roztrždit např. podle předmětu e-mailu do složek podle řešeného úkolu.
- **Uživatel:** Tímto způsobem postupuje velké množství učitelů na mnoha školách v mnoha různých předmětech.
- **Důvod existence:** Všechny ostatní popisované systémy se musí nějakým způsobem instalovat nebo zavádět a často vyžadují údržbu. V některých případech mohou studentům nebo učitelům překážet. Navíc použití specializovaného systému na odevzdávání domácích úkolů vyžaduje jak po učiteli, tak po studentech, aby se tento systém naučili používat. Pokud je počet studentů dostatečně malý a učitel nepotřebuje automatickou kontrolu, nebo žádný z existujících systémů učiteli nevyhovuje, nejspíše využije tuto metodu.
- **Záměnnost:** Předtím, než se začal používat systém XML Check, se odevzdávaly úkoly v Technologiích XML tímto způsobem. Hlavně kvůli tomu, že XML Check obsahuje automatickou kontrolu některých rysů domácích úkolů, není ovšem vhodné ho opustit a k této metodě se vrátit. Na druhou stranu systém XML Check je možné použít téměř ve všech případech, kdy se dosud odevzdává e-mailem. Soubory, které by student poslal jako přílohu e-mailu, by mohl poslat jako řešení úkolu. Učitel by tak získal systém, který by spravoval studentům počty bodů a který by zvládal organizaci pravděpodobně lépe než e-mailový klient.

2.1.3 Grupík a Grupíček

Grupíček (obrázek 1) je webová aplikace, která je součástí informačního systému IS Studium¹. Každé cvičení má založeno v Grupíčku vlastní skupinu, ačkoliv většinou není využita. V aplikaci je pro každou skupinu formulář, kde na každé řádce je buď nějaká informace pro studenta od učitele (např. „počet bodů z úkolu 1“ nebo „zápočetník odevzdán (ano/ne)“ nebo „docházka“) nebo prvek na nahrání souboru, kde student může nahrát domácí úkol.

Grupíček (Studijní mezivýsledky) (verze: 214)
Výsledky studenta

479:39 **Můj Grupíček**

Skupina

Předmět: NSWI098 Principy překladačů
Rok: 2014/15 ZS
Typ výuky: Cvičení
Vyučující: RNDr. David Bednárek, Ph.D. (bednarek@ksi.mff.cuni.cz)
Konání: Út 14:00 S4 sudý
Rozvrhový lístek: 14aNSWI098x03

Výsledky

DÚ1

du1.lex

Vložený soubor: velikost: 3kB, vložení: 14.10.2014 15:58:34

Vložit soubor: No file chosen (max. 200 kB)

Finální verze:

komentář k DÚ1

DÚ2

du2.lex

Vložený soubor: velikost: 8kB, vložení: 04.11.2014 18:22:33

Vložit soubor: No file chosen (max. 200 kB)

Finální verze:

Obrázek 1: Aplikace Grupíček

Student i cvičící si poté mohou tyto nahrané soubory stáhnout. Student může svůj soubor kdykoliv smazat.

Aplikaci *Grupíček* používá student. Cvičící využívá párovou aplikaci *Grupík*.

- **Webová adresa:** <https://is.cuni.cz/studium/grupicek/> (vyžaduje přihlášení a je přístupná pouze studentům a pedagogům Univerzity Karlovy)

¹IS Studium je informační systém firmy Erudio s. r. o., který se používá na několika vysokých školách.[28]

- **Rozsah a rozšiřitelnost:** Systém přijímá libovolné soubory, jejichž velikost může univerzita omezit. Cvičící pak může stanovit ještě nižší omezení. Pomocí Grupíčku je možné dát studentovi zadání, možnost k odevzdání i výsledek. Neprovádí sám žádnou automatickou kontrolu domácích úkolů.
- **Uživatel:** Několik cvičících Univerzity Karlovy systém používá pro své domácí úkoly, konkrétně např. v předmětech *Principy překladačů*, *Pokročilé programování v C++* a *Pokročilé programování pro .NET I* v oboru Informatika. Několik dalších vysokých škol používá systém IS Studium, z nichž alespoň jedna – Vysoká škola chemicko-technologická v Praze – využívá aktivně aplikaci Grupíček.[28]
- **Důvod existence:** Aplikace Grupíček byla původně vyvinuta na odevzdávání seminárních prací. To je vidět také například v tom, že vložené seminární práce (které lze odevzdávat pouze ve formátu PDF) jsou posílány na server *odevzdej.cz* ke kontrole podobnosti. Aplikace je ovšem dostatečně rozsáhlá, aby se dala použít i na jednodušší domácí úkoly.
- **Historie a vývoj:** Firma *Erudio s. r. o.* vyvinula systém IS Studium a spolu s ním také aplikace Grupíček a Grupík. Na vývoji stále pracuje a chyby jsou opravovány. Funkčnost aplikací ovšem zůstala již několik let nezměněná.
- **Technologie:** Hlavní část systému je naprogramovaná v prostředí Borland Delphi, systém používá databázi Oracle. Webové rozhraní je vytvořeno v jazyce PHP.[29]
- **Upravitelnost a licence:** Grupík a Grupíček nelze snadno rozšiřovat. Zdrojový kód aplikací je tajný a je ve vlastnictví firmy *Erudio*. Aplikace Grupík ovšem umožňuje cvičícímu exportovat všechny domácí úkoly od studentů, cvičící na nich takto může spustit vlastní automatické testy, do aplikace může výsledky vkládat také hromadně. Takovým způsobem Grupík využívají například vyučující předmětu *Principy překladačů* na MFF.
- **Kladné stránky:** Aplikace jsou propojeny s informačním systémem univerzity, uživatelé si tedy nepotřebují vytvářet nové uživatelské účty nebo se přihlašovat do jiného systému. Propojením také získá učitel přístup k osobním

informacím studenta. Aplikace umožňuje snadné exportování studentských řešení. V aplikaci může cvičící přidělovat body i jinak než za domácí úkoly a není téměř nijak omezován. Grupík navíc umožňuje kontrolu nad svým používáním i vyšším pracovníkům fakulty – garantům.

- **Záporné stránky:** Aplikace není rozšiřitelná a neobsahuje žádnou formu automatické kontroly. Uživatelské rozhraní této aplikace také není příliš přívětivé.
- **Zaměnitelnost:** Grupíček nemůže být snadno systémem XML Check nahrazen, ani nelze XML Check nahradit Grupíčkem. Grupíček je součástí složitějšího systému, do kterého je pevně zabudován. Grupíček neumožňuje uživatelem nastavitelnou automatickou kontrolu domácího úkolu. Vzájemné posílání informací mezi studentem a cvičícím funguje v Grupíčku obecně jinak než v systému XML Check.

Na FEL ČVUT se v některých předmětech používá systém podobný Grupíčku pod názvem **UploadSystem**[38]. Přístupný je ovšem jen studentům a učitelům ČVUT. UploadSystem navíc provádí jednoduchou kontrolu podobnosti mezi domácími úkoly.

2.1.4 *Doporučené postupy v programování*

Doporučené postupy v programování jsou na MFF předmětem, jehož cílem je „seznámit studenty s praktickými postupy a pravidly, jejichž dodržování a aplikace vedou (nejenom) k psaní kvalitnějšího kódu, [a tak] motivovat studenty k osvojení a používání probíraných postupů v praxi.“[30]

Přednášející pro účely předmětu naprogramovali vlastní webovou aplikaci pro odevzdávání úkolů, hlavně proto, že mají studenti odevzdávat úkol ve dvojicích, a toto chtěli přednášející usnadnit.

- **Rozsah a rozšiřitelnost:** Systém je určen pouze pro konkrétní domácí úkoly, které se používají v tomto předmětu. Student na začátku semestru použije systém k tomu, aby našel svého partnera do dvojice (systém jim navzájem sdělí e-mailové adresy). Studenti mají samozřejmě příležitost se

Odeslat řešení

Upozornění:

- Odesíláte za sebe i za kolegu ze dvojice
- Každé další (úspěšné) odeslání nenávratně přepíše všechna předchozí (ať už bylo nahrané vámi nebo vaším kolegou ze dvojice)
- V úvahu budeme brát pouze poslední odeslané řešení (a čas jeho odevzdání)
- Kontrola velikostí všech souborů jsou nastavena na 1,5 MB
- Pokud řešení není v Javě, C, C++ nebo C# musíte být explicitně dohodnutí s některým cvičícím!!

Odeslání

Je po deadline. Počítejte s bodovou ztrátou

Jazyk implementace Jiný jazyk:

Řešení (.zip) No file chosen

Dokumentace (.zip) No file chosen

Zdůvodnění návrhu (.pdf) No file chosen

Obrázek 2: Formulář pro odevzdání úkolu v systému pro Doporučené postupy v programování

domluvit se spolužákem předem. Poté přes aplikaci mohou odevzdat první domácí úkol (refaktorizace zadaného kódu), který musí sestávat z jednoho souboru JAVA a jednoho souboru TXT podle požadavků zadání; poté mohou odevzdat druhý domácí úkol (naprogramování knihovny pro práci se soubory INI, obrázek 2), který sestává hlavně z archivu ZIP se zdrojovými texty; a nakonec i třetí domácí úkol (oponentura a jednotkové testy k řešení jiné dvojice). Do systému jsou tyto domácí úkoly pevně vloženy a nelze snadno přidávat další.

- **Uživatel:** Systém se nyní používá výlučně v předmětu *Doporučené postupy v programování*, a to každý rok.
- **Důvod existence:** Autoři vytvořili systém na míru svým požadavkům. Webová aplikace je takto velmi jasná, protože obsahuje u každého úkolu právě ty prvky a kontroly, které daný úkol potřebuje. Navíc měli autoři poměrně neobvyklý požadavek – totiž, že studenti pracují ve dvojicích – a kvůli tomu byli nuceni si vlastní aplikaci napsat.
- **Historie a vývoj:** Systém byl vytvořen na Katedře distribuovaných a spolehlivých systémů MFF a poprvé nasazen v roce 2011. Předtím vypadaly

domácí úkoly velmi podobně, jen se neodevzdávaly přes webovou aplikaci.

- **Technologie:** Systém je naprogramován v jazyce PHP.
- **Upravitelnost a licence:** Systém je interním nástrojem katedry.
- **Kladné a záporné stránky:** Jak bylo popsáno výše, systém je vytvořen na míru domácím úkolům v tomto předmětu. Každý domácí úkol se odevzdává trochu jinak a i jeho výsledky jsou prezentovány trochu jinak (viz např. obrázek 2). Například k úkolu 2 (naprogramování knihovny) se v aplikaci oběma studentům zobrazí hodnocení cvičícího i hodnocení všech studentů, kteří na jejich řešení psali oponenturu v úkolu 3, a zároveň se jim zde zobrazí jednotkové testy, které naprogramovali tito studenti. Toto vytváření na míru ovšem zabraňuje snadnému přidávání dalších druhů domácích úkolů.
- **Záměnnost:** Vzhledem k výše uvedenému nelze ani jeden systém nahradit druhým. Největší přednost tohoto systému (fakt, že každý domácí úkol se zobrazuje ve webovém rozhraní trochu jinak) nelze přenést do systému XML Check, protože ten je navržen jako rozšiřitelný a univerzální.

2.1.5 CodEx a jemu podobné systémy

CodEx (zkratka z *The Code Examiner*) je systém pro odevzdávání domácích úkolů vyvíjený od roku 2006 na MFF, kde slouží k odevzdávání a automatické kontrole úkolů z programování v mnoha různých předmětech, např. *Programování 1*, *Programování 2*, *Jazyk C# a platforma .NET*, *Pokročilé programování pro .NET 1*, *Datové struktury* nebo *Java*. Kromě toho je také používán v semináři z programování pro střední školy.

Podrobné srovnání systému CodEx se systémem XML Check lze najít v [1, str. 9-10] v angličtině. Zde uvedeme jen, že CodEx je určen ke kontrole zdrojových kódů. Celý systém je vybudován na myšlence, že student pošle text jednoho zdrojového souboru, který bude na serveru zkompilován a spuštěn na testovacích datech a studentovi se zobrazí, zda jeho zdrojový kód úkol splnil nebo nesplnil.

Naproti tomu o řešeních úkolů v Technologiích XML nelze jednoznačně prohlásit, zda jsou vyhovující nebo nikoliv, zpravidla se nejedná o zdrojové kódy a testovací

data student dodává sám na základě svého tématu. Systém CodEx tedy nelze používat v předmětu Technologie XML a obdobně XML Check nemůže zastat práci CodExu. Každý z těchto systémů lze lépe využít v jiných situacích.

CodEx je vyladěná, funkční a široce užívaná aplikace, přesto existuje velké množství systémů, které jsou CodExu velmi podobné. Všechny tyto systémy se vyznačují tím, že vždy zkompilují studentův kód, spustí ho na testovacích datech a zobrazí výsledek ve formě *úspěch* nebo *neúspěch*, popř. zobrazí chybu vzniklou při kompilaci. Tyto aplikace jsou zaměřeny na algoritmické úlohy v programování nebo na výuku programovacího jazyka a vyžadují, aby úkoly měly jednoznačné řešení, nelze je tedy použít v předmětu Technologie XML.

Jedná se například o systém **ProgTest**, využívaný především na FIT ČVUT², ale i jinde. ProgTest provádí kontrolu podobnosti mezi domácími úkoly. Dále se na ČVUT využívá systém **Odevsys** pro úkoly v jazyce Java.³

Původně pro soutěže v programování byl vyvinut open-source systém **Mooshak** [31], který se ovšem dle autorů používá i pro domácí úkoly nebo přímo během vyučovacích hodin.

Zajímavějším systémem tohoto druhu je **TestSystem Web** popsáný v [32], kde autoři vycházeli ze systému, který byl podobný CodExu, ale nevyhovoval právě z toho důvodu, že poskytoval studentům příliš málo informací. Chtěli, aby systém studenta učil, nikoli zkoušel, proto do něj přidali funkčnost, která měří komplexitu kódu studenta, a snaží se dokonce najít konkrétní chybu ve studentově kódu, která způsobuje nesprávný výsledek. Výsledek své analýzy pak dá studentovi k dispozici, aby měl co nejlepší představu o tom, proč je jeho kód nedostatečný.

Poslední aplikací, kterou bychom představili v této kategorii, je **Web-CAT** [33], open-source systém pro automatickou kontrolu domácích úkolů z programování. Podobně jako předchozí systém i Web-CAT zkoumá komplexitu, styl a komentáře vloženého zdrojového kódu. Web-CAT je navíc zaměřen na úkoly vyžadující, aby student sám napsal automatické testy pro svoje řešení. Systém pak zkontroluje nejen korektnost řešení, ale také, zda je studentův kód dostatečně pokryt jeho vlastními testy korektnosti.

²Fakulta informačních technologií Českého vysokého učení technického v Praze

³Ani systém ProgTest ani Odevsys nejsou přístupné anonymním uživatelům.

2.1.6 Moodle a jemu podobné systémy

Existuje množství softwarových platforem pro e-learning, které správu domácích úkolů a studijních skupin obsahují jako jednu ze svých funkcí. Takové platformy jsou typicky velmi rozsáhlé, jsou schopny spravovat větší množství univerzit a předmětů a mají velké množství modulů. Mezi nejznámější takové platformy patří open-source systém **Moodle** [34].

- **Rozsah a rozšiřitelnost:** Moodle je komplexní systém. Dokáže přijímat velké množství různých souborů, podporuje mnohé funkce systému XML Check. Je určitě dostatečný pro většinu předmětů vyučovaných na vysokých školách. Navíc je rozšiřitelný pomocí pluginů, kterých existuje velké množství ve veřejné databázi; další pluginy si mohou uživatelé vytvářet sami.
- **Uživatel:** Podle statistik na oficiálních stránkách je Moodle nainstalován ve 231 zemích, využíván ve více než sedmi milionech přednášek a používat ho má přes 68 milionů uživatelů. Je otázkou, odkud tyto statistiky pocházejí (je například možné, že se do počtu uživatelů započítává každý uživatel, který se kdy registroval, i kdyby to byl jen testovací uživatel na vývojovém počítači některé univerzity), není ovšem pochyby o tom, že systém je široce využíván. Moodle se v některých předmětech využívá jak na MFF, tak na FEL.
- **Historie a vývoj:** První verzi systému Moodle vydal v roce 2002 Martin Dougiamas. Systém je nyní aktivně vyvíjen komunitou a australskou společností *Moodle Pty Ltd*.
- **Popis:** Učitel si v systému vytvoří modul pro svoji přednášku. Ten může obsahovat text, obrázky a další podstránky. Učitel tak může využívat Moodle nejen k zadávání domácích úkolů, ale také k samotné výuce. Učitel může i zadávat domácí úkoly, zobrazovat si řešení studentů a hodnotit je. Ve výchozí verzi systému není naprogramovaná žádná automatická kontrola vyjma kvízu (tj. série otázek s uzavřenou odpovědí). Určitě ovšem je možné pomocí pluginů přidat funkcionalitu vyžadovanou předmětem Technologie XML. Například plugin VPL (Virtual Programming Lab [35]) demonstruje, že je možné přidat kompilaci a kontrolu zdrojového kódu v C++.

- **Technologie:** Systém je naprogramován v PHP a běží jako webová aplikace. Umí pracovat s několika různými databázovými servery.
- **Upravitelnost a licence:** Zdrojový kód systému je k dispozici na internetu a je možné ho legálně upravovat a používat upravený kód pro výuku. Je také možné rozšířit systém pomocí vlastních pluginů.
- **Kladné a záporné stránky:** V mnoha případech pro univerzity zajišťuje funkčnost jejich instance systému externí firma, protože správa systému není jednoduchá. Nedává smysl používat Moodle jen pro jeden předmět z jedné fakulty, protože jeho údržba by se nevyplatila. Vzhledem ke složitosti systému nemusí být snadné pro studenty a učitele se v pokročilých funkcích systému orientovat, a ani psát vlastní komplikované pluginy na úrovni systému XML Check nebude snadné. Na druhou stranu je Moodle velmi široce zaměřen a s vhodnými pluginy dokáže řešit celý problém správy předmětů na vysoké škole.
- **Záměnnost:** Je možné přeprogramovat pluginy systému XML Check tak, aby fungovaly jako pluginy pro Moodle, nebyla by to ovšem triviální práce. Univerzita Karlova již disponuje nainstalovanou instancí Moodle, která by se k tomuto dala využít. Naproti tomu je Moodle mnohem rozsáhlejší než XML Check, nahradit Moodle systémem XML Check tedy určitě není možné.

Existuje několik dalších platforem pro e-learning, které obsahují podsystém pro odevzdávání domácích úkolů. Jedná se například o systém **Blackboard** [36], jehož pořízení je ovšem velmi nákladné. Podobných komplexních řešení existuje tolik, že je zde nemůžeme všechny vyjmenovat. Seznam takových řešení je možné najít třeba na [37]. Tyto systémy jsme se nepokoušeli srovnávat se systémem XML Check.

2.1.7 HWChecker

Na MFF byl vyvinut ještě jeden systém pro správu domácích úkolů, který ovšem nikdy nebyl uveden do reálného provozu. Tímto systémem je **HWChecker** od

Tadeáše Palusgy [2]. HWChecker byl vytvořen v roce 2012 v rámci bakalářské práce a bylo zamýšleno ho používat v předmětu *Databázové systémy* na FEL ČVUT.

- **Rozsah a rozšiřitelnost:** Systém přijímá archiv ZIP s libovolnými soubory. Jeho webové rozhraní navíc umožňuje učitelům, aby procházel obsah tohoto archivu, a soubory určitého typu (konkrétně soubory SQL, TXT, XML a obrázky) zobrazuje přímo (i se zvýrazňováním syntaxe) v prohlížeči. Pomocí pluginů se dá tento základ rozšířit o kontrolu správnosti a o kontrolu podobnosti s řešeními jiných studentů. Nainplementovány jsou pluginy dva: jeden, který plně kontroluje první domácí úkol předmětu *Databázové systémy*, a jeden, který kontroluje pouze přítomnost souboru s určitým názvem.
- **Uživatel:** Systém nebyl nikdy nasazen do provozu.
- **Důvod existence:** Některé úkoly z předmětu *Databázové systémy* mají mnoho požadavků, které se dají kontrolovat automaticky. Jejich ruční kontrola je naproti tomu velmi náročná (například zjišťování, zda v dotazech SQL jsou skutečně použity všechny vyjmenované konstrukce). Systém byl vyvíjen proto, aby tuto kontrolu usnadnil.
- **Historie a vývoj:** Systém vznikl v roce 2012. Systém samotný byl dokončen, nebylo ovšem vyvinuto dostatečné množství pluginů, aby byl systém v předmětu *Databázové systémy* užitečný. Systém nebyl nikdy otestován. Dále se nevyvíjí.
- **Popis:** Systém funguje z hlediska uživatelů velmi podobně jako XML Check. Studenti se přihlašují do skupin, ve kterých učitel zadává domácí úkoly. Soubory nahrané studentem projdou automatickým hodnocením, které se zobrazí studentovi. Jakmile je student s řešením spokojen, potvrdí ho a to se zobrazí učiteli, který si řešení zobrazí a oznámkuje ho.
- **Technologie:** Systém je naprogramován v jazyce Java, jako databáze je použita MySQL. Webové zobrazovací rozhraní využívá server Apache.
- **Upravitelnost a licence:** Zdrojový kód včetně návodu a dokumentace je k dispozici v repozitáři závěrečných prací Univerzity Karlovy pro nekomerční účely.

- **Kladné a záporné stránky:** V některých ohledech je HWChecker uživatelsky přívětivější než XML Check a práce s ním je rychlejší. Je také velmi univerzální a má zabudovaný i fungující systém pro kontrolu podobnosti. Není nicméně otestovaný a je možné, že je v něm množství chyb, navíc pro něj neexistuje mnoho pluginů.
- **Záměnnost:** Systémy XML Check a HWChecker jsou záměnné. Systém XML Check by bylo možné snadno nasadit v předmětu *Databázové systémy*. Stejně tak by jednoduché přepsání pluginů v systému XML Check do Javy umožnilo využívání HWCheckeru jako systému pro Technologie XML.

Na závěr této kapitoly bychom chtěli čtenáře upozornit, že jsme zdaleka nepopsali všechny systémy na správu domácích úkolů, o kterých víme. Pro nalezení dalších takových systémů doporučujeme hledat podle klíčových slov „homework manager“ nebo „assignment manager“.

2.2 Programy pro detekci plagiátů

Existuje větší množství programů, které se plagiátorství pokouší detekovat. V této kapitole uvedeme seznam takových programů, který není vyčerpávající, a pokusíme se ukázat, na jakém principu tyto programy plagiáty nacházejí.

Algoritmy pro porovnání, na které se zde odkazujeme, popíšeme v následující kapitole. Podrobnější seznam volně dostupných a komerčních programů lze najít v [39, str. 24-26].

2.2.1 Programy prohledávající web

Některé programy, když mají zjistit, zda je dokument plagiátem, jeho text zadají do internetového vyhledávače a zjistí, zda se jedná o plagiát, podle toho, zda vyhledávač vrátí nějaké výsledky. SkyLine, LLC. popisuje tento princip takto [41]:

1. Kontrolovaný dokument je rozdělen na věty.
2. Věty jsou odeslány do internetového vyhledávače (např. Google).
3. Výsledky vyhledávání jsou staženy.

4. Dokument je porovnán s každým výsledkem vyhledávání.
5. Podle toho program vyhodnotí, zda je dokument plagiátem nebo není.

Tento postup volí právě program **Plagiarism Detector** od této společnosti a také program **NetPlag** popsany v [39]).

2.2.2 Programy srovnávající s velkou vlastní databází

Masarykova univerzita udržuje systém **Theses.cz** [42], kam zástupci vysokých škol mohou nahrávat závěrečné práce svých studentů do centrální databáze. Takto systém využívá 42 českých vysokých škol.

Program pracuje pouze s pracemi ve formě čistého textu, z něž vybírá malé kusy dokumentu, které srovnává s kusy dokumentu jiných prací. Procento podobnosti s cizími pracemi je pracovníkovi vysoké školy zobrazeno a je mu umožněno si i prohlížet jednotlivé odstavce, kde systém našel podobnost s jinou prací.

Některé práce z Theses.cz jsou přístupné i v připojeném systému **Odevzdej.cz** (opět od Masarykovy univerzity). Rozdíl mezi oběma systémy je ten, že systém Odevzdej.cz může používat i široká veřejnost – studenti si tak např. mohou svoji práci zkontrolovat na přítomnost plagiátorství sami. Do systému Odevzdej.cz lze vložit navíc nejen závěrečné, ale i seminární práce a smí ho používat také i střední školy.

Mezi zahraniční systémy tohoto druhu patří **iThenticate**[40] od společnosti iParadigms, LLC, a s ním spojené systémy **TurnItIn** a **WriteCheck**, které všechny prohledávají stejnou databázi vědeckých prací, kterou společnost udržuje. Všechny tři systémy jsou komerční (porovnat jeden dokument pomocí iThenticate stojí 50 dolarů) a porovnávají pouze text v přirozeném jazyce (čeština ovšem není podporována).

Oba popsané systémy se zabývají pouze závěrečnými a vědeckými pracemi (což je pochopitelné, protože u nich má plagiátorství největší dopad), navíc porovnávají s vlastní databází místo s nahranými soubory, a nejsou tedy pro použití na srovnávání domácích úkolů vhodné.

2.2.3 Programy, které porovnávají jen zadané soubory

Mezi nejznámější programy, které porovnávají mezi sebou na podobnost jen současně nahrané soubory, patří **MoSS** (zkratka z *Measure of Software Similarity*) a **JPlag**. Jedná se o online služby. Uživatel se zaregistruje a poté může nahrát na servery dané služby sérii souborů, které jsou navzájem porovnány (porovnán je každý soubor s každým). Výsledek analýzy je zpřístupněn uživateli ve formě sestav HTML, ovšem v případě programu MoSS je uživateli zobrazena jen část analýzy (jen ty dvojice, o kterých si MoSS myslí, že je v nich pravděpodobnost plagiátorství největší).

Oba programy jsou určeny k porovnávání souborů zdrojových kódů, a tedy mají primárně sloužit učitelům programování. JPlag navíc umí porovnávat i text v přirozeném jazyce.

Program JPlag je založen na algoritmu Greedy-String-Tiling, který je popsán v následující kapitole.

Vzhledem k tomu, že ani jeden z těchto programů neumí pracovat se soubory XML, což je pro nás nejdůležitější, nemůžeme jich využít.

2.3 Analýza metod pro určování podobnosti

V této kapitole popíšeme různé metody, jakými se dá určovat podobnost textů, zdrojových kódů a souborů XML. Některé z těchto metod byly poté implementovány v modulu na kontrolu podobnosti v systému XML Check.

Všechny následující metody porovnávají vždy jen jeden soubor s jedním souborem. Navíc oba soubory musí být stejného druhu. Bylo by také možné srovnávat celá řešení namísto porovnávání jednotlivých souborů, nebo porovnávat např. soubor XML se souborem DTD, ale takové metody by pravděpodobně nepomohly odhalit opisování.

Dále jsme téměř všechny použité metody spouštěli až na souborech, kde byly bílé znaky odstraněny. Zdá se zřejmé, že se tím detekce plagiátorství jen vylepší: přidání přebytečných bílých znaků nebo naopak jejich odebrání je triviální změna, kterou může plagiátor provést; navíc se tímto sníží délka souboru a porovnávání tedy bude rychlejší. Pro větší přesnost byly také předem z textu odstraněny

podřetězce společné všem studentům, např. šablona zdrojového kódu a hlavičky funkcí v Javě.

V následujícím textu budeme „útokem“ chápat snahu plagiátora („útočníka“) zakrýt, že opisoval. Algoritmy se tedy budou jakoby „bránit“ tomuto útoku a snažit se, aby podobnost dokumentů, kterou stanovují, nebyla těmito útoky ovlivněna.

2.3.1 Identita

Stručné shrnutí: Soubory se porovnají znak po znaku. Pokud se rovnají, jsou označeny za navzájem od sebe opsané.

Identické porovnání je nejjednodušší možný algoritmus na určení plagiátorství. Jeho výhodou oproti všem ostatním metodám je mnohem vyšší rychlost; ovšem vzhledem k tomu, že identických souborů se v databázi moc nevyskytuje a všechny následující metody identické soubory jako podezřelé vyhodnotí taktéž, tak tuto metodu nepoužíváme.

Pro zajímavost jsme ji ovšem na databázi úkolů spustili a skutečně odhalila dvě řešení, která byla navzájem identická. Krátké vyšetření ukázalo, že obě řešení nahrál do systému stejný student, ovšem pod jinými uživatelskými účty (tento student má v systému dokonce tři různé účty, možná proto, že k nim zapomněl heslo (v té době ještě nebyla implementována funkcionality obnovy ztraceného hesla)).

2.3.2 Nejdelší společný podřetězec

Stručné shrnutí: Mezi soubory je nalezeno hladově deset nejdelších společných podřetězců. Jakmile je nejdelší podřetězec nalezen, je z obou souborů smazán a hledá se podřetězec další.

Tuto metodu používá systém HWChecker [2], nevhodně ovšem užívá výsledků porovnání: systém označí za podezřelý každý pár souborů, kde soubory mají dohromady alespoň tři společné podřetězce délky alespoň 10 znaků, z toho jeden délky alespoň 20 znaků. Je zřejmé, že takto bude označen za podezřelý každý domácí úkol v předmětu Technologie XML, už jen kvůli povinnému prologu XML⁴.

⁴Prolog XML je text `<?xml version="1.0" encoding="utf-8"?>`, který se podle specifikace

Zajímavostí v tomto algoritmu je, že kdykoliv je nalezen podřetězec, je z textu odstraněn před dalším hledáním, systém tedy dokáže detekovat pokusy o plagiátorství, kdy útočník vloží další text (např. komentář) doprostřed podobného bloku.

Vzhledem k tomu, že jsme nenašli žádnou vědeckou práci, která by se tímto algoritmem zabývala, tak jsme ho neimplementovali. Algoritmus ovšem funguje podobně jako algoritmus *Greedy-String-Tiling*, který implementován byl (viz dále).

2.3.3 Levenshteinova vzdálenost

Stručné shrnutí: Dynamickým programováním se spočítá počet znaků, které by bylo třeba změnit, přidat nebo odebrat, aby se z jednoho řetězce stal druhý. Například Levenshteinova vzdálenost řetězců *dokument* a *documents* je 2, protože stačí změnit *k* na *c* a přidat *s* na konec řetězce.

Levenshteinova vzdálenost tedy měří určitým způsobem podobnost textů. Používá se hlavně v programech na opravu překlepů – pokud uživatel zadá neexistující slovo, najdou se ve slovníku slova ve vzdálenosti 1 až 2 od zadaného slova a ta se uživateli nabídnou.

Vzdálenost identických dokumentů je 0, vzdálenost dvou naprosto rozdílných dokumentů bude rovná délce většího z nich. Z toho bychom tedy mohli sestavit měřítko odlišnosti na škále od 0 do 1: $\frac{d}{\max(A,B)}$, kde d je Levenshteinova vzdálenost obou dokumentů a kde A a B jsou délky těchto dokumentů.

Podobně jako výše uvedený algoritmus na hledání nejdelšího společného podřetězce je i Levenshteinova vzdálenost odolná vůči vložení textu doprostřed společného bloku, ovšem naprosto ji zmate prohození pořadí podobných částí. Prohodit pořadí je ovšem v dokumentu XML velmi snadné, protože se tím funkčnost nijak nezmění.

musí povinně nacházet na samém začátku každého dokumentu XML. Samozřejmě by porovnávací systém mohl tento prolog vyfiltrovat, ale problém zůstává: určité množství společných podřetězců se dá očekávat.

2.3.4 Algoritmus Greedy-String-Tiling

Stručné shrnutí: Najde se největší společný neoznačený podřetězec, který se označí. Toto se opakuje, dokud nezbydou pouze takové podřetězce, které mají menší délku než stanovený limit.

Algoritmus Greedy-String-Tiling byl navržen v roce 1993 v [16]. Algoritmus srovnává dva texty nebo také řetězce tokenů. Byl totiž navržen zčásti pro detekci plagiátorství v počítačových programech, kde je možné například klíčové slovo programovacího jazyka považovat za jediný token.

Algoritmus podrobně popisuje jak původní [16], tak i [17, str. 10-14] a [18, str. 5-7], z jejichž analýz navíc vyplývá, že program JPlag (který používá právě tento algoritmus) detekuje plagiátorství ve zdrojových kódech velmi spolehlivě.

Parametrem algoritmu je celé číslo pojmenované *MinimumMatchLength* (MML). Nastavení tohoto parametru má zásadní vliv na výsledky porovnání. Toto číslo by mělo označovat počet tokenů, které když se vyskytnou za sebou ve dvou různých dokumentech, tak je možné začít mluvit o podobnosti. Greedy-String-Tiling totiž v zásadě funguje tak, že v obou dokumentech vyhledá co nejdelší podřetězec, jejichž délka je alespoň *MinimumMatchLength*. Součet délek těchto podřetězců odpovídá podobnosti obou dokumentů, kterou můžeme získat například tímto vzorcem: $\frac{2 \cdot m}{A+B}$, kde m je počet označených tokenů, A je délka prvního řetězce a B je délka druhého řetězce.

Zásadní výhodou zde je, že pokud útočník prohodí různé části souboru, tak pokud tyto části mají délku alespoň *MinimumMatchLength*, tak jsou rozpoznány jako podobné.

Jak jsme zmínili, určení parametru *MinimumMatchLength* může mít na úspěch algoritmu velký vliv a nejlepší možný parametr nelze snadno zjistit. Vyzkoušeli jsme na našich datech dvě různé hodnoty (MML = 2 a MML = 8).

2.3.5 Algoritmus podobnosti XMLUnit

Stručné shrnutí: Stromy obou dokumentů XML se musí shodovat ve všem až na pořadí sourozeneckých elementů.

V programu HWChecker jsou dokumenty XML porovnávány funkcí *similar*

knihovny XMLUnit [14]. Tato knihovna obsahuje funkce pro automatické testování programů pracujících s XML. Její funkce *similar* porovnává dva dokumenty XML tak, že se musí shodovat ve všem (kostra, jména uzlů, text, ...) kromě bílých znaků a pořadí sourozeneckých elementů.

To je velmi přísné porovnání a plyne z něj, že cokoliv je určeno za podezřelé, skutečně také plagiátem je. Nicméně studentovi stačí přidat jediný uzel nebo atribut nebo změnit název jednoho uzlu, a funkce *similar* vrátí *false*.

Vzhledem k tomu, že jiné popsané metody takto podobné dokumenty také označí za podezřelé, není třeba ztrácet procesorový čas aplikací tohoto porovnávacího algoritmu.

Tato metoda tedy není použita.

2.3.6 Algoritmus editační vzdálenosti Zhang-Shasha

Stručné shrnutí: Dynamickým programováním se spočítá počet uzlů, které je třeba odstranit, přidat nebo přejmenovat, aby se z jednoho stromu stal druhý strom.

V [19] popisují autoři metodu porovnávání dvou stromů, která určí jejich vzdálenost podobně jako Levenshteinova metrika určuje vzdálenost mezi dvěma řetězci. Povolují operace *přejmenování uzlu* (což v případě souborů XML znamená změnu jména elementu nebo atributu), *vložení uzlu* a *smazání uzlu*. Každé operaci je možné přiřadit jinou váhu. Vzdálenost dvou stromů je pak součtem vah všech použitých operací. Autoři popisují algoritmus, který v čase $O(m \cdot n \cdot \min(\text{depth}(T_1), \text{leaves}(T_1)) \cdot \min(\text{depth}(T_2), \text{leaves}(T_2)))^5$ dokáže určit nejmenší takový součet.

Výhodou této metody je, že malým změnám struktury přiřazuje skutečně jen malý význam. Vložení uzlu doprostřed stromu tak zvýší vzdálenost jen o váhu jedné operace vložení. Tento fakt ovšem paradoxně v našem použití této metodě spíše škodí. Student totiž kromě stromu XML musí dodat i schéma popisující tento strom a přidat nový uzel doprostřed stromu je pro útočníka poměrně obtížné

⁵Zde m značí počet vrcholů prvního stromu, n počet vrcholů druhého stromu, T_1 a T_2 značí první, respektive druhý strom, $\text{depth}()$ hloubku daného stromu a $\text{leaves}()$ počet listů daného stromu.

oproti jiným metodám útoku.

Metodu jsme implementovali.

3. Popis provedené práce

3.1 Oprava chyb

První částí této práce je odstranění chyb v systému. Před začátkem letního semestru 2014 jich bylo známo vyučujícím celkem sedm.

V průběhu semestru ovšem jak studenti, tak vyučující nacházeli nové nedostatky a chyby. Je zvláštní, že se tolik chyb objevilo po několika letech používání až nyní. Většina těchto chyb nebyla způsobena aktualizacemi systému během semestru. Některé z nich nejspíše nahlásili vyučující až teď, přestože o nich věděli již z dřívějších let, protože se systém začal opravovat. Jiné jsou obskurní chyby, které se neprojevovaly příliš často a byly objeveny až během podrobného zkoumání systému a jeho zdrojového kódu.

Během semestru bylo nahlášeno celkem 47 chyb, z toho deset nahlásili studenti a zbytek vyučující, nebo je našel autor práce. Z těchto bylo opraveno 39 chyb. Zbylé chyby nebyly opraveny z větší části proto, že se nepodařilo je reprodukovat. Další chyby byly ještě nalezeny a opraveny až později.

Většina chyb byla způsobena chybným znakem, slovem nebo několika málo řádky ve zdrojovém kódu. Identifikovat, kde se chyba nachází, bylo ovšem leckdy obtížné.

V této kapitole jsou popsány jen vybrané opravené chyby. Úplný seznam je v příloze C.

3.1.1 Parsování DTD souborů

Prvním domácím úkolem pro studenty předmětu je vytvořit vlastní soubor XML, který bude správně zformovaný a popisovat skutečnost zvolenou studentem. Zároveň má student vytvořit definiční soubor DTD, pomocí kterého bude možné soubor XML validovat. Pokud by si student po dohodě s cvičícím vybral například téma *firma*, mohl by soubor XML obsahovat informace o odděleních v této firmě, o tom, kdo pracuje v jakém oddělení, osobní informace o zaměstnancích, vztahy mezi zaměstnanci a například také informace o zakázkách firmy.

Existují navíc požadavky na minimální složitost, konkrétně:

- Dokument musí mít hloubku stromu alespoň pět.
- Alespoň jeden element musí mít alespoň 10 synů.
- Asociovaný dokument DTD musí obsahovat většinu konstrukcí probíraných během přednášky (zadání úkolu je vyjmenovává).
- Buď dokument XML nebo dokument DTD musí obsahovat vlastní entitu, instrukci ke zpracování, sekci CDATA a komentář.

Těmito požadavky je zajištěno, že student pro odevzdání úkolu musí porozumět jazyku XML natolik, aby použil jeho pokročilejší funkce. Mnozí studenti na školách, kde se tento předmět vyučuje, totiž znají a používají jazyk XML, nikoliv ovšem jeho obskurnější možnosti.

Kontrolovat tyto požadavky ručně je ale zdlouhavé a náročné. Hloubku stromu nebo použití všech jedenácti povinných prvků nedokáže cvičící zkontrolovat bez toho, že by něco nepřehlédl. Systém v původní verzi se tedy tyto požadavky snažil kontrolovat automaticky, ovšem využíval ke kontrole souboru DTD balíček XML_DTD¹ z kolekce PEAR², který byl však v roce 2010, kdy systém XML Check vznikl, ještě ve verzi alfa, a přitom již nebyl vyvíjen.

Parser uvnitř balíčku XML_DTD aplikuje regulární výrazy na celý soubor, pomocí nichž extrahuje deklarace elementů a atributů. Používání regulárních výrazů ovšem u jazyka, který vyžaduje správné uzávorkování v podobě podmíněných sekcí, nemůže pokrýt všechny případy. Jazyk XML navíc podporuje použití *parametrických entit*, které fungují jako makra. Ta lze rozvinout na místě použití do jiného textu, který sám může obsahovat parametrické entity.

Převážně následkem nedostatků tohoto balíčku se v kontrolování prvního domácího úkolu v původní verzi vyskytovaly tyto problémy:

- Kvůli chybě v knihovně *libxml2* nelze uvnitř hodnoty notace mít znak procenta, ač to specifikace XML povoluje.
- Podmíněné sekce v souboru DTD jsou ignorovány, což způsobí nesprávné načtení souborů, které tyto sekce používají.
- Notace jsou ignorovány.

¹http://pear.php.net/package/XML_DTD

²PEAR (PHP Extension and Application Repository, <http://pear.php.net/>) je repozitář knihoven PHP. Jeho instalace a správa není ovšem jednoduchá a v době psaní této práce byl nahrazen systémem Composer.

- Entity nejsou správně nacházeny a rozvíjeny.
- V hodnotách atributů je zamítnuto použití znaku > i na místech, kde ho standard povoluje.
- Pokud uživatel použije kódování UTF-16, kontrolující plugin selže nebo zobrazí nesprávný výsledek.
- Elementy se smíšeným obsahem nejsou detekovány.
- Soubor DTD, který není správně zformovaný, projde v některých případech kontrolou.
- Definice elementů uvnitř samotného XML souboru místo uvnitř DTD souboru nejsou detekovány.

Některé z těchto nedostatků pocházely z jiných částí kódu než z balíčku XML_DTD, a bylo tedy možné je celkem snadno opravit. Některé z nich ovšem vyžadovaly použití jiného způsobu parsování souboru DTD než pomocí regulárních výrazů.

Neexistuje ovšem pro jazyk PHP žádný jiný balíček, který by soubory DTD zpracovat dokázal.

Jazyk PHP obsahuje možnost, jak zkontrolovat, že daný soubor DTD je správně zformovaný.[26] Jeho zabudovaný parser pro XML totiž přiložené soubory DTD kontroluje, nezpřístupňuje ovšem definice uvnitř programátorovi. Student by tak mohl obelhat systém například tak, že by použil následující soubor DTD:

1 `<!ELEMENT myRootElement ANY>`

Specifikace ANY říká, že obsah elementu může být libovolný. Zbytek souboru by mohly tvořit podobně vágní definice ostatních použitých elementů a atributů. Dohromady by soubory XML a DTD byly správně zformované a validní, přesto student zjevně nevyřešil úkol správně.

V podmínkách domácího úkolu dále stojí, že student musí deklarovat atributy typu ID a IDREF a tyto je třeba použít. Není ovšem chybou mít v souboru DTD nevyužití elementy, tedy je možné, že by student nadefinoval element s atributem ID a pak ho nepoužil. Také z tohoto důvodu je třeba umět parsovat soubor DTD.

Naším úkolem je vytvořit co nejlepší automatickou kontrolu; v tomto případě napsáním vlastního parseru souboru DTD jsme mohli odstranit všechny výše

uvedené chyby, a zjemnit tak automatickou kontrolu, aby bylo odhaleno více chyb a naopak správné soubory nebyly považované za chybné.

Nový parser je obsažen v jednom souboru, *SoothsilverDtdParser.php*. Protože se dá využít i mimo použití v systému XML Check, byl zapojen do databáze *Packagist*³, která schraňuje knihovny pro PHP.

Parser obsahuje jednu třídu (*DTD*), která pomocí metody *parseText* načte text souboru DTD, popř. také text interní podmnožiny dokumentu XML a znak po znaku projde text od začátku do konce. Během parsování si ukládá informace o definicích jednotlivých elementů, notací a entity, které pak zpřístupňuje uživateli. Parser si je vědom komentářů a podmíněných sekcí, které ignoruje, pokud se jedná o sekce typu IGNORE. Zároveň také umí provádět správné rozvinutí parametrických entit.

Podle automatických testů a na základě testování na datech z minulých ročníků funguje nový parser lépe než předchozí a nedělá chyby nikde, kde by se chybám vyhnul starý parser.

Parser se snaží splňovat specifikaci XML, ovšem takový úkol je velmi obtížný a parser ho nesplňuje, což ovšem nemusí: správnost dokumentu DTD dokáže ověřit i knihovna *libxml2*, kterou systém XML Check také používá. Parser tedy může být použit až na správně zformovaný dokument DTD, ze kterého dokáže získat správné informace.

Parser neumožňuje přistupovat k elementům a atributům sekvenčně, podobně jako například přistupuje parser SAX k souborům XML. Funguje tak, že nejprve přečte celý soubor, vytvoří datové struktury pro uložení definic elementů a atributů, a teprve pak je zpřístupní uživateli. To je zcela dostačující a výhodné pro použití v XML Checku, ovšem pro uživatele, kteří by chtěli sekvenční přístup, stále ještě v PHP neexistuje žádná alternativa.

3.1.2 Výkon tabulek

V původní verzi se tabulky vykreslovaly pomalu. Většina tabulek neměla více než deset řádků, takže to nebylo znát a student téměř nikdy nepoznal, že by vykreslování bylo pomalé. Na straně cvičícího a administrátora ovšem již toto

³<https://packagist.org/>

působilo problémy.

Tabulka uživatelů měla stovky řádků a i tabulka odevzdaných řešení byla někdy poměrně velká, obzvláště pokud daný cvičící měl na starosti více než jedno cvičení. Protože se navíc tabulky překreslovaly po každé změně, musel cvičící čekat na překreslení tabulky pokaždé, když oznámkoval nějaké řešení.

Jan Konopásek navrhoval v závěru své práce, že by systém vykreslování mohl být přepracován tak, aby se vykreslila vždy jen část tabulky, a to ta, kterou uživatel skutečně uvidí. Ukázalo se ovšem, že znatelného zrychlení můžeme dosáhnout i bez takové změny.

Nejvíce času během vykreslování zabírá generování DOM podstromu těla tabulky. Je to proto, že nejprve je vygenerována řádka, pak je na každou položku řádky aplikován widget *text-cutter* a pak jsou ke každé řádce přidány akční tlačítka. Aplikace widgetu *text-cutter* je časově náročná, proto se nyní aplikuje pouze na položky, jejichž obsah je dostatečně dlouhý, aby byl tento widget využit.

Problém s aplikací akčních tlačítek byl v tom, že se místa pro vložení tlačítek vybírala pomocí jQuery selektorů, které pracovaly s celou tabulkou. Takový výběr je ovšem pomalý. Navíc během vkládání tlačítek bylo vytvářeno a ničeno více prvků DOM stromu, operace s DOM stromem jsou taktéž pomalé.

Kód, který generuje obsah tabulky, byl tedy přepracován tak, aby nepoužíval jQuery selektory tam, kde to není nutné, a byly provedeny některé další změny. Výsledkem je znatelné zrychlení vykreslování tabulek.

3.1.3 Plugin mohl tiše selhat

V původní verzi, pokud kvůli chybě v kontrolujícím pluginu došlo během provádění kontroly k výjimce nebo k PHP chybě, interpret PHP se ukončil bez toho, aby zapsal výsledek kontroly do databáze. Student tak v uživatelském rozhraní měl neustále zobrazeno jen, že se řešení vyhodnocuje, ačkoliv se již vyhodnocovat přestalo (neúspěšně).

Plugin může selhat a vyhodit výjimku z více důvodů, ovšem vždy pouze, pokud není naprogramován správně, tj. je v něm bug. Bohužel v původní verzi bylo těchto bugů několik (viz příloha C). Na jednoduchých datech ovšem byly pluginy testovány, plugin tedy selhával, pokud student nahrál řešení něčím neobvyklé.

Takové řešení mohlo být správně nebo nemuselo, v každém případě se o tom student ale nedozvěděl, musel odevzdat úkol ručně (e-mailem) cvičícímu, a mohl tak dostat horší ohodnocení, protože mu systém nenapověděl, co má špatně.

Nyní tedy, pokud se uvnitř pluginu stane nějaká chyba, popř. výjimka, chyba PHP nebo plugin v Javě vrátí systému neplatná data, bude toto vždy zaznamenáno do databáze. Jediný případ, kdy by toto nenastalo, je, pokud by se v pluginu vyskytla parsovací chyba PHP. Odhalit takovou chybu je ovšem velmi snadné, protože nezávisí na vstupních datech, a odhalují ji již vývojová prostředí. Navíc by plugin s takovou chybou nemohl projít automatickými testy.

3.1.4 Zadání nebyla dostatečně srozumitelná

Zadání domácích úkolů sice byla celkem úplná, někteří studenti přesto odevzdávali řešení, která nebyla správně z důvodu nepochopení zadání, a nad tím, co konkrétně zadání vyžaduje, musel student delší dobu přemýšlet.

Není ovšem smyslem domácích úkolů z Technologií XML naučit se rozumět nepřesným požadavkům, proto byla všechna zadání rozšířena a upřesněna tak, aby byla naprosto vyčerpávající.

To spočívalo např. v tom, že byla vyjasněna adresářová struktura, kódování nebo šablona pro zdrojové kódy v Javě, ovšem na mnoha místech byly vyjasněny i samotné požadavky na složitost. Kupříkladu domácí úkol č. 4 (schéma XSD) měl v zadání pokyn, že soubor má obsahovat „povinné i nepovinné atributy“. Znamená to, že je třeba u nepovinných atributů explicitně nastavit hodnotu *use* na *optional*, nebo je možné využít toho, že specifikace XSD považuje *optional* za výchozí hodnotu *use*?

Mírnou nevýhodou takové změny zadání je, že úkoly se tím pádem staly jednoduššími, a student se tak možná naučí o dané XML technologii méně.

3.1.5 Plugin pro DOM/SAX selže při výpisu na obrazovku

Jako příklad chyb, které byly odstraněny z kontrolujících pluginů pro jednotlivé domácí úkoly, uvedme nejznatelnější chybu pluginu pro DOM/SAX.

Ve 2. domácím úkolu (DOM/SAX) má student za úkol vytvořit dva soubory

v Javě ze zadané šablony. V jednom z těchto souborů má být třída, která pomocí transformací DOM upraví studentův XML soubor. V druhém souboru má být třída, která pomocí SAX získá nějaké informace ze studentova XML souboru a vypíše je na standardní výstup. Kontrolující plugin oba zdrojové soubory přeloží kompilátorem Javy a spustí.

V druhém úkolu jsou tedy dvě části: v první části má program měnit strom DOM, v druhé má psát na standardní výstup. Aby cvičící mohl snáze zkontrolovat, že studentův program funguje, kontrolující plugin tento výstup přesměruje do souboru a ten cvičícímu umožní stáhnout.

Plugin ovšem nepřesměřoval výstup z první části tohoto úkolu, protože tam neměl student za úkol nic na standardní výstup vypisovat. Pluginy v Javě ovšem v systému XML Check komunikují s PHP kódem tak, že na standardní výstup vypíší XML data popisující, na kolik procent student daný úkol splnil a obsahující také textový komentář (chybové hlášky). Pokud ovšem student v první části sám vypsál něco na standardní výstup, předcházel pak tento výstup deklaraci XML hlavičky a PHP kódu se již samozřejmě nepodařilo správně pochopit výstup z pluginu, byla vyhozena výjimka, a studentovi se tedy zdálo, že systém jeho úkol stále opravuje, přestože již dávno skončil s neúspěchem (viz také 3.1.3).

3.2 Přidání nové funkcionality

Druhou částí práce je přidání nové funkcionality do systému. Během semestru se ve sdílené tabulce s žádostmi o nové funkce objevilo celkem 37 položek. Z těchto návrhů bylo implementováno 28 funkcí, zbývajících 9 nebylo implementováno z větší části proto, že by bylo příliš náročné je implementovat a systému by přispěly jen málo nebo vůbec.

Konopásek v závěru své práce uvedl množství vylepšení, která by dle jeho názoru nebo podle uživatelů v roce 2011 mohla být užitečná, ovšem do systému je již nepřidal z důvodu nedostatku času. [1, sekce 5.5]

Těmito vylepšeními jsou:

- **Zobrazovat datum odevzdání v seznamu řešení.** Implementováno, aby studenti mohli rozlišit mezi svými odevzdanými řešeními.

- **Uživatel si může sám vybrat, které sloupečky v tabulkách zobrazovat a které ne.** Neimplementováno, protože je náročné na implementaci, způsobilo by nárůst složitosti systému a funkci by nevyužívalo mnoho uživatelů.
- **Přidávat slovní komentář k obodovanému řešení.** Implementováno. Cvičící nyní může studentům kromě udělení počtu bodů dát i slovní komentář k jejich řešení. Tato funkce byla implementována těsně před koncem letního semestru 2014, i přesto ji cvičící ve zbytku semestru využívali.
- **Přidat funkci pro obnovu hesla.** Implementováno. Konopásek ve své práci navrhoval posílat nově vygenerované náhodné heslo na uživatelův e-mail. Implementovaná varianta je trochu výhodnější v tom, že uživatele nutí si poté okamžitě heslo změnit, protože náhodně vygenerované heslo by nejspíše zapomněl.
- **Omezit počet nahraných řešení ke každému úkolu.** Neimplementováno. Důvod pro toto vylepšení je dvojitý: zaprvé by automatická kontrola úkolů mohla být příliš náročná na výpočetní čas, a zadruhé, u úkolů, kde většinu ohodnocení provádí automatická kontrola, by student byl nucen číst pozorně zadání a svoje řešení mít správně napoprvé. Například systém CodEx takové omezení umožňuje, přestože ve většině situací, kde je používán, je limit počtu nahraných řešení nastaven na 50. Nemyslíme ovšem, že by systém XML Check tuto funkcionalitu potřeboval.
- **Zrychlit zobrazování tabulek vylepšením jejich modelu.** Částečně implementováno. Tabulky v systému XML Check fungují tak, že stáhnou veškerá data ze serveru a vygenerují z nich DOM strom pomocí Javascriptu (jsou vygenerovány všechny řádky). Poté tyto řádky již ve formě elementů DOM seřadí, a ty, které se nevejdou na obrazovku (kvůli stránkování) se skryjí pomocí stylů CSS. To zabíralo klientským prohlížečům příliš mnoho času, během kterého je aplikace jakoby zamrzlá. Konopásek navrhoval, že by se zdrojový kód tabulek mohl změnit tak, že by se generovaly pouze řádky, které se skutečně zobrazí na obrazovku. To ovšem není ve stávající architektuře nijak snadné. Místo toho byl zdrojový kód tabulek upraven na

několika místech tak, že se zachovala stejná funkcionální (a stejný princip zobrazování), ale rychlost generování a vykreslování se významně zvýšila.

- **Přidat filtrování na straně serveru.** Neimplementováno. Bylo by sice možné filtry, které zadá uživatel v tabulce, aplikovat na straně serveru, aby se muselo zobrazovat méně řádků, ovšem testování ukázalo, že stahování dat ze serveru a jejich filtrování zdaleka nejsou hlavním úzkým hrdlem v procesu zobrazování tabulky. Implementace této funkcionality by byla složitá a výkonu by téměř nepřispěla.
- **Umožnit přidávání příloh k problémům.** Neimplementováno. Žádný z šesti domácích úkolů předmětu Technologie XML přílohu k zadání nepotřebuje.
- **Umožnit řešení testů online.** Neimplementováno. Systém XML Check sice obsahuje podsystém pro vytváření a tisk testů, za celou dobu existence systému nebyl tento podsystém nikdy využit. Zdá se tedy, že o takovou funkcionální není v předmětu zájem.
- **Vylepšit chybové hlášky vrácené kontrolujícími pluginy.** Implementováno. Chybové hlášky pluginů byly na mnoha místech upraveny, aby byly srozumitelnější. Na jednom místě byla přidána zvláštní chybová hláška upozorňující na bug v knihovně *libxml2*, který pravděpodobně nebude opraven.

V této kapitole jsou dále popsány některé z největších nebo nejdůležitějších nových funkcí, které byly do systému přidány. Ostatní lze najít v přílohách D a E.

3.2.1 Vícejazyčnost

V původní verzi byly všechny texty zobrazené uživateli napsány přímo ve zdrojových souborech aplikace, a to v angličtině. Existoval ovšem zájem ze strany studentů používat systém v češtině. Možností by tedy bylo přeložit všechny textové řetězce do češtiny, což by bylo dostačující pro předmět Technologie XML, který se v jiném jazyce než v češtině nevyučuje.

Na druhou stranu by to znemožnilo použití systému mimo Českou republiku, což by byl krok zpět od původní verze, která byla v angličtině. Z těchto důvodů byla

do systému přidána podpora více jazyků. Na úvodní obrazovce před přihlášením a pak také v menu *Jazyková nastavení* uvnitř aplikace si může uživatel nyní zvolit jazyk pro zobrazení. Lze volit mezi češtinou a angličtinou, přidat další jazyky by ovšem bylo přímočaré.

Některé obskurnější chybové hlášky se ovšem stále zobrazují pouze v angličtině, stejně tak chybové hlášky kontrolujících pluginů. Je to proto, že zavést podporu vícejazyčnosti do pluginů by znamenalo, že by se databázová struktura musela zásadnějším způsobem změnit a byla by komplexnější, a protože mnoho chybových hlášek pluginů pochází z externích knihoven, které svůj výstup produkují výlučně v angličtině.

Implementaci popisujeme v sekci 6.7.

3.2.2 Automatické testy

Velká část změn provedených na systému byla provedena během provozu, konkrétně v letním semestru 2014. Každou změnou ovšem riskujeme zavlečení nové chyby do systému.

Je možná pravda, že chyba, která systém učiní nefunkčním, nebude mít vážných následků. Pokud by systém přestal fungovat úplně, studenti by posílali svá řešení e-mailem. Cvičící by je museli třídit a hodnotit ručně, čímž by se možná stalo, že některá špatná řešení by prošla s plným počtem bodů. Po obnovení funkčnosti systému by ovšem studentům chyběl v systému záznam z tohoto úkolu. Protože termín již mohl uběhnout, studenti by svoje řešení nemohli nahrát do systému zpětně (v tu dobu ještě nebylo možné nahrát řešení až po termínu).

V každém případě by to způsobilo nepříjemnosti a chtěli bychom se chybám v provozu vyhnout. Proto byly do systému přidány automatické testy kontrolující správnost. Automatické testy byly vytvořené pomocí knihovny PHPUnit, což je PHP knihovna pro psaní testů, původně určená pro jednotkové testy.

Používáme testy dvou druhů:

- **Jednotkové testy**, které kontrolují funkčnost ucelené části aplikace. Pomocí jednotkových testů bylo odhaleno několik chyb v kódu, ovšem jednotkovými testy zdaleka nepokrýváme celou aplikaci. Část, která běží na Javascriptu, netestujeme vůbec.

- **Testy pluginů**, které kontrolují správnost kontrolujících pluginů. Pro každý ze šesti domácích úkolů je připravena sada testovacích dat spolu s očekávanou správností těchto dat, popř. s očekávanou chybovou hláškou.

Pravidelné spouštění testů způsobí, že je menší šance zavlečení chyby, která by způsobila, že existující kód přestane fungovat. Automatických testů ovšem nemáme zdaleka dostatek. Naprosto bez testů zůstává celá uživatelská část aplikace napsaná v Javascriptu, která se dá testovat jen obtížně. Interakce s databází také zůstává z větší části neotestovaná.

Obzvláště testy pluginů ovšem výrazně pomohly s udržováním korektnosti.

3.2.3 Nový systém potvrzování úkolů

V původní verzi systému XML Check fungovalo odevzdávání úkolů následujícím způsobem:

1. Student vyřešil úkol a nahrál do systému archiv ZIP. Systém ho ohodnotil 0 až 100 procenty, popř. vypsal chybové hlášky. Na základě této zpětné vazby student mohl úkol upravovat a nahrávat znovu, dokud s ním nebyl spokojen. Nahrávat nová řešení student mohl, dokud neuplynul termín odevzdání nebo dokud některé své řešení nepotvrdil (viz dále).
2. Dokud neuplynul termín odevzdání, mohl student stisknout u nahraného úkolu tlačítko **Potvrdit**, čímž uzamknul svoje řešení: od této chvíle mu již systém nedovolil svoje řešení změnit. Pokud tedy student našel ve svém řešení chybu po potvrzení, ale stále ještě před termínem, nemohl již svoje řešení změnit.
3. Stiskem tlačítka *Potvrdit* se odevzdané řešení zpřístupnilo cvičícímu.
4. Cvičící řešení ohodnotil počtem bodů (většinou od 0 do 20). Pokud se cvičící spletl, mohl svoje ohodnocení ještě změnit.

S tímto systémem potvrzování úkolů je spojeno několik problémů:

- **Odevzdání po termínu není možné.** Celkem často se stávalo, že student nestihl odevzdat úkol do termínu. Když ho pak chtěl odevzdat třeba několik

minut po termínu, systém mu to neumožnil. Toto bylo opraveno ještě během letního semestru 2014 tak, že student sice mohl potvrdit po termínu, ovšem cvičicímu se zobrazila informace, že student odevzdal pozdě, a mohl tedy studentovi strhnout body.

- **Studenti zapomínají potvrzovat úkoly.** Také se často stávalo, že student sice úkol vyřešil a nahrál do systému, ale zapomněl ho potvrdit, a tedy cvičicímu se úkol nezobrazoval a nemohl ho tak ohodnotit. Obzvláště na začátku semestru také někteří studenti nevěděli, že úkoly je třeba potvrzovat.
- **Po potvrzení nemůže student svoje řešení upravit.** Ve snaze nezapomenout na potvrzení tedy student bude chtít svoje řešení potvrdit co nejdříve. Pokud ho ovšem potvrdí příliš brzy, může poté přijít na nápad, jak své řešení ještě vylepšit, což už mu systém neumožní. Toto vytváří pro studenta dilema, kterému se chceme vyhnout.

Ve své práci Konopásek vysvětluje, proč zvolil tuto metodu pro potvrzování úkolů: [1, sekce 3.3.6]

„V této sekci jsou popsány dva klíčové body procedury odevzdání řešení. Jedním z nich je možnost smazat řešení na základě výsledků automatické kontroly předtím, než student řešení odevzdá k ohodnocení. Hlavní motivací k tomuto je předpoklad, že čas cvičících je mnohem důležitější než výpočetní čas. Také se předpokládá, že hlavním důvodem pro domácí úkoly je cvičení, nikoliv zkoušení. To je pravda pro prostředí, kde je AsM [*Assignment Manager, původní název systému XML Check*] použit nyní, není to pravda pro všechny možné případy. Pro nápravu by mohl být ustaven limit na počet řešení, které může jeden student nahrát k jednomu úkolu.

Dalším klíčovým bodem v proceduře odevzdání řešení je nemožnost vrátit řešení autorovi pro úpravy, jakmile bylo řešení potvrzeno. Toto omezení je vynucováno nejen kvůli komplikacím asociovaným s implementací takové možnosti, ale také v zájmu spravedlivosti. Pokud by bylo možné vracet potvrzená řešení, musela by být implementována

dodatečná ochranná opatření, která by zajistila, že studenti budou včas informováni o vrácení svých řešení a že budou mít dost času na odevzdání nových řešení před termínem.“ *(přeložil autor této práce)*

Ztotožňujeme se s prvním citovaným odstavcem; zdá se ovšem, že problém v druhém citovaném odstavci se dá vyřešit. Pokud by se tak stalo, mohl by být systém přepracován, aby zanikly tři výše popsané problémy se současnou metodou. Odstavec popisuje dva problémy: složitost implementace a spravedlivost. Implementovat jiné řešení je možné, jak uvidíme později.

Co se týče spravedlivosti, určitě by byl problém, kdyby cvičící vrátil studentovi úkol k přepracování a nedal mu na to dostatečný čas. To ovšem cvičící dělat nemusí a systém mu to nemusí umožňovat, i když povolí studentovi odevzdat řešení i po potvrzení.

Jak tedy můžeme systém upravit, abychom se zbavili třech uváděných nevýhod a přitom ponechali funkční systém?

Jednou možností by bylo zobrazovat cvičícímu veškerá studentova řešení. Takto funguje například systém CodEx. V systému CodEx se ovšem od cvičícího očekává, že bude studentovo řešení prohlížet jen ve výjimečných případech. V typickém případě použití systému XML Check by byl cvičící zahlcen mnoha řešeními.

Další možností je zobrazovat cvičícímu vždy jen poslední řešení odevzdané studentem. V takovém případě uvidí cvičící od jednoho studenta vždy nejvýše jedno řešení (jako doposud), a to většinou to nejlepší. Pokud student odevzdá nové řešení horší než své předchozí, může své předchozí nahrát znovu, protože není stanoven limit na počet nahraných řešení. Alternativně by mohl nové špatné řešení smazat. Může ovšem nastat jiný problém: cvičící může opravit řešení studenta ještě před termínem; někteří cvičící zadávají všechny úkoly na začátku semestru, schopný student je tak může vypracovat všechny hned na začátku, a pokud mu je cvičící opraví před termínem, studenta tak obere o možnost chybu opravit ještě před ohodnocením.

Byla tedy implementována poslední diskutovaná možnost s malým vylepšením: totiž, že student může tlačítkem „upozornit učitele“ svoje řešení označit za finální, a požádat tak učitele, aby ho opravil předčasně. Cvičící dostane e-mail a v rozhraní se mu takové řešení zobrazí zvlášť a rozhraní cvičícího upozorní, že si student přeje

být oznámkován předčasně. Od cvičících se pak očekává, že budou před uplynutím termínu hodnotit pouze ta řešení, na která byli upozorněni.

Technicky je implementován proces tak, že každé řešení má svůj „stav“: *nahráno*, *poslední*, *učitel upozorněn* nebo *oznámkováno*. Kdykoliv student nahraje řešení, jeho dosavadní *poslední* řešení změní svůj stav na *nahráno*, a nové řešení se stane *posledním*. Student může maximálně jedno své řešení označit jako *učitel upozorněn*, a toto označení může přehazovat mezi svými řešeními. Řešení, které bylo předtím označeno stavem *učitel upozorněn* získá stav *nahráno*. Když cvičící ohodnotí nějaké řešení (cvičící vidí jen řešení se stavem *poslední* nebo *učitel upozorněn*), toto získá stav *oznámkováno* a všechna ostatní řešení ho ztratí. Student tedy může mít u jednoho úkolu zároveň jedno řešení označené *učitel upozorněn* a jedno řešení označené *poslední*; nepředpokládá se ovšem, že by tohoto student často využíval.

Dovolujeme navíc studentovi smazat řešení, na které již upozornil učitele nebo upozornit učitele na jiné své řešení, přestože již na jedno upozornil. Stejně tak dovolujeme studentovi nahrávat nová řešení po termínu. Toto ovšem může způsobit problémy s časováním: představme si situaci, kdy student nahraje řešení a termín uplyne. Cvičící si nějaký čas po termínu stáhne studentovo řešení a bude ho kontrolovat. Zatímco ho kontroluje, student si uvědomí nějakou chybu ve svém řešení, a myslí si, že je natolik zásadní, že se mu vyplatí riskovat bodový postih za pozdní odevzdání, opraví ji tedy a nahraje svůj nový soubor. Cvičící pak studentovo původní řešení dokontroluje a zadá mu počet bodů za toto řešení. Nyní má tedy student v databázi jedno obodované řešení (to staré) a jedno pozdě odevzdané řešení, které se nyní bude zobrazovat cvičícímu. Záleží teď na cvičícím, zda toto nové řešení ohodnotí nebo zda ho bude ignorovat. Neočekáváme, že by se tato situace stávala často a možnost studentů opravit své chyby nebo špatné kliknutí, kterou takto získáme, za to stojí.

3.2.4 Odesílání e-mailů

V původní verzi systém posílal uživateli e-mail jen v jednom případě, a to během registrace, aby zkontroloval, že uživatel zadal svojí vlastní e-mailovou adresu. Tuto adresu přitom poté nijak nevyužíval, její zadávání a kontrola tedy ani nebyly užitečné.

Nyní systém XML Check odesílá e-maily při více příležitostech, konkrétně:

- Při registraci, aby zkontroloval, že nově registrovaný uživatel zadal svoji e-mailovou adresu (tento e-mail odesílal systém již v původní verzi).
- Při obnově hesla posílá kód k obnovení hesla. V e-mailu je odkaz na stránku k dokončení obnovy hesla.
- Když cvičící oznámkuje studentovo řešení, studentovi je o tom poslána informace e-mailem spolu se slovním komentářem cvičícího.
- Když cvičící zadá nový úkol, je o tom poslána informace studentovi. V e-mailu je odkaz na stránku s podrobným zadáním úkolu.
- Když student požádá o předčasné oznámkování úkolu, jeho cvičícímu je o tom poslán e-mail. V e-mailu je odkaz na stránku, kde může cvičící úkol stáhnout a ohodnotit.

U posledních třech případů může uživatel zakázat posílání e-mailu.

Aby bylo možné zobrazovat v e-mailech odkazy na stránky, bylo nutné provést několik úprav: zaprvé musí administrátor při instalaci systému do konfiguračního souboru napsat, na jaké webové adrese je systém přístupný, protože tuto informaci nemůže zjistit systém sám; a zadruhé, když uživatel klikne v e-mailu na odkaz do zabezpečené stránky na webu, ačkoliv aktuálně není do systému přihlášen, je místo toho navrácen na přihlašovací obrazovku. Dříve se takto ztratila informace o tom, na kterou stránku chtěl přejít, nyní si ovšem systém v adresním řádku zapamatuje cílovou stránku a po přihlášení uživatele na tuto stránku nasměruje.

Šablony textů e-mailů jsou modifikovatelné a jsou uloženy ve složce *documents*. Nejsou však lokalizovatelné a jsou tedy pouze v jednom jazyce. To je na překážku vícejazyčnosti, ovšem je třeba si uvědomit, že e-maily jsou posílány i uživatelům, kteří právě nejsou přihlášení, a systém tedy neví, jakým jazykem jim má e-mail poslat (protože jazykové nastavení se ukládá jako cookie na uživatelské počítači). Možné řešení tohoto problému je popsáno v sekci 7.2.1.

3.2.5 Slovní ohodnocení řešení

Jednou z nejžádanějších funkcí byla možnost přidat slovní ohodnocení řešení. Cvičící, když zadává do systému počet bodů, které chce studentovi za řešení udělit, může zároveň připsat i slovní poznámku. Text, který takto napíše, se studentovi pošle e-mailem a navíc se studentovi zobrazí v uživatelském rozhraní.

Okamžitě po zavedení této funkcionality ji cvičící začali využívat, hlavně k tomu, aby studentovi vysvětlili, z jakého důvodu mu dali menší počet bodů, než je maximum.

3.2.6 Uvolnění nároků na adresářovou strukturu řešení

Aby bylo jasné, jak systém funguje, a kvůli jednoduchosti implementace byly na řešení studentů kladeny některé formální nároky, které způsobovaly, že studenti často museli svoje řešení posílat vícekrát, než jim ho systém uznal jako správné. Tyto nároky totiž nebyly intuitivní.

Jednalo se o tyto nároky:

- Všechny soubory musí být zabaleny přímo v archivu ZIP. Nestačí, aby byly ve složce, která je celá zabalená v archivu. To je problém, protože studenti svoje řešení archivovali někdy tak, že archivačnímu programu poslali celou složku, v níž měli své řešení.
- Soubory musí mít přesně daný název, konkrétně *data.xml*, *data.dtd*, *data.xsd*, *data2html.xsl*, *data2xml.xsl*, *data.xsl*, *dom/user/MyDomTransformer.java*, *sax/user/MySaxHandler.java*, *xpath/xpath1.xp* až *xpath/xpath5.xp* a *xquery/query1.xq* až *xquery/query5.xq*. Studenti ovšem občas posílali soubory s jinými názvy jako třeba *knihovna.xml* nebo *schema.xsd*.

Většina těchto požadavků byla odstraněna. Pokud student pošle jednu složku zabalenou v archivu (a žádné další soubory), systém rozbalí vnitřek této složky. Soubory nyní rozpoznává pouze podle přípony místo celého názvu, takže soubory *knihovna.xml* a *schema.xsd* dohromady budou fungovat jako řešení čtvrtého domácího úkolu.

Na dvou místech se ale eliminovat požadavky nepodařilo. V úkolu číslo 5 (XQuery) musí mít stále jeden soubor jméno *data.xml*, a to proto, že XQuery

vyžaduje jeden soubor jako úvodní kontext. V ostatních pluginech jsou podobné požadavky řešeny tak, že se vybere jediný soubor s koncovkou *xml*, který v řešení je. Ovšem součástí zadání XQuery je i spojování dvou XML souborů, tedy v řešení nutně budou alespoň dva soubory s koncovkou *xml*, jménem lze tedy určit ten, který bude sloužit jako prvotní kontext.

V úkolu 2 (DOM/SAX) pak byla požadovaná adresářová struktura ponechána, protože opět je třeba odlišit oba soubory s koncovkou *JAVA*, a vynucovaná struktura přiměřeně odpovídá tomu, jak se v Javě používají balíčky.

3.2.7 Refaktorizace přístupu k databázi

V původní verzi systém XML Check využíval vlastní ORM framework od Konopáska. Tento framework měl zajistit snadnou rozšiřitelnost, např. umožnit snadno změnit databázový server. Také měl zjednodušit kontrolu vstupních dat a ochránit tak databázi a měl také zabránit některým programátorským chybám.

Framework je poměrně rozsáhlý. Konopásek sám ve své práci uznává, že ho vytvoření frameworku stálo mnoho úsilí, které se nakonec nevyplatilo. Framework nepodporuje některé pokročilejší funkce, proto je většina dotazů tvořena nikoliv přes tento framework, nýbrž přes databázové pohledy. Udržovat systém XML Check při použití tohoto frameworku by bylo hodně namáhavé, jak píše Konopásek[1]:

„Ačkoliv modularita umožňuje provádět některé podstatné změny, např. změnu databázového serveru, bez větších potíží, způsobuje, že všechny menší změny jsou mnohem obtížnější. Přidání nové funkce téměř vždy vyžaduje změny na několika oddělených úrovních a rozšíření rozhraní mezi nimi. Například umožnění slovního hodnocení řešení by vyžadovalo rozšíření databázové struktury, aplikování této změny do abstraktní databázové struktury v PHP kódu v jádře, změnu nebo přidání databázového požadavku, změnu nebo přidání požadavku na jádro a rozšíření uživatelského rozhraní, aby tuto změnu podporovalo.“ *(přeložil autor této práce)*

Z tohoto důvodu byl původní framework pro přístup k databázi úplně odstraněn a nahrazen použitím open-source frameworku Doctrine.

To s sebou přináší některé nevýhody. Zaprvé je velmi pracné přepracovat celou PHP část zdrojového kódu, aby používala Doctrine místo dosud zavedeného frameworku. Zadruhé je nyní projekt závislý na poměrně velké knihovně třetí strany, což zvyšuje jeho složitost.

Výhody změny ovšem za to stojí. Použijeme-li příklad uváděný Konopáskem, tak nyní, s použitím Doctrine, by zavedení funkcionality slovního hodnocení řešení obnášelo změnu databázové struktury v PHP kódu, spuštění příkazu `doctrine orm:schema-tool:update -force`, změnu požadavku na jádro a rozšíření uživatelského rozhraní. Framework Doctrine navíc umožňuje použít automatické doplňování kódu v integrovaných vývojových prostředích, které jsou tak nyní navíc schopné odhalit některé chyby v kódu. Přidávat nebo měnit funkcionality související s databází je nyní jednodušší.

3.2.8 Objekty nelze smazat, pouze schovat

Uživatel má právo mazat svoje řešení, administrátor může mazat uživatele, přednášející může mazat své přednášky, cvičící skupiny a úkoly a podobně. Dříve, kdykoliv byl nějaký objekt smazán, byl úplně odstraněn z databáze. Všechny na něm závislé objekty byly také odstraněny. Tedy například, pokud byla odstraněna skupina, byly odstraněny i všechny její úkoly.

Tím ovšem bylo způsobeno, že v systému je nyní archiv zhruba 2500 řešení, která nejsou přiřazena k úkolům, navíc řešení, která studenti neodevzdali k oznámkování, nýbrž smazali, jsou nadobro pryč. Tato řešení se hodí ke dvěma účelům: zaprvé ke kontrole správnosti kontrolujících pluginů (pokud se plugin neshoduje s ohodnocením cvičícího, možná je v pluginu chyba), a zadruhé je pro kontrolu podobnosti třeba uchovávat domácí úkoly jiných studentů, kterým by úkoly mohly být podobné. Nestačí přitom uchovávat jen samotné soubory řešení, nýbrž i metadata určující, který student řešení odevzdal pro jaký úkol.

Z tohoto důvodu nyní pomocí uživatelského rozhraní nelze objekty mazat, nýbrž pouze skrývat. Uživatelé navenek neuvidí rozdíl, jen objekty v databázi zůstanou s příznakem *deleted* a nebudou zobrazovány.

Bylo třeba také vyřešit situaci stávajících řešení, která nebyla k úkolům přiřazena, protože tyto úkoly již byly smazány. Bylo tedy vytvořeno šest virtuálních

úkolů, jeden úkol pro každý problém, a k těmto úkolům byla řešení přiřazena poloautomatickým způsobem podle jejich adresářové struktury.

Zde bylo s výhodou využito rigidní adresářové struktury, kterou vynucovala původní verze systému. Tak tedy např. řešení, které obsahovalo jen soubory *data.xml* a *data.xsd* bylo zjevně řešením domácího úkolu 4 (XSD), a řešení, které obsahovalo soubor *data.xml* a složky *dom* a *sax* bylo zjevně řešením domácího úkolu 2 (DOM/SAX). Některá řešení se ovšem nedala automaticky zařadit a musela by být zkontrolována ručně. Jedná se ovšem zhruba o sto řešení, a u většiny se nejedná o správné řešení (jde například o řešení, která obsahují jediný soubor *test.txt*), tato zbývající řešení tedy nebudou použita.

3.2.9 Algoritmus pro hashování hesel

V původní verzi byla hesla uživatelů chráněna v databázi pouze hashovacím algoritmem MD5 bez použití soli.⁴ Hashování hesel je u webových aplikací běžný postup, kdy je na hesla od uživatelů uplatněna jednosměrná hashovací funkce. Jejímu výstupu říkáme *hash* a ten se ukládá do databází místo hesla. Při přihlašování uživatele se pak použije hashovací funkce na heslo zadané uživatelem a její výstup se porovná s *hashem* v databázi. Uživatel je přihlášen, pokud se oba rovnají.

Pokud by do databáze získal přístup neoprávněný uživatel, a hesla nebyla hashována, mohl by zjistit, jaké heslo má jaký uživatel, a poté se přihlašovat jejich jménem. Kromě toho je pravděpodobné, že někteří uživatelé používají v systému XML Check stejné heslo jako v jiných aplikacích, a proto by se takový útočník mohl jejich jménem přihlásit i do těchto dalších aplikací.

Hashovací algoritmus MD5 je však velmi rychlý a existuje množství předpřipravených tabulek, pomocí nichž lze získat z hashe zpět pravděpodobné původní heslo. Kdyby se útočník dostal k databázi systému, nejspíše by velmi rychle dokázal získat hesla všech jeho uživatelů. [15]

⁴Algoritmus MD5 býval dříve kvůli své jednoduchosti používán na mnoha webových stránkách. Jeho velkou nevýhodou ovšem je, že je příliš efektivní, a útočník tedy může vyzkoušet velké množství hesel velmi rychle. Hashování s použitím soli znamená, že se vygeneruje náhodný řetězec znaků („sůl“), který je připojen na konec hesla a hashován je až výsledný řetězec. Sůl je poté uložena spolu s hashem. Tímto lze útočnickovi zabránit v použití předpočítaných tabulek a donutit ho prolamovat každé heslo samostatně. I při použití soli ovšem zůstává takový systém zranitelný. Více informací lze najít v manuálu PHP (<http://php.net/manual/en/faq.passwords.php>).

Z tohoto důvodu byla do systému přidána knihovna *PHPass*, která implementuje bezpečnější hashovací algoritmus. Z hashů generovaných tímto algoritmem již není snadné získat zpět původní heslo. Všichni nově založení uživatelé od této změny mají hesla hashována touto knihovnou.

Bylo několik možností, jak naložit s hesly stávajících uživatelů:

- Vygenerovat jim náhodné heslo a poslat jim ho na e-mail. To je pro uživatele velmi nepříjemné, také vzhledem k tomu, že velké množství uživatelů již systém XML Check nepoužívá. Je také možné, že někteří z nich použili při registraci e-mail, který pravidelně nepoužívají, a e-mail s novým heslem by jim tedy nepřišel.
- Použít tabulky ke zjištění hesel všech uživatelů, aplikovat na ně novou hashovací funkci a znovu uložit do databáze. Takto by uživatelé vůbec nemuseli o změně vědět, a neměli by žádnou práci se změnou hesla. Je ale možné, že některý uživatel má tak složité heslo, že by se ho pomocí tabulek nepodařilo zjistit. Navíc takové zjišťování hesel není etické ani legální.
- Ponechat uživatelům jejich hesla v MD5 hashi, při změně hesla bude jejich nové heslo zašifrované pomocí PHPass; upozornit uživatele, že by si měli změnit heslo. To ovšem vyžaduje uchovávat si v tabulce v databázi informaci o tom, jakým způsobem je heslo hashováno. Tato možnost byla vybrána.

Ze 108 uživatelů, kteří systém v letním semestru 2014 používali, svoje heslo převedlo na bezpečnější algoritmus jen 21 uživatelů, ačkoliv upozornění na vhodnost změny hesla bylo vyvěšeno delší dobu na úvodní stránce systému. Nezájem uživatelů o změnu hesla je nejspíše zčásti dán tím, že pro systém XML Check zvolili málo významné heslo, jako např. *12345*, a nevdá jim příliš, pokud bude jejich heslo zjištěno.

Celkem 591 uživatelů má nyní v systému heslo chráněné jen slabým hashovacím algoritmem. Celkem 112 z těchto uživatelů sdílí své heslo alespoň s jedním dalším uživatelem, dvě hesla jsou dokonce používána každé 5 různými uživateli. Je možné uvažovat o smazání hesel všech uživatelů, aby se zajistila jejich větší bezpečnost, o tomto diskutujeme více v závěru práce.

3.2.10 Zapomenuté heslo

Na konci své bakalářské práce Konopásek uvádí nemožnost obnovy ztraceného nebo zapomenutého hesla jako jeden z nedostatků vytvořeného systému. Nyní může uživatel zadat svůj e-mail do formuláře pro obnovu hesla. Tím se jeho heslo nezmění, ovšem dostane e-mail, ve kterém bude odkaz na nový formulář, kde si může nastavit nové heslo bez toho, aby musel zadat své předchozí.

Takto je zaručené, že pouze vlastník e-mailu asociovaného s uživatelským účtem může změnit heslo.

3.3 Kontrola podobnosti domácích úkolů

Třetí částí této práce je implementace kontroly podobnosti domácích úkolů. Tato kontrola proběhne při vložení každého nového řešení a byla také spuštěna na všech řešeních odevzdaných do roku 2014.

Kontrolou podobnosti chceme bránit studentům v odevzdávání opsaných úkolů, a tedy studenty donutit, aby domácí úkoly skutečně vyřešili sami v plné míře, a osvojili si tak učební látku. Studentům, kteří dodají opsané řešení, pak může být například poslán výstražný e-mail nebo mohou být požádáni o dodání vlastního řešení.

Kontrola podobnosti se může zdát v tomto předmětu zbytečná, protože opisování je velmi obtížné. Na začátku semestru si každý student zvolí téma a cvičící přitom dbá na to, aby stejné téma nezvolili dva studenti. Již z toho by mělo vyplývat, že jednoduché opisování není možné a že žádné plagiáty odhaleny nebudou. Předpokládali jsme tedy, že naimplementovaná kontrola podobnosti neodhalí žádné podvody.

Ukázalo se ovšem, že někteří studenti přesto podváděli. V opsaném textu například změnili identifikátory (např. z popisu knihovny jen změnou názvu vytvořili popis firmy) nebo, v několika případech, dokonce použili téměř stejný soubor XML.

V této kapitole nejprve popíšeme metodologii kontroly, poté uvedeme výsledky spuštění různých algoritmů na odevzdaná data, a nakonec uvedeme, jak bude kontrola podobnosti fungovat v systému XML Check do budoucna.

Budeme používat tuto terminologii:

- **Podezřelé řešení** je řešení, které systém označil jako opsané.
- **Správně podezřelé řešení** je podezřelé řešení, které student skutečně od někoho opsal.
- **Falešný poplach** je řešení, které systém označil jako opsané, přestože opsané nebylo.

3.3.1 Metodologie

Pro každý použitý algoritmus nebo jeho konfiguraci jsme procházeli nalezené páry řešení podle míry podobnosti sestupně, dokud jsme nezačali narážet na řešení označená za podezřelá, ač po prozkoumání podezřelá nebyla (falešné poplachy).

Ve všech testech jsme samozřejmě ignorovali podobnosti mezi různými řešeními téhož autora (student má právo nahrát řešení vícekrát).

Pro analýzu bylo důležité zjistit, zda řešení, která byla automaticky označena za podezřelá, skutečně studenti od sebe opsali. Toto pozorování provedl autor této práce, přičemž se snažil jej provádět velmi konzervativně: pokud byla rozumná šance, že řešení nebyla od sebe opsána, označil autor určení systému za chybné.

3.3.2 Seznam provedených testů

Provedeny byly po řadě tyto testy:

1. Identita; pouze řešení 1. domácího úkolu
2. Levenshteinova vzdálenost, pokus 1; pouze řešení 1. domácího úkolu, velikost dokumentů omezena na 2 kB bez mezer, pouze dokumenty XML
3. Levenshteinova vzdálenost, pokus 2; pouze řešení 1. domácího úkolu, velikost dokumentů omezena na 2 kB bez mezer, kontrolovány také dokumenty DTD
4. Levenshteinova vzdálenost, pokus 3; pouze řešení 1. domácího úkolu, velikost dokumentů omezena na 10 kB bez mezer, důležitost dokumentu DTD snížena
5. Levenshteinova vzdálenost, pokus 4; všechna nesmazaná řešení, malé soubory ignorovány, velikost dokumentů omezena na 200 kB bez mezer

6. Levenshteinova vzdálenost, pokus 5; jako předtím, ovšem ze souborů jsou smazány části šablon pro DOM a SAX
7. Greedy-String-Tiling, pokus 1; pouze řešení 1. domácího úkolu, velikost dokumentů omezena na 2 kB; MinimumMatchLength nastaveno na 2
8. Greedy-String-Tiling, pokus 2; pouze řešení 1. domácího úkolu, velikost dokumentů omezena na 2 kB; MinimumMatchLength nastaveno na 8
9. Zhang-Shasha, pokus 1; všechna nesmazaná řešení, žádné omezení velikosti, nastavení vah: $INSERT=1$, $DELETE=1$, $RELABEL=1$, pouze dokumenty XML
10. Zhang-Shasha, pokus 2; pouze řešení 1. domácího úkolu, žádné omezení velikosti, nastavení vah: $INSERT=1$, $DELETE=1$, $RELABEL=0$, pouze dokumenty XML

3.3.3 Test: Identita

V tomto testu byla odhalena a odstraněna některá testovací řešení administrátorů. Jeden student také předmět poprvé neabsolvoval úspěšně a pro následující rok si založil nový uživatelský účet, na kterém nahrál stejné řešení.

3.3.4 Test: Levenshteinova vzdálenost

Levenshteinova vzdálenost se ukazuje být dobrým měřítkem podobnosti. Ač se jedná o poměrně jednoduchý algoritmus, našel mnoho správně podezřelých řešení, a přitom nevykazoval vysokou míru falešných poplachů.

První pokusy (Levenshtein-1 až Levenshtein-3) operovaly nad omezeným množstvím dat. Podařilo se v nich zachytit některé páry, které v pozdějších pokusech (Levenshtein-4 a Levenshtein-5) již byly pod prahem podezřelosti. Byl tak objeven pár dvou řešení s podobností 42% u XML souboru a 56% u DTD souboru, kde se jednalo o opisování. Na druhou stranu na této hladině podobnosti byly nalezeny i páry, které opsané nebyly a u nichž byla podobnost jen náhodná.

Samozřejmě tyto soubory přesto vypadaly podobně, ale je třeba si uvědomit, že systém vybral z 90 000 párů těch pár desítek, které si byly nejpodobnější.

Teprve testy na téměř všech datech odhalily velké množství podobností. V testu

Levenshtein-4 se objevilo velké množství řešení, která si byla podobná jen tím, že používala stejnou šablonu zdrojového kódu Javy jako základ. Proto jsme tyto společné části odfiltrovali a v testu Levenshtein-5 již tato řešení podezřívána nebyla.

Tento druh testování se ukazuje stejně dobrým jako testování vzdáleností Zhang-Shasha. Výsledky testování a srovnání těchto dvou algoritmů uvádíme dále (sekce 3.3.7).

3.3.5 Test: Greedy-String-Tiling

Algoritmus Greedy-String-Tiling byl spuštěn ve dvou variantách, poprvé s nastavením $MML = 2$ (varianta 1) a podruhé s $MML = 8$ (varianta 2).

V omezeném vzorku, na kterém byl algoritmus spuštěn, objevila varianta 1 celkem 5 správně podezřelých řešení, než vrátila falešný poplach na úrovni podobnosti 66%. Varianta 2 na témže vzorku objevila celkem 6 správně podezřelých řešení, než našla první falešný poplach na úrovni podobnosti 30%. Obě tyto úrovně jsou velmi nízké a je docela možné, že by tato metoda mohla být pro kontrolu podobnosti velmi účinná.

Zkoušení na větším vzorku řešení však bránila její výpočetní náročnost. Časová složitost algoritmu je v průměrném i nejhorším případě $O(n^3)$, kde n je počet znaků delšího z porovnávaných řešení. Násobící konstanta je navíc poměrně velká.

Přestože jsme tuto metodu z důvodu její časové složitosti zavrhnuli a systém ji nyní nepoužívá, autor původního algoritmu v [16] také navrhuje několik optimalizací. Nejzásadnější z nich je kombinace s variantou Robin-Karpova algoritmu, která sice časovou složitost v nejhorším případě nezlepší, ale autor tvrdí, že v praxi se pak rychlost blíží lineární.

Implementace této optimalizace může být námětem pro další práci.

3.3.6 Test: Editační vzdálenost Zhang-Shasha

Ač se tato metoda ze začátku zdála velmi vhodná pro použití na XML datech, v praxi dosahuje velmi podobných výsledků jako prostá Levenshteinova vzdálenost.

Samozřejmě, jako u všech metod, může záležet na nastavení parametrů. Nezkoušeli jsme všechna možná nastavení. Stálo by za zkoušku především ohodnocení

vah *INSERT*, *DELETE* i *RELABEL* Levenshteinovou vzdáleností mezi jmény obou uzlů (v případě *INSERT* a *DELETE* délkou daného uzlu).

Zjistili jsme, že rozhodně nefunguje nastavení váhy *RELABEL* na 0. Porovnávání struktury bez ohledu na jména uzlů produkuje příliš velké množství falešných poplachů. Při pokusu byl první falešný poplach nalezen již na úrovni podobnosti 95%.

Zajímavé je ovšem nastavení $INSERT=DELETE=RELABEL=1$, které se ukazuje být právě tak dobrým jako Levenshteinova vzdálenost. Srovnání obou algoritmů uvádíme v následující sekci.

Zjevnou nevýhodou tohoto testu ovšem je, že funguje pouze na správně zformovaných XML datech. Nedokáže tedy odhalit podobnosti ve zdrojových kódech Javy, a dokonce ani v nesprávně zformovaných souborech XML. Použití této metody tedy stejně vynucuje použití jiné metody pro ostatní druhy souborů.

3.3.7 Zhodnocení

Zdroje správně podezřelých řešení můžeme roztrždit do těchto kategorií:

1. Student opsal řešení od jiného studenta. Přitom udělal v souborech malé změny, aby opisování zamaskoval.
2. Student je součástí většího okruhu studentů, kteří všichni použili stejný soubor XML.
3. Student opakoval předmět a založil si kvůli tomu nový účet.
4. Student si založil dva uživatelské účty, ač je nepotřeboval (možná proto, že zapomněl své heslo? – v té době systém ještě neměl funkci obnovy hesla).
5. Administrátor (cvičící) prováděl pokusy a nahrával studentovo řešení ze svého účtu.

Ačkoliv nás zajímají pouze první dva body z tohoto seznamu, všechny tyto příčiny musíme považovat za správně podezřelá řešení. Automatický systém prostě nemůže rozlišit, že dva uživatelské účty založila stejná fyzická osoba.

Ve většině „úmyslných podvodů“ (správně podezřelých řešení podle prvního bodu výše uvedeného seznamu) student maskoval svoje opisování tak, že řešení přepsal do jiného jazyka (např. ze slovenštiny do angličtiny) nebo změnil téma (např. v jednom případě změnil popis pekárny na popis univerzity).

Co se týče druhého bodu, v datech jsou dva shluky řešení (jeden jsme nazvali „knihovna“ a druhý „půjčovna aut“), kde přibližně deset studentů odevzdalo stejný soubor XML. Soubory týkající se technologie, jejíž znalost byla právě testována (např. XPath), ovšem měli studenti odlišné. Nepodařilo se nám zjistit, jak tyto shluky vznikly.

Zdroje falešných poplachů byly hlavně:

- Příliš krátký soubor, který může způsobit vysokou podobnost čistou náhodou.
- Student nahrál nejen soubor XML s řešením, ale celý projekt NetBeans, ve kterém bylo mnoho XML konfiguračních souborů. Protože toto učinilo více studentů, byla jejich řešení vyhodnocena jako téměř identická, protože se shodovaly jejich konfigurační soubory.

S těmito problémy jsme se vypořádali tímto způsobem:

- Soubory, které po odstranění bílých znaků jsou menší než 500 bajtů, nejsou porovnávány na podobnost. Také byla zakázána kontrola souborů DTD, XPath a XQuery, které jsou příliš malé vždy.
- Studentům bylo zakázáno nahrávat celé projekty NetBeans a Eclipse.

Algoritmus Levenshteinovy vzdálenosti a algoritmus editační vzdálenosti Zhang-Shasha vykazovaly velmi podobné výsledky, které jsou shrnuty v tabulce 1. Podrobnější data jsou na přiloženém disku (viz příloha A).

Zdánlivý rozpor mezi čísly 218 (počet správně podezřelých párů) a 17 (počet úmyslných podvodů) se vysvětlí tím, že většina z těchto 218 párů se týkala dvou zmíněných shluků řešení – půjčovna aut a knihovna.

Algoritmus Zhang-Shasha dva falešné popluchy nahlásil již na úrovních 75% a 74%, zatímco Levenshteinův test má první poplach na úrovni 60%, a zde se navíc jedná o malý soubor XSD v úkolu, který se schémata nesouvisí. Počty nahlášených

	Zhang-Shasha	Levenshtein
Počet správně podezřelých párů před prvním falešným poplachem	218	220
Úroveň podobnosti u prvního falešného poplachu	75%	60%
Počet nalezených úmyslných podvodů před prvním falešným poplachem	17	13
Počet úmyslných podvodů s podobností nad 90%	8	5
Počet úmyslných podvodů s podobností nad 80%	14	7
Celkem porovnávaných řešení	2592	

Tabulka 1: Výsledky analýzy podobnosti nad odevzdanými úkoly

podvodů jsou u obou algoritmů podobné. Algoritmus Zhang-Shasha sice nahlásil podvodů víc, ale většinu až blízko úrovni prvního falešného poplachu.

Na vývojovém počítači⁵ trvá porovnání jednoho nového studentského řešení se všemi předcházejícími přibližně 50 sekund, a kompletní analýza všech řešení 5 hodin.

3.3.8 Kontrola podobnosti do budoucna

Implementační detaily jsou popsány v programátorské dokumentaci. Stručně řečeno, kontrola podobnosti běží v procesu Javy na pozadí. Spouští se, kdykoliv student nahraje nové řešení. Cvičící má k proběhlým kontrolám podobnosti přístup a může si je zobrazit v uživatelském rozhraní, kde si může příslušná řešení také stáhnout.

Používá se jak algoritmus Zhang-Shasha, tak algoritmus Levenshteinovy vzdálenosti. Pokud podobnost zjištěná prvním algoritmem přesáhne uživatelem zadaný práh, je řešení označeno za podezřelé rovnou, jinak se spustí ještě Levenshteinův test. Ten se také spustí rovnou, pokud se nejedná o soubor XML nebo soubor není správně zformovaný.

Vzhledem k tomu, že absence falešných poplachů je pro nás důležitější než nalezení všech podvodníků, nastavili jsme výchozí hodnotu prahu podezřelosti na 90% pro vzdálenost Zhang-Shasha a 80% pro Levenshteinovu vzdálenost.

⁵Intel Core2 Duo E8500, 2x3.16 GHz, 8.00 GB RAM, Java 8, doporučená nastavení porovnávání podle textu této práce

4. Časová osa projektu

Předmět Technologie XML se na MFF vyučuje od roku 2005, doc. RNDr. Irena Holubová Ph.D. ho samostatně vyučuje od roku 2008. Domácí úkoly však svoji podobu od roku 2005 téměř nezměnily.

System XML Check v původní verzi (tehdy ještě nazvaný *Assignment Manager*) byl poprvé nasazen Janem Konopáskem v létě roku 2011. Během tohoto prvního semestru používání byl systém upravován. Byly odstraňovány chyby a byla přidána nová funkcionalita, kterou si vyžádali uživatelé při používání systému, např. možnost cvičícího změnit ohodnocení řešení, které již ohodnotil. Změny, které byly v tomto semestru provedeny, popisuje Jan Konopásek ve své práci.

Poté, co Jan Konopásek v zimě 2011/2012 obhájil svoji bakalářskou práci, nebyl systém dále rozšiřován nebo opravován. Jediná změna, která byla provedena, je změna popisu pluginu pro 2. domácí úkol (do popisu byla přidána šablona zdrojových souborů v Javě). Systém se totiž nedá upravovat dostatečně snadno, aby vzniklé chyby mohli cvičící nebo přednášející řešit bez toho, že by věnovali velké množství času seznámením se s implementací. Místo toho tedy spíše sepisovali nedostatky nalezené během těchto let, aby mohly být opraveny později.

Autor této práce začal systém upravovat během letního semestru roku 2013/2014. V tuto dobu se zabýval převážně opravou chyb a přidáváním funkcionality. Během semestru byl systém aktivně používán, což přispělo k nalezení mnoha chyb, zároveň to však znatelně ztížilo jeho opravy. Před každým uvedením nové verze bylo třeba systém překontrolovat, zda všechno funguje správně. Přesto bylo možné, že by se zavedená chyba dostala do reálného provozu, což se také skutečně stalo v jednom případě, kdy zavlečená chyba způsobila, že systém ztratil schopnost posílat e-maily.

Z tohoto důvodu bylo rozhodnuto, že nové verze budou nahrávány jen po předchozí záloze databáze, a to jen ve dnech, kdy není termín odevzdání žádného domácího úkolu. Tato opatrnost se vyplatila.

Když skončil letní semestr, byly ještě prováděny další opravy a přidávány nové funkce a také byl kód refaktorován. Teprve v prosinci 2014 byl implementován podsystém na kontrolu podobnosti domácích úkolů. Před začátkem další výuky byl systém důkladně otestován, ručně i automaticky.

V letním semestru 2015 byl plně upravený systém znovu nasazen. Studenti a vyučující během tohoto semestru objevili několik dalších chyb (převážně regresí), všechny ale byly rychle opraveny. V době odevzdání této práce (jaro 2015) je systém plně funkční a je v plánu pokračovat v jeho používání i v dalších letech.

5. Uživatelská dokumentace

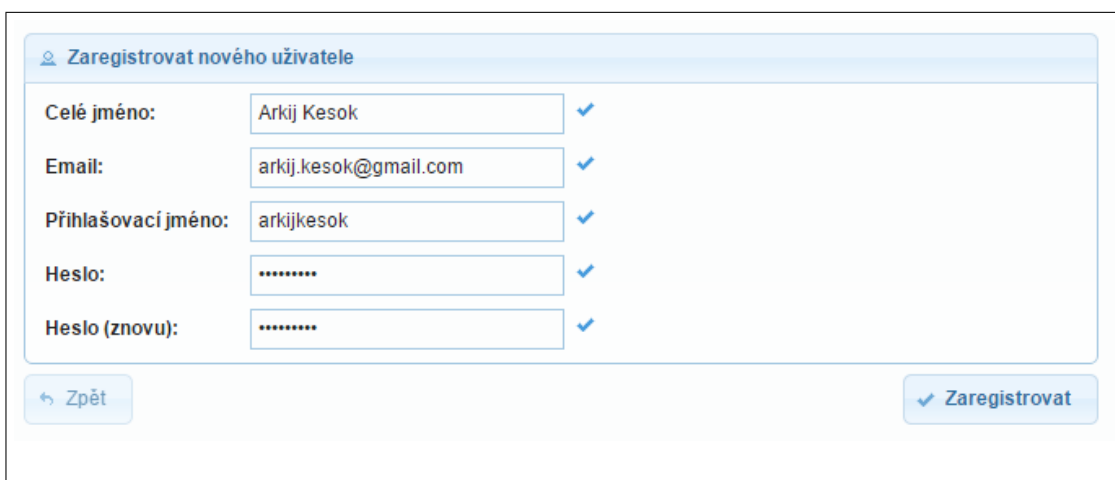
Poznámka: Tato uživatelská dokumentace je k dispozici také ve formě dokumentu PDF, který se nachází na přiloženém disku (příloha A). Informace zde a v dokumentu PDF jsou shodné. Tato uživatelská dokumentace je v češtině – starší dokumentaci v angličtině najdete v [1].

XML Check je systém pro správu a poloautomatické hodnocení domácích úkolů využívaný v předmětu Technologie XML. Jeho první verze byla nasazena v roce 2011. Dříve byl systém XML Check někdy také nazýván *Assignment Manager*.

5.1 Běžné úkony

5.1.1 Registrovat se

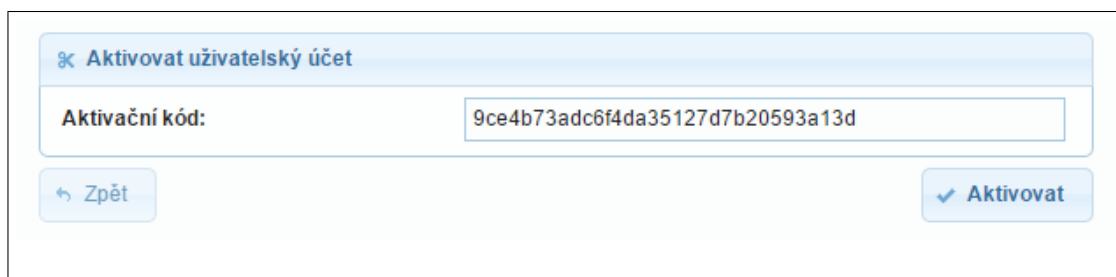
Na úvodní stránce klikněte na tlačítko **Registrace** a poté vyplňte registrační formulář (obrázek 3). Minimální délka uživatelského jména a hesla je 5 znaků, všechny ostatní údaje mohou mít libovolný počet znaků. Celé jméno musí být vaše skutečné občanské jméno a musí obsahovat jen písmena české abecedy a mezery. Systém vás upozorní, pokud vaše přihlašovací jméno již někdo používá.



Obrázek 3: Registrační formulář

Po vyplnění formuláře stiskněte **Zaregistrovat** a bude vám na zadanou adresu poslán e-mail s aktivačním kódem a odkazem na formulář k zadání aktivačního

kódu (obrázek 4). Na tento formulář se po stisknutí tlačítka **Zaregistrovat** také automaticky přesměruje webová stránka. Můžete k němu přistoupit také pomocí tlačítka **Aktivování** z přihlašovací obrazovky.

The image shows a web form titled "Aktivovat uživatelský účet" (Activate user account). It features a text input field labeled "Aktivační kód:" (Activation code) containing the alphanumeric string "9ce4b73adc6f4da35127d7b20593a13d". Below the input field are two buttons: "Zpět" (Back) on the left and "Aktivovat" (Activate) on the right.

Obrázek 4: Formulář pro zadání aktivačního kódu

Zkopírujte kód z e-mailu do jediného políčka na obrazovce a klikněte na tlačítko **Aktivovat**. Nyní jste zaregistrováni a můžete se přihlásit. Na přihlašovací obrazovku budete automaticky přesměrováni.

Hesla jsou hashována silným algoritmem a útočník, který by přistoupil k databázi, by je neměl dokázat získat.

Uživatelské účty v systému XML Check nejsou nijak spojeny se školními autentifikačními systémy jako je SIS nebo KOS.

Upozornění: Přihlašovací jméno nelze po registraci změnit.

5.1.2 Přihlásit se

Přihlašovací obrazovka se vám zobrazí, kdykoliv nejste přihlášení do aplikace (obrázek 5). Z formulářů pro registraci a pro obnovu hesla se dá do přihlašovací obrazovky dostat kliknutím na tlačítko **Zpět**.

The image shows a web form titled "Přihlašovací obrazovka" (Login screen). It contains two text input fields: "Uživatelské jméno:" (Username) with the value "arkijkesok" and "Heslo:" (Password) with masked characters ".....". At the bottom of the form are four buttons: "Registrace" (Registration), "Aktivování" (Activation), "Zapomenuté heslo?" (Forgot password?), and "Přihlásit se" (Login).

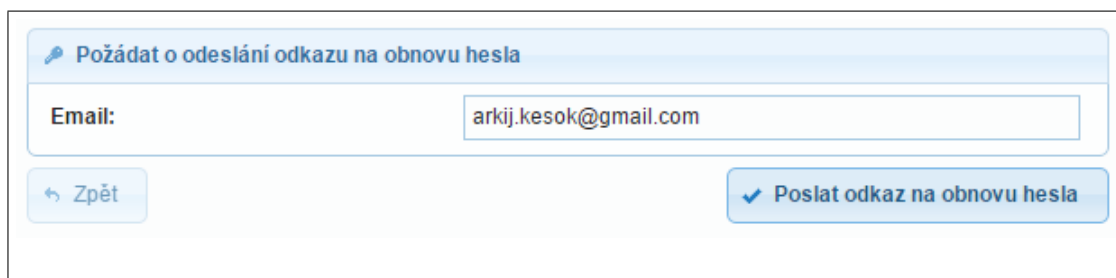
Obrázek 5: Přihlašovací obrazovka

Vaše uživatelské jméno je jménem, které jste při registraci uvedli jako přihlašovací jméno. Jméno i heslo musí mít alespoň 5 znaků, ale nejsou na ně kladena

žádná další omezení. Po zadání údajů klikněte na tlačítko **Přihlásit se**.

5.1.3 Obnovit heslo

Z přihlašovací obrazovky klikněte na tlačítko **Zapomenuté heslo?** a vyplňte e-mail, který jste zadali při registraci (obrázek 6). Pak klikněte na **Poslat odkaz na obnovu hesla**.

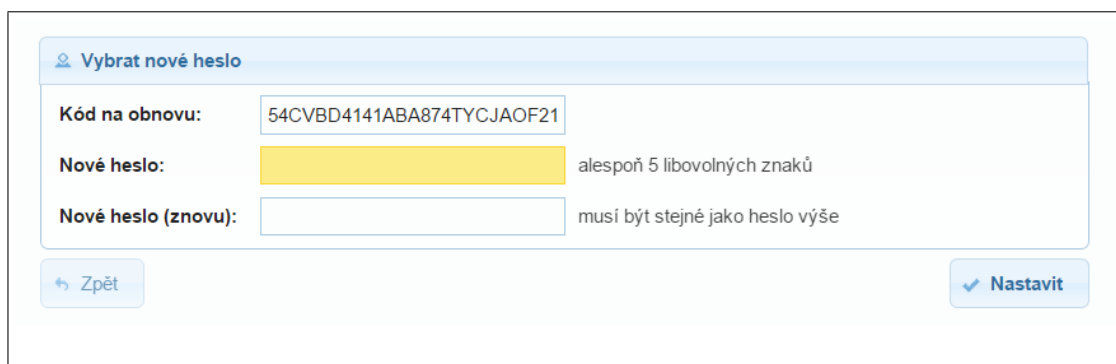


The screenshot shows a web form titled "Požádat o odeslání odkazu na obnovu hesla". It features an "Email:" label followed by a text input field containing "arkij.kesok@gmail.com". Below the input field are two buttons: "Zpět" (Back) on the left and "Poslat odkaz na obnovu hesla" (Send password reset link) on the right.

Obrázek 6: Žádost o obnovu hesla

Do vaší schránky přijde zpráva s odkazem, pomocí kterého můžete po následujících 24 hodin změnit své heslo. Poté bude odkaz zneplatněn.

Po kliknutí na tento odkaz (nebo jeho zkopírování do adresního řádku webového prohlížeče) můžete pro svůj účet určit nové heslo (obrázek 7). Heslo musí mít alespoň 5 znaků, jinak pro něj neplatí žádná omezení.



The screenshot shows a web form titled "Vybrat nové heslo". It contains three input fields: "Kód na obnovu:" with the value "54CVBD4141ABA874TYC.JAOF21", "Nové heslo:" which is highlighted in yellow and has a note "alespoň 5 libovolných znaků", and "Nové heslo (znovu):" with a note "musí být stejné jako heslo výše". At the bottom are two buttons: "Zpět" (Back) on the left and "Nastavit" (Set) on the right.

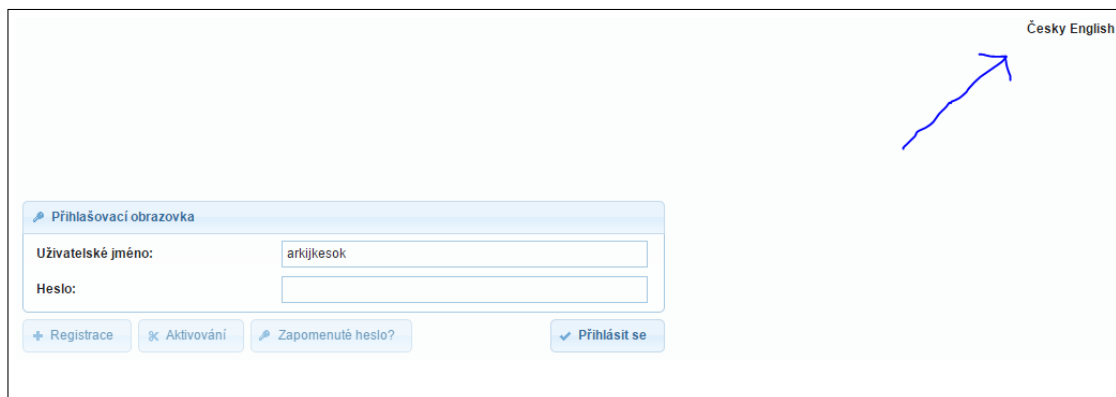
Obrázek 7: Formulář pro určení nového hesla

5.1.4 Změnit jazyk

System **XML Check** lze používat v češtině nebo v angličtině. Můžete nastavit, v jakém jazyce se vám má systém zobrazovat. Toto nastavení bude uloženo v souboru cookie na vašem počítači.

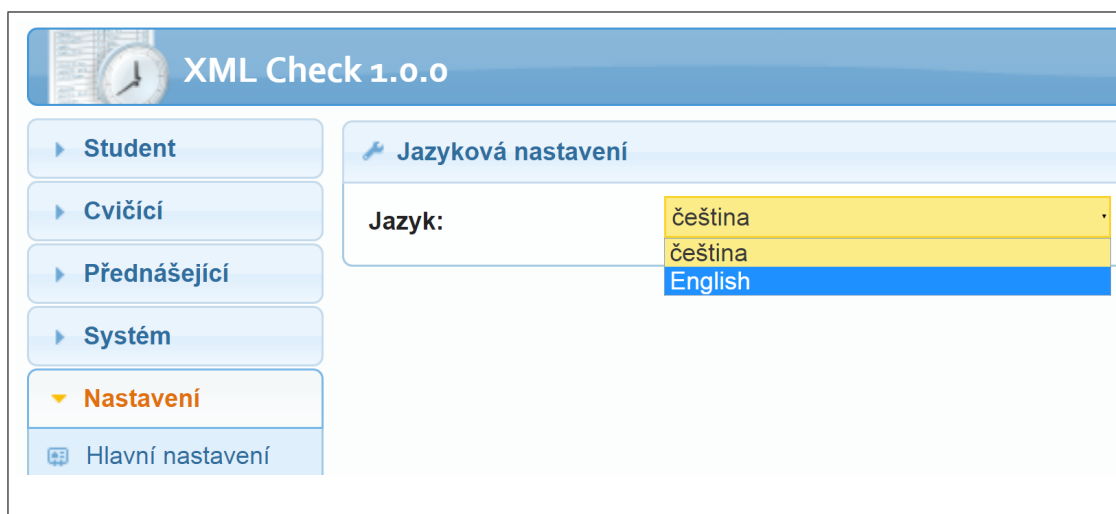
Změnit jazyk můžete na dvou místech:

1. Kliknutím na odkaz **Česky** nebo **English** u přihlašovacího formuláře v pravém horním rohu obrazovky. (obrázek 8)



Obrázek 8: Změna jazyka z přihlašovací obrazovky

2. Po přihlášení v rubrice **Nastavení** na stránce **Změnit jazyk** můžete vybrat jazyk z rozbalovacího seznamu. Poté klikněte na tlačítko **Potvrdit**. (obrázek 9)



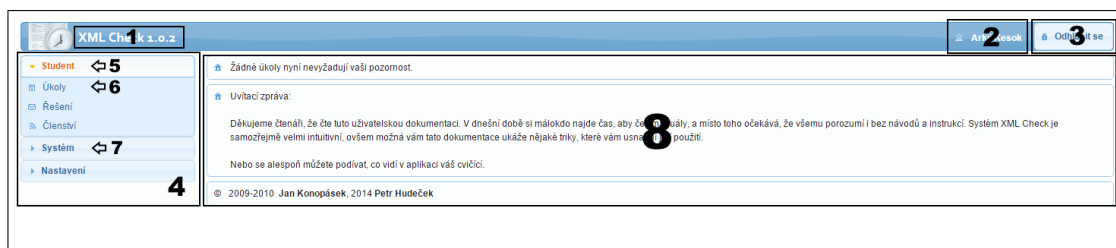
Obrázek 9: Změna jazyka po přihlášení

5.1.5 Obrazovka po přihlášení v aplikaci XML Check

Po přihlášení uvidíte obrazovku podobnou té na obrázku 10.

Na této obrazovce se nachází:

1. Jméno a verze systému XML Check (kliknutím sem se vrátíte na hlavní stránku)
2. Celé jméno právě přihlášeného uživatele



Obrázek 10: Hlavní obrazovka systému XML Check

3. Tlačítko pro odhlášení
4. Menu (menu je dělené do *rubrik*; každá rubrika obsahuje několik odkazů na *stránky*; zobrazují se vám pouze ty rubriky a stránky, ke kterým máte právo přistupovat)
5. Aktivní rubrika
6. Odkaz na stránku (kliknutím se přesunete na tuto stránku; pokud je tato stránka již vybraná, nestane se nic)
7. Neaktivní rubrika (kliknutím ji rozbalíte; zůstanete ovšem na stránce, kde právě jste)
8. Obsahové okno (zde se zobrazuje obsah stránky, na které se nacházíte)

5.1.6 Změnit uživatelská nastavení

V rubrice **Nastavení** je několik formulářů, kterými můžete změnit svá nastavení.

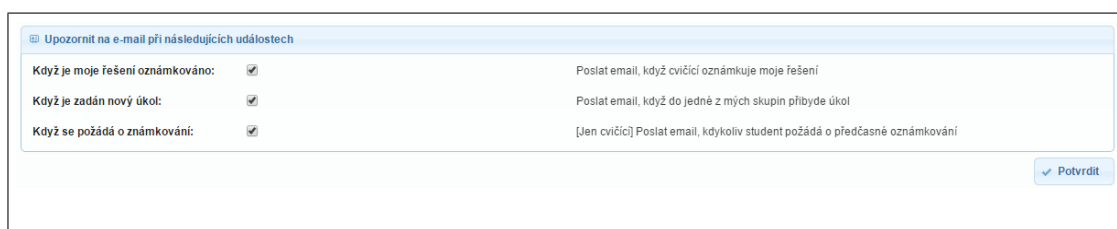
Na stránce **Hlavní nastavení** můžete změnit své celé jméno a e-mailovou adresu (obrázek 11). Možnost změny celého jména využijte jen tehdy, pokud jste se spletli při registraci. Pokud ve formuláři necháte položku **Nové heslo** prázdnou, zůstane vám heslo staré.

Obrázek 11: Hlavní nastavení

Na stránce **E-mailové notifikace** můžete zaškrtnout, při jakých příležitostech vám má být poslán e-mail (obrázek 12). Poté klikněte na tlačítko **Potvrdit**.

Jednotlivé možnosti jsou:

- **Když je moje řešení oznámkováno:** Bude vám poslán e-mail s vaším počtem bodů a komentářem cvičícího v momentě, kdy vám cvičící přiřadí počet bodů za vaše řešení. Automatické hodnocení nikdy neposílá e-maily. Pokud se cvičící splete při zadávání počtu bodů, může se opravit. V takovém případě vám může k jednomu domácímu úkolu přijít více e-mailů. Tyto e-maily se posílají jen studentům.
- **Když je zadán nový úkol:** Bude vám poslán e-mail s odkazem na zadání úkolu v momentě, kdy cvičící zadá vaší skupině nový domácí úkol. Někteří cvičící ovšem zadávají do systému všechny úkoly již na začátku semestru, v takovém případě vám buď přijde mnoho e-mailů současně (pokud se zaregistrujete předtím, než cvičící úkoly zadá), nebo vám neprijde žádný e-mail (pokud je cvičící zadal dříve, než jste se zaregistrovali). Tyto e-maily se posílají jen studentům.
- **Když se požádá o známkování:** Bude vám poslán e-mail s odkazem na studentovo řešení, kdykoliv student explicitně požádá o oznámkování svého řešení před termínem odevzdání. Tyto e-maily se posílají jen cvičícím.

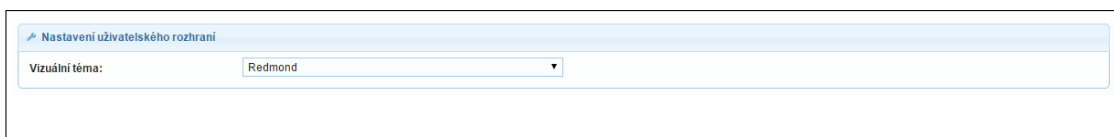


Upozornit na e-mail při následujících událostech	
Když je moje řešení oznámkováno:	<input checked="" type="checkbox"/> Poslat email, když cvičící oznámkuje moje řešení
Když je zadán nový úkol:	<input checked="" type="checkbox"/> Poslat email, když do jedné z mých skupin přibude úkol
Když se požádá o známkování:	<input checked="" type="checkbox"/> [Jen cvičící] Poslat email, kdykoliv student požádá o předčasné oznámkování

Obrázek 12: Nastavení e-mailových notifikací

Na stránce **Uživatelské rozhraní** můžete změnit vzhled systému (obrázek 13). Vzhled se změní okamžitě, jakmile vyberete téma v rozbalovacím seznamu. Pomocí kurzorových kláves *nahoru* a *dolů* můžete snadno procházet všechna témata, abyste rychleji vybrali to, které se vám líbí nejvíce. Tento formulář nemá žádné potvrzovací tlačítko, změna tématu je provedena okamžitě a je uložena jako soubor cookie na vašem počítači.

Na stránce **Změnit jazyk** můžete změnit jazyk systému, viz sekce 5.1.4.



Obrázek 13: Nastavení uživatelského rozhraní

5.1.7 Práce s tabulkami v systému XML Check

Mnoho informací v systému XML Check je reprezentováno tabulkami. Obzvláště cvičícím a přednášejícím, kteří pracují s velkými tabulkami (všechna řešení, všichni uživatelé, ...), se může hodit znalost některých pokročilých funkcí těchto tabulek.

Většina ovládacích prvků tabulky zobrazí krátkou nápovědu po najetí myši nad prvek.

	Jméno	Termín	Body	Přednáška	Skupina	Řešení
	MFF XML Schema	2014-04-23 21:59:59	20	NPRG036	MFF UK ST 0900	
	MFF XSLT	2014-05-07 21:59:59	20	NPRG036	MFF UK ST 0900	
	MFF XQuery	2014-05-28 21:59:59	20	NPRG036	MFF UK ST 0900	

Zobrazit 7 řádků na stránce

termín > "11-04-22"

8 1 9 10

Obrázek 14: Typická tabulka systému XML Check

Na tabulce v obrázku 14 se nachází tyto prvky:

1. **Prázdňá tabulka.** Symbol prázdné množiny znázorňuje, že v této tabulce není žádný řádek.
2. **Titulek.** Kliknutím na titulek sbalíte nebo rozbalíte tabulku.
3. **Vytvořit nový řádek.** V některých tabulkách je zde zobrazen symbol *plus*, který uživateli umožňuje přidat do tabulky nový řádek, např. vytvořit novou skupinu nebo zadat nový domácí úkol. Student ovšem nemá přístup k žádné z takových tabulek.
4. **Provést akci s řádkem.** Zde se zobrazují tlačítka, která umožňují uživateli provést nějakou akci nad daným řádkem, např. stáhnout dané řešení nebo se stát členem dané skupiny.
5. **Jméno sloupce.** Podle některých sloupců je možné tabulku řadit. Pokud najedete myši nad záhlaví takového sloupce, změní barvu. Kliknutím pak

můžete tabulku podle hodnot v tomto sloupci seřadit. Ikonka šipky vlevo od jména sloupce znázorňuje, zda je tabulka řazena vzestupně (šipka nahoru) nebo sestupně (šipka dolů).

6. **Aktivní řádek.** Nad tímto řádkem se právě nachází kurzor myši. V některých políčkách (například ve výpisu podrobností kontroly) může být tolik textu, že je zobrazena pouze jeho část a ikonka *plus*. V takovém případě, abyste zobrazili celý text, poklepejte na příslušný řádek nebo jednou klikněte na ikonku *plus*. Pro sbalení řádku na něj opět poklepejte nebo klikněte na ikonku *mínus*.
7. **Počet řádků na stránce.** Zde můžete nastavit, kolik řádků se má na stránce v tabulce zobrazit. Pokud nastavíte číslo menší než je počet řádků v tabulce, zobrazí se stránkovací tlačítka, pomocí nichž můžete přepínat mezi stránkami. Počet zobrazených řádků se aplikuje a uloží jako cookie na vašem počítači, když po vepsání čísla stisknete Enter, tabulátor nebo jinak opustíte prvek.
8. **Filtry.** Kliknutím na toto tlačítko zobrazíte nebo skryjete filtry. Filtry budou fungovat stále, i pokud je skryjete. Skrytím filtrů zmizí prvky 10 a 11.
9. **Zrušit všechny filtry.** Všechny filtry budou zrušeny a tabulka bude okamžitě překreslena.
10. **Přidat nový filtr.** Zobrazí malý formulář pro zadání nového filtru (viz dále).
11. **Aktivní filtr.** Tento filtr na obrázku 14 způsobuje, že jsou zobrazovány jen úkoly, jejichž termín odevzdání je až po 22. dubnu 2014. Kliknutím na ikonku popelnice můžete filtr zrušit.

Pokud je v tabulce velké množství řádků, možná se vám bude hodit vytvořit filtr.

Po kliknutí na tlačítko **Přidat nový filtr** se zobrazí malý formulář jako na obrázku 15. V něm jsou po řadě tyto ovládací prvky:

1. **Jméno sloupce.** Podle tohoto sloupce bude filtr vybírat řádky. Je možné filtrovat i podle sloupců, které nejsou zobrazeny, jako např. *id*.
2. **Možnost negace.** Vyberete-li položku ! (vykřičník), bude následující operátor znegován.

Obrázek 15: Formulář na vytvoření nového filtru

3. **Operátor.** Hodnota sloupce v každém řádku bude pomocí tohoto operátoru porovnána se zadanou hodnotou vpravo a podle toho bude řádek zobrazen nebo ne. Operátory jsou:
 - (a) == (rovnost; hodnota v řádku musí být úplně stejná jako zadaná hodnota)
 - (b) >, popřípadě < (větší nebo menší než; srovnávat takto lze pouze číselné hodnoty)
 - (c) ^= (začátek řetězce; obsah políčka musí začínat zadanou hodnotou)
 - (d) \$= (konec řetězce; obsah políčka musí končit zadanou hodnotou)
 - (e) *= (obsahuje; obsah políčka musí obsahovat zadanou hodnotu)
4. **Zadaná hodnota.** Obsah políčka bude porovnáván s touto hodnotou.
5. **Potvrdit filtr.** Kliknutím na toto tlačítko bude filtr aktivován a okamžitě aplikován.
6. **Zrušit vytváření filtru.** Kliknutím skryjete tento malý formulář.

5.2 Úkony studenta

Na začátku semestru se zaregistrujte, aktivujte účet přes e-mail a pak se přidejte do skupiny, kam chodíte na cvičení. Jakmile bude zadán první domácí úkol, přečtěte si jeho zadání, vypracujte ho a nahrajte řešení do systému. Systém vám okamžitě zkontroluje určité prvky vašeho řešení, např. přítomnost všech součástí DTD dle zadání nebo validitu souborů.

Pokud vám oznámí, že vaše řešení neprochází na 100%, řekne vám, kde jste udělali chybu. Řešení můžete opravit a nahrát znovu. Jako odevzdané řešení se

počítá pouze to řešení, které jste do systému nahráli jako poslední.

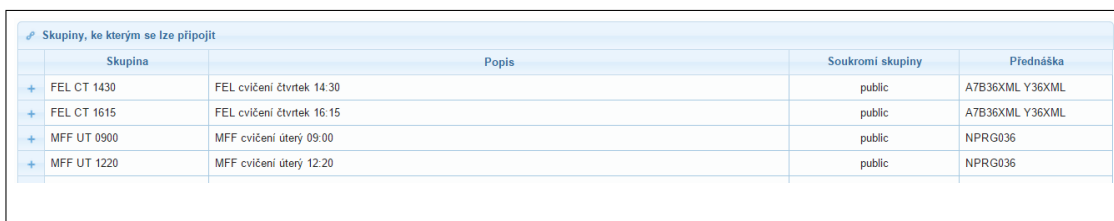
Nebo vám systém oznámí, že vaše řešení prochází na 100%. Tím ovšem úkol nemáte splněný. Podívá se na něj ještě váš cvičící, který vám teprve přidělí body – může jich přidělit méně než maximum, např. pokud váš dokument XML nemodeluje vhodně zvolenou realitu, je příliš jednoduchý, nesmyslný nebo jen chytře obchází automatické kontroly. Na druhou stranu, cvičící vám může přidělit maximální počet bodů, i když systém hlásí méně než 100%, pokud usoudí, že vaše řešení je v pořádku a chyba je v automatickém ohodnocování.

Cvičící se na vaše řešení bude dívat až poté, co uplyne termín odevzdání. Pokud chcete, aby bylo vaše řešení oznámkováno dříve, máte možnost *zažádat o předčasné oznámkování* (viz sekce 5.2.4).

Svůj aktuální stav bodů můžete sledovat na hlavní stránce systému hned po přihlášení.

5.2.1 Přidat se do skupiny

V rubrice **Student** je odkaz na stránku **Členství**, kde najdete tabulku **Skupiny, ke kterým se lze připojit** (obrázek 16). Kliknutím na tlačítko + u skupiny, do které se chcete přidat, se do této skupiny přidáte. Pokud je ve sloupci *Soukromí skupiny* ovšem napsáno, že skupina je *private* (soukromá), pak kliknutím na tlačítko pouze o členství požádáte a členství musí být ještě schváleno cvičícím, který má takovou skupinu na starosti.



Skupiny, ke kterým se lze připojit				
	Skupina	Popis	Soukromí skupiny	Předmět
+	FEL CT 1430	FEL cvičení čtvrtek 14:30	public	A7B36XML Y36XML
+	FEL CT 1615	FEL cvičení čtvrtek 16:15	public	A7B36XML Y36XML
+	MFF UT 0900	MFF cvičení úterý 09:00	public	NPRG036
+	MFF UT 1220	MFF cvičení úterý 12:20	public	NPRG036

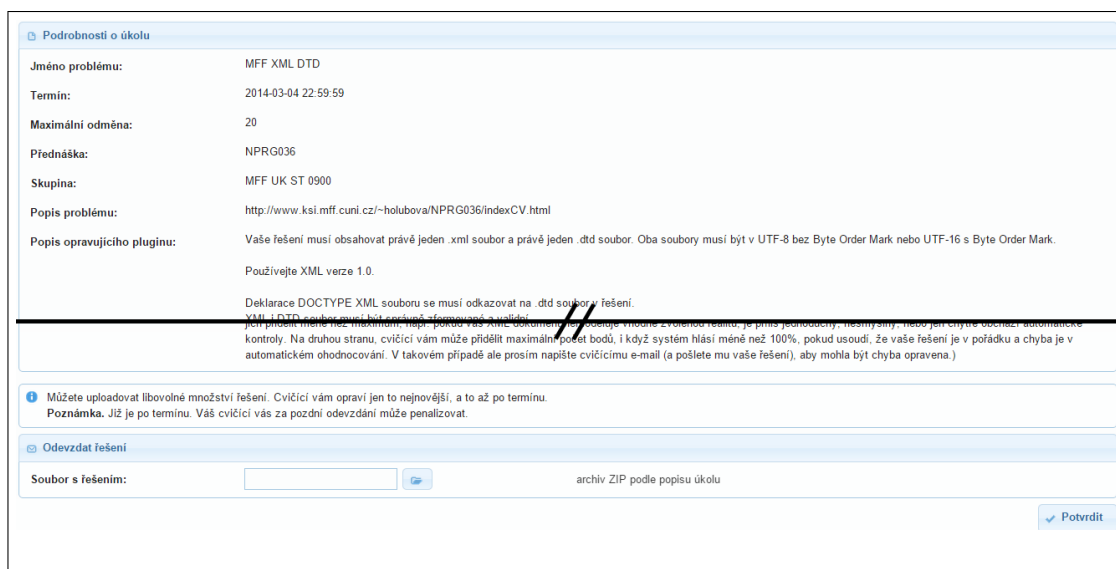
Obrázek 16: Skupiny, ke kterým se lze připojit

V předmětu Technologie XML se typicky soukromé skupiny nevyužívají.

Na stejné stránce je i tabulka **Aktivní členství a žádosti o členství**, která vám dovolí ze skupiny odejít. Odchodem ze skupiny se nesmažou žádné z vašich domácích úkolů ani bodů, ovšem nebudou se vám zobrazovat, dokud se do stejné skupiny opět nepřipojíte.

5.2.2 Odevzdat řešení domácího úkolu

V rubrice **Student** je odkaz na stránku **Úkoly**, kde jsou zadané úkoly pro vaši skupinu rozděleny do dvou tabulek podle toho, jestli termín jejich odevzdání již uplynul. Kliknutím na ikonku souboru vedle úkolu se dostanete na stránku s podrobnostmi, kde je vypsáno přesné zadání (obrázek 17). Ve spodní části této stránky je také formulář, s jehož pomocí můžete odevzdat řešení.



Podrobnosti o úkolu

Jméno problému: MFF XML DTD
Termín: 2014-03-04 22:59:59
Maximální odměna: 20
Přednáška: NPRG036
Skupina: MFF UK ST 0900
Popis problému: <http://www.ksi.mff.cuni.cz/~holubova/NPRG036/indexCV.html>
Popis opravujícího pluginu: Vaše řešení musí obsahovat právě jeden .xml soubor a právě jeden .dtd soubor. Oba soubory musí být v UTF-8 bez Byte Order Mark nebo UTF-16 s Byte Order Mark. Použijte XML verze 1.0.

Deklarace DOCTYPE XML souboru se musí odkazovat na .dtd soubor v řešení. XML DTD soubor musí být správně sformátován a validní. Je-li přidán méně než maximální počet bodů, můžete mít neúspěšnou zkoušku. Pokud jste zvolili možnost "Zvoleno řešení", je příloha jednoduchá, nezmyslná, nebo jen chybně označená automatické kontroly. Na druhou stranu, cvičící vám může přidělit maximální počet bodů, i když systém hlásí méně než 100%, pokud usoudí, že vaše řešení je v pořádku a chyba je v automatickém ohodnocování. V takovém případě ale prosím napište cvičícímu e-mail (a pošlete mu vaše řešení), aby mohla být chyba opravena.)

Můžete uploadovat libovolné množství řešení. Cvičící vám opraví jen to nejnovější, a to až po termínu.
Poznámka. Již je po termínu. Váš cvičící vás za pozdní odevzdání může penalizovat.

Odevzdat řešení

Soubor s řešením: archiv ZIP podle popisu úkolu

Obrázek 17: Formulář k odevzdání řešení

Mechanismus, jakým se odevzdávají řešení, je podrobněji popsán na začátku této kapitoly (sekce 5.2). Můžete nahrát, kolik řešení chcete, ale ohodnoceno bude pouze to, které nahrajete jako poslední.

Poznámka: Místo cesty, kterou zvolíte v dialogu na výběr souboru, se vám možná zobrazuje v textovém okénku cesta `C:\fakepath\`, respektive `/fakepath/`. To je v pořádku, jedná se o bezpečnostní prvek webových prohlížečů.

5.2.3 Zobrazit informace o nahraných domácích úkolech

V rubrice **Student** je odkaz na stránku **Řešení**, kde jsou všechna vaše odevzdaná řešení rozřazena do dvou tabulek: v jedné tabulce jsou neoznámkovaná (obrázek 18) a v druhé oznamkovaná řešení. Poklepnutím na řádek s řešením rozbalíte podrobnější informace pocházející z pluginu pro automatické hodnocení. Pozor, stoprocentní úspěch u automatického hodnocení a text ve stylu *Passed 7 out of 7 criteria*. pořád neznamená, že musíte dostat plný počet bodů.

Vaše řešení							
	Problém	Termín	%	Stav	Nahráno	Podrobnosti	
	MFF XML DTD	2014-03-04 22:59:59	100%	nejnovější	2014-11-29 12:40:53	Passed 7 out of 7 criteria. [+]	
Zobrazit 500 řádků na stránce							

Obrázek 18: Neoznámkovaná řešení (z pohledu studenta)

Řešení mohou být v těchto stavech:

- **normální:** toto je jedno z vašich starších řešení; můžete ho bez obav smazat
- **nejnovější:** jakmile nadejde termín odevzdání, tak toto bude řešení, které cvičící oznámkuje
- **učitel upozorněn:** upozornili jste cvičícího, že toto je vaše finální řešení a že mu dáváte právo ho oznámkovat i před termínem
- **oznámkováno:** tomuto řešení přiřadil cvičící nějaké body; jakmile je vaše řešení oznámkováno, nemůžete s ním již nic dále dělat

Tlačítka vlevo od každého řešení vám mohou umožnit provádět následující akce:

- **stáhnout řešení:** tímto si můžete stáhnout nahrané řešení a zkontrolovat tak, že jste nahráli správný soubor
- **stáhnout výstup:** pokud plugin automatického hodnocení vyprodukoval nějaký výstup, můžete si ho stáhnout
- **upozornit cvičícího:** obvykle by váš cvičící oznámkoval vaše nejnovější řešení a udělal by to někdy po termínu; pokud ovšem chcete mít toto řešení oznámkováno již dříve, můžete cvičícího na tento fakt upozornit – tím dáte cvičícímu právo váš úkol oznámkovat; jakmile to udělá, již nemůžete svoje řešení změnit
- **odstranit řešení:** řešení bude navždy skryto jak vám, tak cvičícím

5.2.4 Zažádat o předčasné oznámkování

Pokud nechcete čekat na termín odevzdání a jste si jistí, že již nebudete chtít vaše řešení upravovat, můžete svoje řešení prohlásit za finální a upozornit cvičícího, že byste byli rádi, kdyby řešení bylo oznámkováno. K tomu využijte tlačítko *upozornit cvičícího* na stránce **Řešení** v rubrice **Student** (podrobný popis je v předcházející sekci).

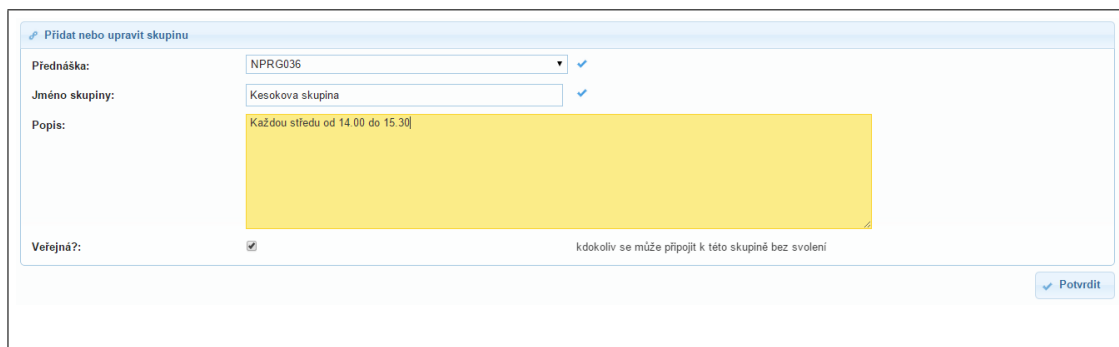
5.3 Úkony cvičícího

Přednáška v sobě seskupuje několik **skupin** a **problémů**. Problémy odpovídají zadáním, např. „vytvořte schéma XSD“. Skupiny odpovídají rozvrhovým lístkům. **Úkol** je instance problému, které je navíc přiřazen termín.

Jako cvičící budete spravovat **skupinu** (nebo několik skupin). Přednášku za vás vytvoří přednášející.

5.3.1 Vytvořit skupinu

V rubrice **Cvičící** je odkaz na stránku **Skupiny**, kde můžete kliknutím na tlačítko + v levém horním rohu tabulky vytvořit novou skupinu (obrázek 19). Jméno skupiny nebudete moci po vytvoření změnit, stejně tak nebudete moci změnit, k jaké přednášce skupina patří. Skupina by měla odpovídat rozvrhovému lístku.



The screenshot shows a web form titled "Přidat nebo upravit skupinu". It contains the following fields and controls:

- Přednáška:** A dropdown menu with the selected value "NPRG036".
- Jméno skupiny:** A text input field containing "Kesokova skupina".
- Popis:** A large yellow text area containing the text "Každou středu od 14 00 do 15 30".
- Veřejná?:** A checkbox that is checked.
- Below the checkbox, there is a small text note: "kdokoliv se může připojit k této skupině bez svolení".
- At the bottom right, there is a blue button labeled "Potvrdit".

Obrázek 19: Formulář pro vytvoření nebo úpravu skupiny

Pomocí zaškrtnutí tlačítka *Veřejná* můžete určit, zda se může do této skupiny připojit kdokoliv, nebo zda musíte každou žádost o připojení nejdříve schválit. Doporučuje se, aby všechny skupiny předmětu *Technologie XML* byly veřejné.

5.3.2 Zadat domácí úkol

V rubrice **Cvičící** je odkaz na stránku **Zadat úkoly**, kde můžete zadat nové úkoly nebo upravovat termín odevzdání existujících úkolů (obrázek 20).

Jakmile zadáte úkol, většině studentů ve vaší skupině přijde e-mail, že byl zadán nový úkol; dávejte proto pozor, že již napoprvé zadáte správnou odměnu

Obrázek 20: Formulář pro vytvoření nebo úpravu úkolu

a správný termín odevzdání. V předmětu Technologie XML se typicky používá odměna 20 bodů.

5.3.3 Hodnotit odevzdaná řešení

V rubrice **Cvičící** je odkaz na stránku **Oznámkovat řešení**, kde se vám zobrazí řešení studentů z vašich skupin (obrázek 21). Zobrazují se od každého studenta jen jeho nejnovější řešení. Pomocí tlačítka *stáhnout řešení* si můžete stáhnout studentovo řešení na svůj počítač, ohodnotit ho a počet bodů, který chcete udělit, pak můžete vepsat do formuláře po kliknutí na tlačítko *oznámkovat řešení* (obrázek 22).

Problém	Skupina	Nahráno	%	Podrobnosti
MFF XML DTD	Kesokova skupina	2014-11-29 21:40:52	100%	Passed 7 out of 7 criteria. [+]

Obrázek 21: Seznam neoznámkovaných řešení

Neznámkuje řešení, pokud ještě neprošel termín odevzdání, pokud vás o to student výslovně nepožádá (například pomocí funkce systému na upozornění učitele).

Pokud jste se rozhodli neudělit plný počet bodů, napište prosím ve formuláři studentovi poznámku s důvodem, proč jste plný počet bodů neudělili.

5.3.4 Kontrolovat, zda je řešení plagiátem

V rubrice **Cvičící** na stránce **Oznámkovat řešení** se vám ve sloupečku *Podrobnosti* může u nějakého řešení objevit text „Toto řešení je podezřele podobné

Obrázek 22: Formulář pro přidělení počtu bodů

jiným řešením.“ V takovém případě klikněte na čtvrté tlačítko v řádce (s titulkem „zobrazit podrobnosti o podobnosti s jinými domácími úkoly“). Zobrazí se vám podrobné informace o domácím úkolu spolu se jménem autora a jeho e-mailovou adresou (obrázek 23).

Pod těmito informacemi se nachází tabulka s informacemi o řešeních, která byla odevzdána dříve, a přitom jsou tomuto řešení velmi podobná. Kliknutím na druhou ikonku v řádku (*prozkoumat toto řešení*) se můžete dozvědět podrobnosti o těchto podobných řešeních, včetně e-mailových adres jejich autorů.

5.3.5 Zobrazit bodové hodnocení všech studentů

V rubrice **Cvičící** je odkaz na stránku **Hodnocení studentů**, kde se zobrazují průběžné výsledky všech studentů ve vašich skupinách (obrázek 24). Zobrazují se tam pouze ti studenti, kterým byl oznámkován alespoň jeden domácí úkol.

Pokud jste administrátorem nebo jste dostali právo na správu všech skupin, uvidíte na této stránce hodnocení studentů ze všech skupin, nikoliv pouze z vašich vlastních.

O tomto řešení

Jméno studenta: Arkij Kesok
E-mail studenta: petrhudec2010@gmail.com
Přidělené body: 0
Výsledky automatického hodnocení: Toto řešení je podezřele podobné jiným řešením.
 =====
 Passed 7 out of 7 criteria.
 XML is well-formed ... PASSED
 XML is valid to supplied DTD ... PASSED
 DTD is valid ... PASSED
 The XML's DOCTYPE declaration refers to the provided DTD. ... PASSED
 DTD document contains required constructs ... PASSED
 XML document contains required constructs ... PASSED
 Documents contain required special constructs ... PASSED

Nahráno: 2015-01-12 15:53:43
Stav řešení: latest
Odkaz ke stažení: [Stáhnout toto řešení](#)
Nápověda: Tabulka na této stránce ukazuje řešení jiných studentů, kterým se toto řešení podobá. Fakt, že tabulka není prázdná, neznamená nutně, že student opisuje. Řešení, o nichž si systém myslí, že z nich bylo toto řešení pravděpodobně opsáno, jsou označena jako "podezřelá" a mají vysoké skóre podobnosti.

Řešení, kterým se toto řešení podobá

	ID	Podezřelá shoda	Podobnost	Podrobnosti	Autor	Nahráno	Stav
	12837	Ano	100%	PRIMARY_XML_FILE comparison (100%, suspicious): [✖]	Petr Hudecěk	2014-02-28 20:52:26	graded

Zobrazit 47 řádků na stránce 0 / 0

Obrázek 23: Informace o podobnosti s jinými řešeními

MF F UK ST 0900 (NPRG036)

Student	MF F XML DTD	MF F DOM SAX	MF F XPath	MF F XML Schema	MF F XSLT	MF F XQuery	Součet
Ondřej Papík	20	20	20	20	20	20	120
Pavel Brožek	20	20	20	20	20	20	120
Jiri Svancara	19	20	20	20	20	20	119
Vojtěch Vondra	19	20	20	20	20	20	119
Petr Hudecěk	20	19	20	20	20	20	119

Obrázek 24: Průběžné hodnocení všech studentů

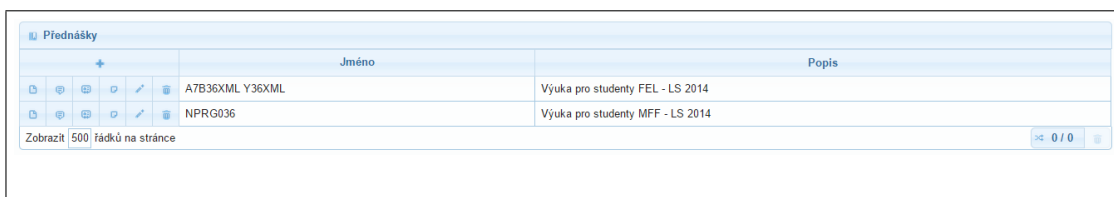
5.4 Úkony přednášejícího









Přednáška v sobě seskupuje několik **skupin** a **problémů**. Problémy odpovídají zadáním, např. „vytvořte schéma XSD“. Skupiny odpovídají rozvrhovým lístkům. **Úkol** je instance problému, které je navíc přiřazen termín.

Poznámka: Jako přednášející vytvořte přednášku, ale skupiny vytvořte jen ty, kde sami fungujete jako cvičící. Je důležité, aby si daný cvičící vytvořil svoji skupinu sám, aby pak mohl hodnotit řešení studentů v této skupině.

5.4.1 Vytvořit přednášku

V rubrice **Přednášející** je odkaz na stránku **Přednášky**. Zde můžete vytvářet nové přednášky nebo měnit popis existujících přednášek. Pomocí čtyř tlačítek u každé přednášky se můžete přesunout na objekty související s danou přednáškou (obrázek 25).



Přednášky		
	Jméno	Popis
   	A7B36XML Y36XML	Vyuka pro studenty FEL - LS 2014
   	NPRG036	Vyuka pro studenty MFF - LS 2014

Zobrazit 500 řádků na stránce < 0 / 0

Obrázek 25: Seznam přednášek

5.4.2 Vytvořit problém

V rubrice **Přednášející** je odkaz na stránku **Problémy**. Zde můžete do systému zadávat nové problémy (obrázek 26). Problémy mohou být volitelně asociované s *hodnotícím pluginem*. Hodnotící plugin se spustí vždy po nahrání nového řešení a pokusí se detekovat, zda je řešení správné nebo ne. Tyto informace dá k dispozici studentovi i cvičícímu.

Některé pluginy vyžadují **parametry**. Hodnoty těchto parametrů jsou společné pro celý problém a zpravidla se jedná o čísla oddělená středníkem. Například úkol číslo 3 má jako parametr minimální počet výrazů XPath, které musí řešení obsahovat. Pokud žádné parametry nezadáte, použije plugin výchozí hodnotu, která je v jeho zdrojovém kódu napevno.

Obrázek 26: Formulář pro vytvoření nebo úpravu problému

Výchozí hodnoty ve zdrojových kódech pluginů pro Technologie XML odpovídají správným a používaným parametrům.

5.4.3 Nahrát nový plugin

V rubrice **Přednášející** je odkaz na stránku **Pluginy**, kde můžete přidat do systému nový plugin (obrázek 27). Pokud chcete upravit popis existujícího pluginu, musíte změnit text manifestu pluginu na disku a poté použít zvláštní akci na stránce **Jiná administrace**.

Obrázek 27: Formulář pro nahrání pluginu

Přidání pluginu způsobí vytvoření záznamu o pluginu v databázi a obsah nahraného archivu ZIP se rozbalí do složky s pluginy. Jméno pluginu nelze po přidání snadno změnit.

Návod na vytvoření pluginu najdete v [1]. Jediná změna ve formátu pluginů oproti původní verzi je ta, že manifest pluginu musí nyní obsahovat ještě prvek `<identifier></identifier>`, jehož obsahem je jedinečný textový identifikátor daného pluginu. Tohoto identifikátoru využívá kontrola podobnosti. Například plugin pro 1. domácí úkol (XML/DTD) má identifikátor *XMLDTD*.

5.4.4 Vytvořit kvíz nebo písemku

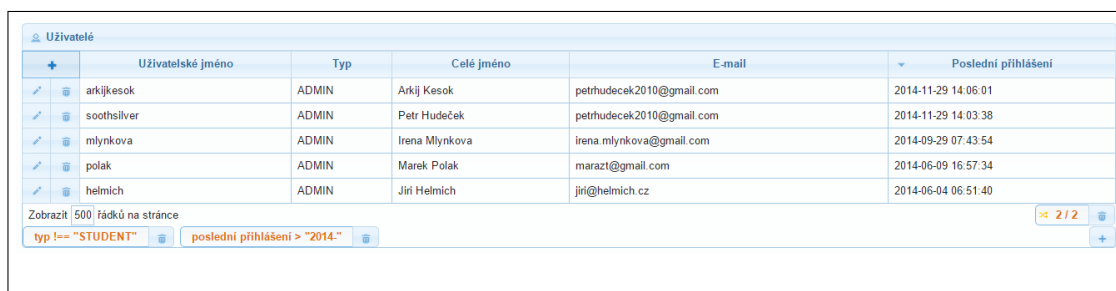
Součástí původní verze systému XML Check byla i možnost tvorby otázek a testů (test se skládá z otázek), které se daly vytisknout a předložit studentům. Tato možnost zatím nebyla nikdy využita, ovšem je stále dostupná.

Uživatelskou dokumentaci k této části programu naleznete v [1].

5.5 Úkony administrátora

5.5.1 Spravovat uživatelské účty

V rubrice **Systém** je odkaz na stránku **Uživatelé**, kde můžete vytvářet nové uživatele a prohledávat databázi existujících uživatelů (obrázek 28). U každého uživatele se zobrazuje jeho celé jméno a e-mail. Uživatelům můžete nastavovat jejich roli (v systému XML Check se uživatelská role nazývá „druh uživatele“ nebo „typ uživatele“). Role určuje, jaká práva uživatel má. Uživatelské role lze spravovat nezávisle, viz sekce 5.5.4.



	Uživatelské jméno	Typ	Celé jméno	E-mail	Poslední přihlášení
+	arkijkesok	ADMIN	Arkij Kesok	petrhudecek2010@gmail.com	2014-11-29 14:06:01
+	soothsilver	ADMIN	Petr Hudeček	petrhudecek2010@gmail.com	2014-11-29 14:03:38
+	mlynkova	ADMIN	Irena Mlynkova	irena.mlynkova@gmail.com	2014-09-29 07:43:54
+	polak	ADMIN	Marek Polak	marazt@gmail.com	2014-06-09 16:57:34
+	helmich	ADMIN	Jiří Helmich	jiři@helmich.cz	2014-06-04 06:51:40

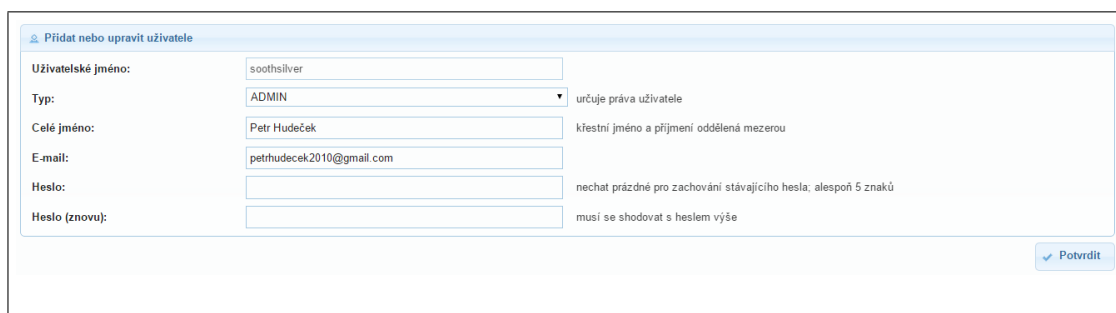
Zobrazit 500 řádků na stránce

typ != "STUDENT" poslední přihlášení > "2014."

2 / 2

Obrázek 28: Seznam všech uživatelů

Kliknutím na tlačítko + v levém horním rohu tabulky můžete založit nového uživatele (obrázek 29). Takový uživatel si nemusí aktivovat svůj účet přes e-mail, jeho účet bude rovnou aktivovaný.



Přidat nebo upravit uživatele

Uživatelské jméno: soothsilver

Typ: ADMIN určuje práva uživatele

Celé jméno: Petr Hudeček křestní jméno a příjmení oddělená mezerou

E-mail: petrhudecek2010@gmail.com

Heslo: nechat prázdné pro zachování stávajícího hesla; alespoň 5 znaků

Heslo (znovu): musí se shodovat s heslem výše

Potvrdit

Obrázek 29: Formulář pro vytvoření nebo úpravu uživatelského účtu

5.5.2 Upravit uvítací zprávu nebo text posílaných e-mailů

Ve složce *documents/* v kořenovém adresáři aplikace se nachází několik textových dokumentů, které můžete upravit. Dokumenty budou vypsaný uživatelům nebo poslány jako e-mail tak, jak je napíšete. Značky HTML budou interpretovány, značky PHP ovšem nikoliv. Kromě toho můžete použít v každém dokumentu, který reprezentuje e-mail, zvláštní řetězce, které budou nahrazeny příslušným textem při generování e-mailu.

Ve složce jsou tyto soubory, které reprezentují text:

- **changelog.txt:** Tento text se zobrazí na stránce **Seznam změn** pod rubrikou **System**.
- **motd.txt:** Tento text se zobrazí na úvodní stránce po přihlášení. Pokud tento soubor neexistuje, zobrazí se text „*Žádná uvítací zpráva není nastavena.*“

Následující soubory reprezentují e-maily. První řádka v souboru tvoří předmět, zbytek řádek tvoří tělo e-mailu.

- **registrationEmail.txt:** Tento e-mail se pošle uživateli poté, co klikne na tlačítko *Zaregistrovat*. V e-mailu se nahradí výrazy:
 - *{Username}* přihlašovací jménem uživatele
 - *{ActivationCode}* aktivačním kódem, který musí uživatel zadat
 - *{Link}* odkazem na formulář pro zadání aktivačního kódu
- **newAssignmentEmail.txt:** Tento e-mail se pošle studentovi, když cvičící v jeho skupině zadá nový domácí úkol. V e-mailu se nahradí výrazy:
 - *{Problem}* jménem řešeného problému
 - *{Group}* jménem skupiny, do níž byl úkol zadán
 - *{Deadline}* termínem odevzdání úkolu
 - *{Date}* datem a časem, kdy byl e-mail odeslán
 - *{Link}* odkazem na zadání tohoto úkolu
- **submissionHandedOffEmail.txt:** Tento e-mail se pošle cvičícímu, když student požádá o předčasné oznámkování svého řešení. V e-mailu se nahradí výrazy:
 - *{RealName}* celým jménem studenta

- *%{Email}* e-mailovou adresou studenta
- *%{Link}* odkazem na místo, kde je možné řešení stáhnout
- **successEmail.txt, failureEmail.txt:** Takový e-mail se pošle studentovi, když cvičící oznámkuje jeho řešení. Pokud cvičící přidělil maximální počet bodů, pošle se *successEmail.txt*, jinak se pošle *failureEmail.txt*. V e-mailu se nahradí výrazy:
 - *%{Assignment}* jménem problému, který byl řešen
 - *%{Points}* počtem bodů, které student získal
 - *%{Maximum}* maximálním počtem bodů, které student mohl získat
 - *%{Explanation}* krátkým vysvětlením, které podal cvičící (může být prázdné)
 - *%{Link}* odkazem na místo, kde si student může své řešení stáhnout

5.5.3 Upravit pokročilá nastavení v souboru config.ini

Ve složce *core/* v kořenovém adresáři je soubor *config.ini*. Úpravu tohoto souboru musíte provést před prvním spuštěním systému XML Check. Soubor používá syntaxi konfiguračních souborů Windows kromě toho, že hodnoty vlastností mohou být uzavřeny v uvozovkách. Středník značí komentář až do konce řádku. Mezery jsou ignorovány. V souboru můžete nastavit tyto vlastnosti:

- Sekce **database**
 - **host:** IP adresa nebo DNS jméno databázového serveru
 - **user:** přihlašovací jméno k databázi
 - **pass:** heslo k databázi
 - **db:** jméno používané databáze
- Sekce **mail**
 - **host:** IP adresa nebo DNS jméno SMTP serveru
 - **port:** port, na kterém SMTP server komunikuje
 - **security:** buď „“(prázdný řetězec, žádná bezpečnost není použita) nebo „ssl“nebo „tls“
 - **user:** přihlašovací jméno k SMTP serveru
 - **pass:** heslo k SMTP serveru
 - **from_name:** jméno odesilatele e-mailů ze systému XML Check

- **from_address**: e-mailová adresa, ze které budou odesílány e-maily ze systému
- Sekce **bin**
 - **phpCli**: cesta k souboru interpreta PHP; pokud se nachází v proměnné prostředí PATH, stačí napsat jen jméno souboru – *php* (i na Windows).
 - **java**: cesta ke spustitelnému souboru Java; pokud se nachází v proměnné prostředí PATH, stačí napsat jen jméno souboru – *java* (i na Windows).
 - **javaArguments**: dodatečné argumenty pro příkazovou řádku pro spuštění modulu na kontrolu podobnosti; doporučenou hodnotou této vlastnosti je `-Xms128M -Xmx2048M`, čímž nastavíte dostatečně velký paměťový limit
- Sekce **defaults**
 - **submissionFileName**: název, pod jakým si může student a cvičící stáhnout odevzdaná řešení; za název bude připojena koncovka *zip*.
 - **pluginTestFileName**: to samé pro testy pluginů
 - **pluginOutputFileName**: to samé pro výstup hodnotících pluginů
- Sekce **roots**
 - **http**: adresa (včetně začátku *http://*), na které lze přistoupit k této instanci systému XML Check
- Sekce **similarity**
 - **enableSimilarityChecking**: uveďte *true*, pokud se má provádět kontrola podobnosti domácích úkolů
 - **enableZhangShasha**: uveďte *true*, pokud se má používat algoritmus Zhang-Shasha; pokud uvedete *false*, bude se používat pouze Levenshteinova vzdálenost
 - **levenshteinMasterThreshold**: pokud úroveň podobnosti v procentech podle Levenshteinova algoritmu bude menší než toto číslo, nebude o této podobnosti cvičící nijak informován
 - **zhangShashaSuspicionThreshold**: pokud úroveň podobnosti podle algoritmu Zhang-Shasha bude vyšší nebo rovna tomuto číslu, bude cvičící upozorněn, že se jedná o pravděpodobný plagiát
 - **levenshteinSuspicionThreshold**: to samé pro Levenshteinův algorit-

mus

Soubor *config.ini* by tedy mohl vypadat například takto:

```
1 [database]
2 host = "127.0.0.1"
3 user = "asm_user"
4 pass = "asm_password"
5 db = "asm_database"
6
7 [mail]
8 host = "smtp.gmail.com"
9 port = "465"
10 security = "ssl"
11 user = "xmlcheck@gmail.com"
12 password = "email_password"
13 from_name = "XML Check System"
14 from_address = "xmlcheck@gmail.com"
15
16 [bin]
17 phpCli = "php"
18 java = "java"
19 javaArguments = "-Xms128M -Xmx2048M"
20
21 [defaults]
22 submissionFileName = "submission"
23 pluginTestFileName = "input"
24 pluginOutputFileName = "output"
25
26 [roots]
27 http = "http://xmlcheck.projekty.ms.mff.cuni.cz/"
28
29 [similarity]
30 enableSimilarityChecking = true
31 enableZhangShasha = true
32 levenshteinMasterThreshold = 60
33 zhangShashaSuspicionThreshold = 90
34 levenshteinSuspicionThreshold = 80
```

5.5.4 Spravovat druhy uživatelských účtů

V rubrice **System** je odkaz na stránku **Druhy uživatelů**, kde můžete zakládat nové druhy uživatelů (uživatelské role), zjišťovat, k jakým oprávněním dávají jednotlivé role přístup a měnit tato oprávnění (obrázek 30).

Ve formuláři pro přidání nebo změnu druhu uživatele kliknutím na ikonku oprávnění toto oprávnění dané roli přidáte nebo odeberete (obrázek 31). Dávejte pozor, ať neodeberete administrátorovi oprávnění měnit uživatelské role, jinak budete muset systém opravovat přímým přístupem do databáze.

Druhy uživatelů a jejich práva									
+	Jméno	Uživatelé	Členství ve skupinách	Pluginy	Úkoly	Známkování	Přednášky	Skupiny	Jiné
	STUDENT								
	ADMIN	+ / / /		+ / / /			+ / /	+ / /	
	lecturer			+ / /			+ /		
	tutor							+ /	
	administrator	+ / / /		+ / /					

Zobrazit 500 řádků na stránce 0 / 0

Obrázek 30: Seznam všech typů uživatelů

Přidat nebo upravit typ uživatele

Jméno:

Uživatelé: + / / /

Členství ve skupinách: + / /

Pluginy: + / / /

Úkoly: +

Známkování: + / /

Přednášky: + / /

Skupiny: + / /

Jiné: +

Potvrdit

Obrázek 31: Formulář pro vytvoření nebo úpravu typu uživatele

Jednotlivá oprávnění, která lze přidělit, jsou:

- Sekce **Uživatelé**
 - Přidat uživatele
 - Zobrazit uživatele
 - Upravovat uživatele
 - Odstraňovat uživatele
 - Upravovat druhy uživatelů
- Sekce **Členství ve skupinách**
 - Připojovat se k veřejným skupinám
 - Žádat o členství u soukromých skupin
 - Připojovat se k soukromým skupinám bez povolení
- Sekce **Pluginy**
 - Přidat pluginy
 - Zobrazovat pluginy
 - Upravovat pluginy
 - Odstraňovat pluginy. *Pozor, odstraněním pluginu můžete napáchat velkou škodu, neboť budou všechna řešení nadobro odloučena od svého pluginu, a nebude tedy možné je použít pro kontrolu podobnosti.*
 - Spouštět testy pluginů
- Sekce **Úkoly**
 - Odevzdávat řešení úkolů
- Sekce **Známkování**
 - Známkovat úkoly

- Zobrazovat autory řešení. *Cvičící, který tuto pravomoc nemá, místo jména studenta uvidí text „skryto“.*
- Měnit počet bodů již oznámkovaných úkolů
- Sekce **Přednášky**
 - Vytvářet přednášky
 - Upravovat vlastní přednášky
 - Upravovat libovolné přednášky
- Sekce **Skupiny**
 - Vytvářet skupiny
 - Upravovat vlastní skupiny
 - Upravovat libovolné skupiny. *I tehdy, když má cvičící toto právo, tak ve svých tabulkách uvidí řešení pouze těch studentů, kteří jsou v jeho skupině.*
- Sekce **Jiné**
 - Provádět jiné administrativní úkony. *Toto právo umožňuje uživateli znovu nahrát popis pluginů ze souboru do databáze a prohlížet výsledky kontroly podobnosti.*

Poznámka. Není možné smazat roli STUDENT (s databázovým identifikačním číslem 1), protože tato role je automaticky přiřazena všem uživatelům, když se zaregistrují.

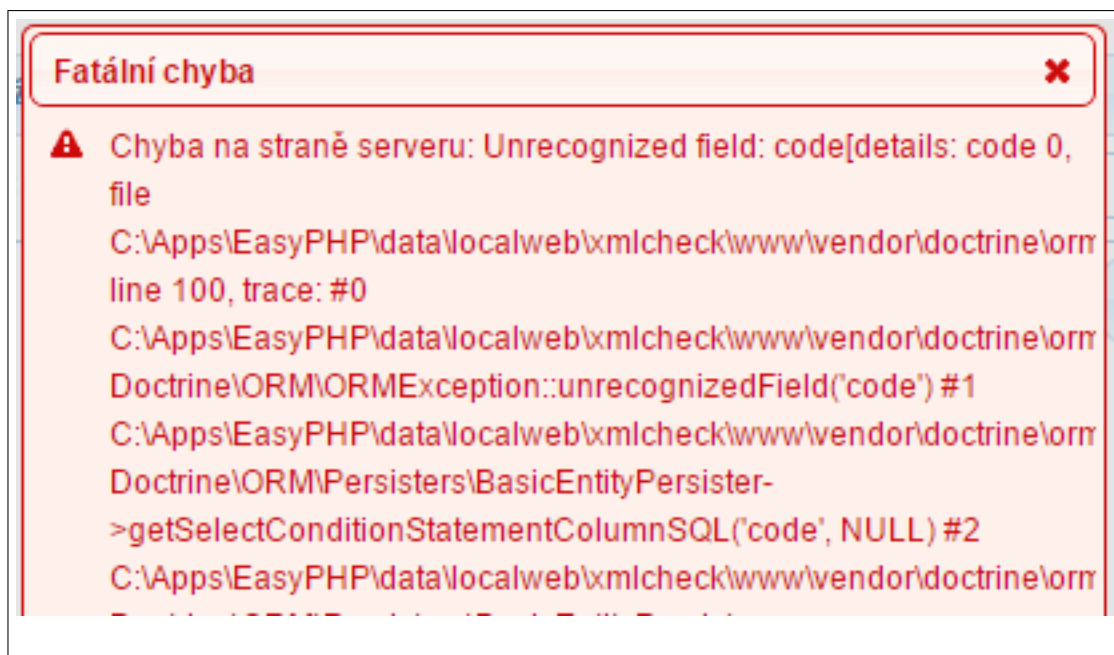
5.6 Řešení problémů

5.6.1 Systém mi vrátil nesrozumitelnou chybovou hlášku.

Nesrozumitelné chybové hlášky jsou většinou nadepsány **Fatální chyba** a jejich text začíná slovy **Chyba na straně serveru** (obrázek 32). Takové chyby jsou zapříčiněné programátorskou chybou v systému. Kontaktujte administrátora systému nebo vašeho cvičícího. Pokud kontaktujete cvičícího včas a řeknete, že vám chyba zabraňuje v odeslání domácího úkolu, nebudete penalizováni za pozdní odevzdání.

5.6.2 Vidím jen prázdnou obrazovku.

Při vykonávání kódu Javascriptu narazil prohlížeč nejspíše na chybu a ukončil vykonávání kódu, což znemožnilo dokončení vykreslení systému. Je možné, že je to způsobeno programátorskou chybou. Je také možné, že jste se pokusili o provedení



Obrázek 32: Příklad nesrozumitelné chybové hlášky

nemožné akce. Zkuste obnovit stránku stisknutím klávesy F5, popř. zkuste vymazat mezipaměť prohlížeče.

Pokud se vám prázdná obrazovka zobrazila během normálního chodu programu, kontaktujte administrátora nebo cvičícího.

5.6.3 Zobrazují se mi stará data; provedl jsem akci a ta se neprojevila.

Váš prohlížeč nejspíše data stále načítá z mezipaměti. Obnovte stránku kliknutím na tlačítko *Obnovit* v prohlížeči nebo stisknutím klávesy F5 nebo klávesové zkratky Ctrl+R. Pokud ani toto nepomůže, vymažte mezipaměť a cookies v prohlížeči.

5.7 Instalace

Systém *XML Check* je distribuován v jednom archivu ZIP. Tento archiv obsahuje soubory, které se přímo rozbalí na váš webový server. Kromě toho také obsahuje jeden databázový skript, *install.sql*, který má pomoci s prvotní instalací systému.

Postup při instalaci se liší podle toho, zda chcete upravovat zdrojový kód systému. Pokud chcete kód upravovat, použijte návod v sekci 5.7.3.

Kromě samotné instalace systému je ještě třeba nainstalovat hodnotící pluginy pro jednotlivé úkoly (viz také sekce 5.7.4).

5.7.1 Požadavky na systém

Systém XML Check neklade zvláštní požadavky na operační systém, vyžaduje ovšem, aby byl nainstalován webový server Apache alespoň ve verzi 2.2, databázový server MySQL alespoň ve verzi 5.0, a PHP alespoň ve verzi 5.6; PHP musí být nainstalováno navíc zaprvé jako modul pro server Apache, a zadruhé jako spustitelný soubor pro příkazovou řádku.

Pokud chcete, aby fungovaly pluginy pro Technologie XML a kontrola podobnosti, nainstalujte také JRE pro Javu, alespoň ve verzi 8.

Konkrétní verze, na kterých byl systém otestován, jsou:

- Apache 2.4.7 [22]
- MySQL 5.6.15 Community Edition [23]
- PHP 5.6.1 [24] (nižší verze PHP než 5.6 nebudou fungovat)
- Java JRE 1.8.0_20 [25]

Pro odesílání e-mailů navíc budete potřebovat SMTP server a údaje k němu. Je možné využít veřejný SMTP server Googlu.

5.7.2 Instalace pro uživatele

Tento návod použijte pouze, pokud chcete systém XML Check nainstalovat a používat bez toho, že byste upravovali jeho zdrojový kód. Pokud chcete systém XML Check také upravovat a vylepšovat, použijte místo toho návod označený *Instalace pro programátora*.

1. V konfiguračním souboru *php.ini* nastavte `date.timezone = Europe/Prague` nebo jinou časovou zónu. Bez tohoto nastavení nebude systém správně fungovat. Dále se ujistěte, že má PHP povolené použití rozšíření *openssl*. Také zajistěte, aby *max_execution_time* v PHP byl dostatečně vysoký (doporučeno alespoň 60 sekund).
2. Ujistěte se, že Apache má povolené rozšíření *mod_rewrite*.

3. Extrahujte obsah přiloženého archivu ZIP do složky uvnitř složky DocumentRoot serveru Apache.
4. Vytvořte MySQL databázi, kterou bude používat systém XML Check.
5. Založte pro server MySQL uživatelský účet, s jehož pomocí se bude systém XML Check připojovat k databázovému serveru. Tento účet potřebuje úplný přístup k databázi, kterou má systém XML Check používat, vyjma privilegia GRANT.
6. Pokud chcete používat kontrolu podobnosti, nastavte v konfiguračním souboru databáze *my.ini* vlastnosti *max_allowed_packet* na *512M* nebo více.
7. Spusťte skript *install.sql* například pomocí příkazu `mysql -u (MySQLUserName) -p -d(DatabaseName) < install.sql`.
8. Mezi extrahovanými soubory najdete složku *core*, uvnitř ní soubor *config.ini*. Nastavte v něm správné údaje podle instrukcí v sekci 5.5.3, konkrétně:
 - (a) Nastavte správné přihlašovací údaje k databázi MySQL.
 - (b) Nastavte správné přihlašovací údaje pro server SMTP, pomocí kterého bude systém odesílat e-maily.
 - (c) Nastavte vlastnost *roots/http* na webovou adresu, kterou server Apache přeloží na soubor *index.php* v adresáři, kam jste extrahovali systém XML Check. Tato adresa by měla obsahovat i prefix protokolu *http://*.
9. Ujistěte se, že spustitelný program *java* (popř. *java.exe*) a interpret *php* (popř. *php.exe*) jsou v proměnné prostředí PATH.
10. Ujistěte se, že uživatelský účet operačního systému, pod kterým běží webový server, může zapisovat do adresáře *files* uvnitř adresáře, kam jste extrahovali systém, a to i do všech jeho podadresářů.
11. Systém je nyní nainstalován. Administrátorský účet se jmenuje *admin* a má přiřazené heslo *admin*.

5.7.3 Instalace pro programátora

Pokud chcete upravovat zdrojový kód systému XML Check, nainstalujte systém pomocí tohoto návodu.

1. Provedte všechny kroky z *Instalace pro uživatele*, až na to, že místo instalač-

ního balíčku použijte projekt IntelliJ IDEA, který se nachází na přiloženém disku (viz příloha A). Pro jeho použití nepotřebujete prostředí IntelliJ IDEA mít nebo používat.

(a) Adresář *java-plugins* a podadresář *phptests/plugins* se vztahují pouze k pluginům pro instanci systému XML Check na MFF. Možná je nebudete potřebovat a můžete je bez problémů smazat.

2. Nainstalujte aplikaci *Composer* a ujistěte se, že cesta k jejím binárním souborům je v proměnné prostředí PATH.
3. V adresáři, kam jste nainstalovali systém XML Check, spusťte v podadresáři *www/* příkaz `composer install`.
4. Systém je nyní nainstalován.

5.7.4 Pracujete na instanci XML Check na MFF?

Pokud chcete programovat změny v systému XML Check, a to přímo v instanci pro předmět Technologie XML, můžete s výhodou využít balíček na přiloženém disku, kde je připraven projekt pro vývojové prostředí *IntelliJ IDEA*, který v sobě obsahuje všechny části projektu: samotnou webovou aplikaci, pluginy pro Java, modul na kontrolu podobnosti, jednotkové testy, parser DTD a generátor dokumentace.

Co se týče instalace pluginů, je v tomto případě doporučeno nainstalovat pluginy přes uživatelské rozhraní systému XML Check, nýbrž zkopírováním databázových dat z běžícího serveru.

6. Programátorská dokumentace

Součástí programátorské dokumentace je také dokumentace tříd vygenerovaná nástrojem Doxygen. Ta se také nachází na přiloženém disku.

6.1 Základní dokumentace

Podstatná část programátorské dokumentace a architektury systému se nezměnila od původní verze. V této práci tedy popíšeme jen ty části, kde původní dokumentace již není kvůli změnám platná, a ty části aplikace, které v původní verzi neexistovaly.

Původní dokumentaci najdete v [1].

6.2 Adresářová struktura projektu

Celý projekt je možné nalézt na přiloženém disku. Má tuto adresářovou strukturu:

- **dtd-parser**
 - V této složce se nachází balíček *SoothsilverDtdParser*, který byl naprogramován jako součást práce a je v projektu využíván (viz sekce 3.1.1).
- **java-plugins**
 - V této složce se nachází zdrojové kódy Javy, ze kterých se sestavují pluginy pro XQuery a DOM/SAX. Skript *build.xml* toto sestavení automatizuje.
- **phptests**
 - V této složce jsou automatické testy ve frameworku PHPUnit. Jsou roztríděné do několika podsložek:
 - **fileio**: jednotkové testy, které přistupují k disku, a jejich vyhodnocení je tedy pomalé
 - **unit**: jednoduché jednotkové testy
 - **plugins**: testy pluginů, které kontrolují, zda pluginy hodnotí řešení studentů správně; tato složka je ještě rozdělena do podsložek

- Podložka *workspace* je určena jako aktuální adresář pro automatické testy pluginů.
- Testy spustíte tak, že dáte *PHPUnit* instrukci provést všechny testy ve složce *phptests* (nebo jen v některé podsložce), musíte ovšem označit podsložku *workspace* jako aktuální adresář, jinak nemusí některé testy fungovat (konkrétně *ConfigTest.php* na tomto závisí).
- **documentation-and-generator**
 - Zde jsou skripty pro generování dokumentace tříd.
- **similarity**
 - V této složce jsou zdrojové kódy Javy, ze kterých se sestavuje modul na kontrolu podobnosti domácích úkolů. Sestavení má na starosti stejný *build.xml* skript.
- **www**
 - Toto je hlavní složka projektu a obsahuje samotný systém XML Check. Na počítači, kde bude XML Check nainstalován pro použití studenty, by se měla vyskytovat pouze tato složka – ostatní nejsou třeba.
 - **core/**: serverová část aplikace naprogramovaná v PHP
 - **web/**: klientská část aplikace naprogramovaná v Javascriptu
 - **vendor/**: stažené knihovny třetích stran pomocí Composeru¹
 - **files/**: soubory nahrané uživatelem
 - **documents/**: šablony e-mailů, které systém posílá, a textů, které zobrazuje, viz sekce 5.5.2
 - **.htaccess**: tento soubor zabraňuje uživateli v přístupu k souboru *config.ini*, kde jsou uložena hesla do databáze, a ke složce *files/*, kde jsou řešení jiných uživatelů
 - **index.php**: hlavní stránka, která se zobrazuje uživateli
- **build.xml** (toto je skript pro Ant², který umožňuje snadnou kompilaci částí naprogramovaných v Javě a umístění výsledných archivů JAR na správné místo)

¹**Composer** je systém pro správu závislostí pro PHP; my jsme používali Composer verze 1.0-dev, který byl nejnovější v době psaní práce (jaro 2015); více informací lze najít na adrese <http://getcomposer.org/>.

²**Apache Ant** je nástroj pro správu sestavení, používá se převážně pro projekty v Javě; my jsme používali Ant verze 1.8.2; více informací lze najít na adrese <http://ant.apache.org/>.

6.3 Přístup k databázi

Jak je popsáno v sekci 3.2.7, přístup k databázi z PHP kódu byl změněn. Nyní se k databázi přistupuje pomocí ORM frameworku Doctrine2³.

Pro základní přehled o používání tohoto frameworku doporučujeme přečíst si úvodní instrukce v dokumentaci Doctrine2 [20] nebo se podívat do některých PHP skriptů v projektu, které s databází komunikují.

Tabulky jsou definovány entitami ve složce **www/core/doctrine**. K databázi se přistupuje vždy pomocí funkcí statické třídy **Repositories**. Před použitím databáze je třeba zavolat funkci **Config::init(...)**, aby se načetly přístupové údaje k databázi. To zpravidla činí již soubor **request.php**. Poté jsou v zásadě tři možnosti, jak k databázi přistoupit.

1. Nejjednodušší je vyhledat entitu pomocí ID:

```
1 $group = Repositories::findEntity(Repositories::Group, 2);  
2 echo "The group number 2 is called: " . $group->getName();
```

2. Někdy ovšem budete chtít získat celý repositář:

```
1 $groups = Repositories::getRepository(Repositories::Group)->findAll();  
2 // $groups is now an array of all groups.
```

3. Pro pokročilejší přístup k databázi budete potřebovat celý EntityManager:

```
1 $entityManager = Repositories::getEntityManager();
```

Třída **Repositories** nabízí ještě několik dalších funkcí pro usnadnění přístupu k databázi. Nejčastěji je používána funkce **persistAndFlush(\$entity)**, která uloží změny provedené na entitě zpět do databáze (nebo, pokud v databázi entita ještě není, vytvoří nový řádek v databázové tabulce).

6.4 Komunikace Javascriptu s PHP

V původní verzi byl projekt velmi závislý na souboru **.htaccess**, který zaručoval různá přesměrování v rámci aplikace. Aby tato přesměrování fungovala, musí být

³V době psaní této práce byla aktuální verze 2.4.

v souboru *.htaccess* přímo napsáno, v jaké podsložce *DocumentRoot* se soubor nachází. Není tedy možné přesunout projekt z jedné podsložky do jiné bez toho, že by se změnil soubor *.htaccess*.

Abychom umožnili snazší přesouvání projektu a abychom projekt zjednodušili, všechna přesměrování jsme ze souboru odstranili. Nyní slouží již jen k tomu, aby zabránil uživateli v přístupu ke konfiguračnímu souboru a k řešení ostatních studentů.

Uživateli se tedy nyní při zadání adresy aplikace do prohlížeče zobrazí přímo soubor *index.php* (dříve byl přesměrován na vnořený soubor *index.html*), který se na soubory Javascriptu, na styly CSS a na obrázky odkazuje do podsložky *web*.

Když potřebuje klientská část aplikace poslat požadavek serveru, např. aby získala seznam zadaných úkolů, pošle ho pomocí požadavku AJAX přímo na adresu **/core/request.php**. Většinou přitom použije metodu POST. Tento PHP skript vrací výsledek ve formě objektu JSON.

Na klientské straně se o komunikaci starají třídy **JsonCoreCommunicator** a **CoreCommunicator**. Samotné odesílání požadavku provádí knihovna třetí strany nazvaná **webtoolkit.Aim** [21].

6.5 Soothsilver DTD Parser

V sekci 3.1.1 popisujeme nový parser DTD naprogramovaný pro účely prvního domácího úkolu v Technologiích XML. V této sekci popíšeme, jak funguje uvnitř. Pojmy, které zde budeme používat, pochází ze specifikace XML [4].

Celé parsování probíhá jen jednou, při volání statické funkce `DTD::parseText(...)`. Tato funkce nejprve zavolá metodu `parseGlobalSpace` na interní podmnožině dokumentu XML a poté na text vlastního DTD dokumentu, protože specifikace XML určuje, že na interní podmnožinu se hledí, jako by předcházela externímu DTD souboru.

Tato funkce nejprve normalizuje konce řádků a odstraní komentáře. V budoucnu by komentáře neměly být odstraňovány, nýbrž parsovány, hlavně proto, aby chybové hlášky obsahovaly správná čísla řádků.

Hlavní smyčka parseru pak probíhá následovně:

1. Ukazatel se posune na následující nebílý znak.
2. Další postup se určí podle toho, jaký podřetězec začíná tímto znakem:
 - (a) Pokud je to `]]>`, ukončí se podmíněná sekce.
 - (b) Pokud je to `<![`, je třeba začít podmíněnou sekci. Abychom zjistili, zda se jedná o `INCLUDE` nebo `IGNORE` sekci, je nutné nejprve vyhodnotit parametrické entity těsně po tomto podřetězci, k čemuž slouží funkce *evaluatePEReferencesIn*.
 - (c) Jinak, pokud je ukazatel uvnitř podmíněné ignorované sekce, se znak ignoruje.
 - (d) Pokud je to `%[entita];`, pak se rekurzivně zavolá stejná funkce a vyhodnotí se obsah parametrické entity.
 - (e) Pokud je to `<!`, pak se najde odpovídající uzavírací špičatá závorka `>` a celý takto nalezený řetězec se předá funkci *parseMarkupDeclaration*. Tato funkce volá parsovací funkce pro jednotlivé druhy deklarácí.
 - (f) Pokud je to `<?`, pak se najde nejbližší další uzavírající špičatá závorka a celý takto nalezený řetězec se bude parsovat jako instrukce ke zpracování.
 - (g) Jinak bude ohlášena chyba.
3. Pokračuje se zpět od kroku 1, dokud není ukazatel na konci souboru.

Parser tedy dokument prochází od začátku do konce a nalezené deklarace přidává do vnitřních asociativních polí, které jsou po dokončení parsování uživateli k dispozici.

Uvnitř jednotlivých deklarácí se využívá metoda **tokenize**, která z řetězce vytvoří pole podřetězců, zpravidla oddělených mezerami. Na různých místech v DTD dokumentu je třeba rozlišovat různé tokeny, metoda *tokenize* tedy není úplně přímočará a v některých místech považuje za tokeny také například závorky, uvozovky, rouru nebo apostrofy.

Metody pro parsování jednotlivých druhů deklarácí pak typicky fungují tak, že svůj řetězec tokenizují pomocí metody *tokenize* a pak podle vlastní logiky kontrolují, zda tyto tokeny odpovídají specifikaci XML. Pokud tomuto tak je, pak výsledný objekt uloží do pole. Pokud ne, pak přidají chybu do pole vygenerovaných chyb, ale – pokud je to možné – pokouší se v parsování pokračovat, aby bylo odhyceno co největší množství chyb.

Problémové je vyhodnocování parametrických entit, neboť specifikace určuje, že na různých místech v dokumentu DTD se parametrické entity vyhodnocují jinak. Proto každá metoda na parsování volá funkci *evaluatePEReferencesIn* na jiné tokeny a popř. s jiným parametrem tak, aby byly reference vyhodnoceny korektně.

6.6 Odstraňování objektů z databáze

Uživatel může pomocí žádostí *DeleteUser*, *DeleteAssignment* a podobných smazat určité objekty z databáze. Dříve bylo toto provedeno tak, že např. žádost *DeleteLecture* zavolala statickou funkci *deleteLecture* ve třídě *RemovalManager*. Tato funkce, kromě toho, že smazala samotnou přednášku z databáze, také zavolala na všech závislých skupinách a problémech funkci *deleteGroup*, resp. *deleteProblem*. Tímto se kaskádovitě mazaly závislé objekty a udržela se tak konzistence databáze.

Mělo to ovšem problém: ačkoliv se samotná řešení studentů nemazala, smazaly se úkoly, ke kterým tato řešení příslušela, a tudíž nebylo možné zjistit, k jakému (smazanému) úkolu jaké řešení patří. To vadí, protože na těchto starých řešeních chceme provádět analýzy: konkrétně je chceme srovnávat na podobnost s novými řešeními kvůli možnosti opisování a také na nich chceme někdy spouštět testy korektnosti, abychom zjistili, jestli změny v pluginech nezpůsobily, že by stará korektní řešení přestala procházet na 100%. K tomu potřebujeme vědět, se kterým pluginem je řešení spojeno – v řetězu tedy musíme zachovat informaci o řešení, o úkolu, o problému a o pluginu.

Rozhodli jsme se proto nedovolit mazání žádných databázových objektů. Funkce třídy *RemovalManager* byly přejmenovány na *hideGroupAndItsAssignments* a podobně. Tyto funkce jen nastaví sloupec *deleted* příslušné tabulky na *true* a učiní to samé pro všechny závislé objekty. U tabulky *submissions* sloupec *deleted* není, používá se k těmto účelům sloupec *status* s obsahem *deleted*.

Při skrývání úkolu se nemažou příslušná řešení. Student tak může stále vidět svá řešení, i když už příslušný školní rok skončil.

K tomuto pravidlu o skrývání existují výjimky:

- **DeleteSubscription** skutečně maže informaci o tom, k jaké skupině je

uživatel přihlášen. Zde není třeba starý záznam v databázi uchovávat, protože historie přihlašování a odhlašování pro nás není zajímavá a tuto funkci využije student většinou pouze, pokud se překlikne.

- **DeleteUserType** smaže daný druh uživatele přímo a nastaví uživatelům, kteří byli tohoto druhu, druh „student“. Každý uživatel totiž musí mít nutně roli, aby měl vůbec nějaké pravomoci.
- **DeletePlugin** způsobí, že všechny problémy asociované s tímto pluginem nebudou nyní automaticky kontrolovány.
- **DeletePluginTest** smaže test pluginu a soubory s ním asociované.
- **DeleteAttachment**, **DeleteQuestion** a **DeleteTest** fungují jinak. Smazání přílohy ji zároveň odstraní ze všech otázek. Nelze smazat otázku, která je obsažena v nějakém testu. Jinak se tyto tři objekty mažou přímo z databáze. Není třeba o nich informace uchovávat.

6.7 Vícejazyčnost

Soubory `lang.js` (pro angličtinu) a `lang_cs.js` (pro češtinu) obsahují objekty, které přiřazují klíčům (vlastnostem objektu) anglická nebo česká slova (popř. věty). Při načtení aplikace se načte soubor `lang.js`, a pokud je jako jazyk zvolena čeština, tak se objekt v `lang_cs.js` sloučí s objektem z `lang.js` a přepíše jeho hodnoty. Takto, pokud nějaký výraz nebyl přeložen do češtiny, může aplikace stále fungovat, jen zde bude používat výraz anglický. Když poté některý soubor Javascriptu potřebuje zobrazit nějaký text uživateli, získá ho z tohoto sloučeného objektu.

Přestože se většina textů produkuje z Javascript kódu, některé texty jsou produkovány na straně serveru pomocí kódu PHP. Proto i PHP má vlastní mechanismus pro vícejazyčnost, který je celý v souboru `Language.php`. V tomto souboru je pro každý jazyk jedna funkce, která dostane jako argument číselný identifikátor textu (např. konstantu `StringID::InsufficientPrivileges`) a vrátí text v daném jazyce (např. „Nemáte dostatečná práva pro provedení této akce.“).

Nastavení, v jakém jazyce se má systém zobrazovat, je uloženo ve formě cookie na uživatelské počítači. To je zaprvé z důvodu, že podobná funkcionality – konkrétně výběr barevného stylu – se již jako cookie ukládá, a zadruhé z toho důvodu, že alternativou je jen ukládání vybraného jazyka v preferencích uživatele, ale v takovém případě by nebylo možné přeložit přihlašovací obrazovku, registrační formulář a formulář pro obnovu hesla, protože při práci s těmito prvky ještě uživatel není přihlášen.

Žádná knihovna třetí strany pro vícejazyčnost nebyla použita, ač jich existuje větší množství. Pro to existuje několik důvodů: zaprvé je významným cílem této práce zjednodušit systém, aby následné úpravy byly co nejjednodušší, a přidávání závislostí tomuto neprospívá; zadruhé, oproti knihovnám pro vícejazyčnost tento přístup dovoluje použití automatického doplňování kódu v integrovaných vývojových prostředích (IDE⁴); zatřetí, nenašli jsme žádnou knihovnu, která by dokázala dobře spolupracovat se systémem založeným na generování HTML z Javascriptu; začtvrté, nenašli jsme žádnou knihovnu, která by dokázala pracovat zároveň s PHP kódem a Javascript kódem.

6.8 Hlášení chyb

Původní systém hlášení chyb v části PHP byl celkem komplexní. Chyba se skládala z důležitosti (*level*), příčiny (*cause*), důsledku (*effect*) a dalších informací (*details*). Všechny tyto informace byly vypisovány do logu a zobrazovány uživateli.

Takové hlášení chyb se ovšem neukázalo v praxi jako zvláště užitečné. Log nebyl nikdy využit, neboť všechny hlášené chyby se daly reprodukovat a bylo možné je vyvolat znovu.

Některé chybové hlášky dále nevyžadují dělení na příčinu a důsledek. Pokud je například uživateli odmítnuto přihlášení, není třeba oddělovat informaci o příčině („tato kombinace jména a hesla není v databázi“) a o důsledku („nejste přihlášen“). Také z hlediska vícejazyčnosti je jednodušší přepisovat celé souvislé věty než oddělenou příčinu a důsledek.

⁴**Integrated Development Environment** (IDE) je počítačová aplikace usnadňující programování. Jejimi součástmi jsou typicky textový editor se zvýrazňováním syntaxe a napovídáním. *Visual Studio*, *Netbeans* a *IntelliJ IDEA* jsou příklady takových aplikací.

Proto většina chybových hlášek (resp. všechny, které jsou vícejazyčné) nyní nepoužívá rozdělení na příčinu, důsledek a další informace, nýbrž obsahuje jen informaci o důležitosti a samotnou chybovou hlášku. K tomu využívá existující třídy, chybová hláška je celá vložena do položky *cause*.

To mělo bohužel za následek také zjednodušení chybových hlášek, které jsou nyní méně podrobné. Například místo chybové hlášky „nepovedlo se získat z databáze seznam přednášek, a proto není zobrazen seznam odevzdaných řešení“ se nyní zobrazí spíše text ve smyslu „nepodařilo se připojit k databázi“.

6.9 Modul *similarity* (kontrola podobnosti)

Modul kontroly podobnosti je aplikace napsaná v Javě, která komunikuje přímo s databází a je volaná hlavní aplikací XML Check.

Všechny třídy modulu *similarity* se nacházejí ve jmenném prostoru *sooth*⁵.

6.9.1 Zdůvodnění volby programovacího jazyka

Pokud bychom napsali modul v PHP, odpadly by problémy s rozhraním mezi modulem a hlavní aplikací a nasazení nových verzí modulu by bylo jednodušší. Nicméně obzvláště během prvotní analýzy jsme potřebovali spouštět velké množství porovnávaní na všech datech, což vyžadovalo mnoho výpočetního času. Několik microbenchmarků ukázalo, že implementace v PHP by nejspíše byla několikrát pomalejší než implementace v Javě. Vícevláknové programování v PHP navíc není jednoduché.

Pro vyšší efektivitu bychom také mohli zvolit C# (kde by nám velmi pomohla možnost použití hodnotových typů v seznamech), nebo dokonce C++. Rozhraní mezi systémem XML Check a modulem by mohlo být stejné jako v Javě.

Přidali bychom tak do systému ale ještě další programovací jazyk. Javu už systém potřebuje kvůli druhému a čtvrtému domácímu úkolu. Pokud bychom modul psali v C# nebo C++, musel by potenciální další programátor, který bude chtít systém dále rozšířit, ovládat všechny tři jazyky. Vývojová prostředí pro Javu, PHP a C#/C++ jsou také poměrně odlišná.

⁵archaické anglické slovo odpovídající českému „pravda“

Z těchto důvodů jsme se rozhodli implementovat modul kontroly podobnosti v Javě.

6.9.2 Rozhraní mezi modulem similarity a hlavní aplikací

Všechny třídy modulu musí být zabaleny v souboru *similarity.jar*, který se musí nacházet v adresáři *core/* uvnitř hlavní aplikace. Hlavní aplikace kontrolu podobnosti spustí tak, že tento archiv vyvolá s parametrem *comparenew* na příkazové řádce. Musí přitom zajistit, že adresář *core/* bude pracovním adresářem, protože z něj modul bude načítat konfigurační soubor *config.ini*.

Vyvolání archivu spustí metodu *main* ve třídě *EntryPoint*. Ta přijímá více různých parametrů, ale ty slouží jen k ladění. Automaticky (v běžném provozu) bude systém používat pouze parametr *comparenew*. Pokud vyvoláte archiv bez parametrů, vypíše se na standartní výstup popis možných parametrů.

6.9.3 Pojmy

V databázi jsou uložena **řešení**, **dokumenty** a **podobnosti**. Jedno řešení se skládá z nuly nebo více dokumentů.

Dokumenty jsou uloženy ve zvláštní tabulce (*documents*) a reprezentují jednotlivé soubory v nějakém řešení. Dokumenty mají přiřazený typ, např. soubor s koncovkou *.xml* je dokumentem typu *PRIMARY_XML_FILE* a soubor s koncovkou *.xsd* je dokumentem typu *XSD_SCHEMA*.

6.9.4 Porovnání dvou řešení

Porovnání dvou řešení v databázi probíhá tak, že se párově porovnají všechny dokumenty jednoho řešení se všemi dokumenty druhého řešení, ovšem porovnávají se pouze dokumenty stejného typu. Ze všech těchto porovnání dokumentů se vybere ten pár dokumentů, který si je nejpodobnější, a jeho úroveň podobnosti se stane úrovní podobnosti těchto dvou řešení. Pokud je například úroveň podobnosti XML souborů v řešení X a v řešení Y nižší než podobnost XSD souborů v těchto řešeních, bude páru (X, Y) přiřazena úroveň podobnosti jejich XSD souborů. Tato trojice (X, Y, úroveň podobnosti) se uloží do databáze jako **podobnost** (v tabulce

similarities). V komentářích ve zdrojovém kódu se na tuto trojici odkazujeme jako na *similarity record* a v této dokumentaci jako na *záznam o podobnosti*.

6.9.5 Vývojový diagram

Kdykoliv student nahraje nové řešení, spustí se tento modul, který po řadě učiní následující kroky:

1. Pokud již jiná instance modulu běží, ukončí se.
2. Stáhni z databáze všechna řešení, která ještě nebyla analyzována.
3. Pokud žádné takové řešení není, ukončí se.
4. Pro tato řešení, nahraj do databázové tabulky *documents* informace o souborech, které se nachází v těchto řešeních.
5. Stáhni z databáze všechna starší řešení.
6. Porovnej všechna nová řešení se všemi staršími řešeními a zapiš výsledky do databáze.
7. Přepiš v databázové tabulce řešení informaci o výsledku analýzy.
8. Opakuj od 2.

V následujícím textu podrobně vysvětlíme jednotlivé kroky.

6.9.5.1 Aplikace může běžet jen jednou.

Protože aplikace stahuje všechna řešení z databáze, bude zabírat velké množství operační paměti. Proto je výhodné, aby neběžely dvě instance současně. K zabezpečení tohoto používáme zámek na souboru *similarity.lock*, který se odemkne, když je aplikace ukončena (i pokud je ukončena výjimkou).

6.9.5.2 Stáhni nová řešení z databáze.

Nestačí stáhnout jen nejnovější řešení, protože od posledního spuštění tohoto modulu mohli studenti do databáze nahrát řešení několik.

Nová řešení se identifikují pomocí stavu řešení v databázi, viz sekce 6.9.7.

6.9.5.3 Pokud žádné takové řešení není, ukonči se.

Při první iteraci cyklu se toto určitě nestane (protože program je spuštěn pouze v reakci na nahrání nového řešení), ovšem při druhé iteraci se zde cyklus zastaví, pokud během běhu první iterace nenahrál některý student další řešení.

6.9.5.4 Nahraj do databáze informace o souborech v nových řešeních.

Zde aplikace rozbalí studentem nahraný archiv ZIP a analyzuje jeho soubory. To probíhá v metodách třídy *DocumentExtractor*. Jednotlivé soubory jsou potom nahrány do databáze do tabulky *documents*. Archiv ZIP od této chvíle již nebude dále potřeba pro porovnávání řešení. Některé soubory v této fázi nahrány nejsou, a nebudou se tedy nikdy porovnávat (např. soubory s neznámou koncovkou nebo příliš velké soubory).

6.9.5.5 Stáhni z databáze všechna starší řešení.

V této fázi (ve třídě *Database*) se stáhnou a seřadí do vhodné datové struktury informace o všech řešeních a jejich dokumentech. Více informací o této datové struktuře podáváme v sekci 6.9.11.

6.9.5.6 Porovnej nová řešení se všemi starými.

Nejprve se vytvoří seznam „příkazů k porovnání“ (třída *SimilarityCommand*). Příkaz k porovnání je dvojice dvou řešení (starého a nového), která mají být porovnána. Vytvoří se tedy maximálně $(M + N) \times N$ dvojic, kde M je počet starých řešení a N je počet nových řešení.

Většinou bude počet dvojic ovšem mnohem menší, zaprvé proto, že se řešení vždy porovnává jen se staršími řešeními, ale hlavně proto, že se porovná pouze s řešeními vůči stejnému domácímu úkolu (řešení úkolu XML/DTD se tedy neporovnává s řešeními úkolu DOM/SAX). Rozhodli jsme se porovnávat podle jména pluginu (těch je šest) spíše než podle problému (kterých je dvanáct). Problémy jsou totiž určeny zvláště pro každou školu. Ač je nepravděpodobné, že by student z FEL opisoval od studenta z MFF, je to možné, a takto to systém dokáže zachytit.

Následně se porovnávání spustí v tolika vláknech, kolik má počítač k dispozici procesorů (třída *SimilarityCheckingBatch*). Pro práci ve více vláknech se používá

třída *Thread*, neboť několik pokusů ukázalo, že je to v našem případě rychlejší než použití novějších primitiv a tříd typu *Executor*.

Každé vlákno dostane stejný počet příkazů k porovnání a provede je. Nalezené podobnosti ukládají všechna vlákna do jediné instance třídy *SimilarityInsertionQueue*, která je *thread-safe*.

Nalezené podobnosti jsou po dávkách vkládány do databáze.

6.9.5.7 Přepiš informaci o stavu analýzy v tabulce řešení.

V této fázi se operuje s novým sloupečkem dat v tabulce *submissions*, který popisujeme v sekci 6.9.7.

Nejprve každé z nových řešení, která jsme zpracovávali, dostane přiřazeno stav *checked*, abychom je dokázali odlišit od řešení, která studenti nahráli, zatímco běžely předcházející fáze tohoto cyklu.

Poté jsou spuštěny dva MySQL příkazy, které změní tento stav buď na *guilty* (pokud byl vytvořen alespoň jeden záznam o podezřelé podobnosti) nebo na *innocent* (v opačném případě).

Poté se celý cyklus opakuje pro případ, že studenti mezitím nahráli další řešení.

6.9.6 Načtení konfigurace

Okamžitě po spuštění modul načte do statických vlastností třídy *Configuration* nastavení ze souboru *config.ini*. Potřebuje načíst především přístupové údaje k databázi, aby mohl načítat obsahy řešení studentů a nahrávat zpátky do databáze výsledky porovnání. Načtou se také některé další hodnoty, např. prahy pro označení podezřelosti. Více informací o možných nastaveních je v sekci 5.5.3.

6.9.7 Stav řešení v databázi

Do tabulky *submissions* jsme přidali nový sloupec nazvaný *similarityStatus*, který může nabývat těchto hodnot:

- *new*: Toto řešení systém právě vložil do databáze a ještě neprošlo kontrolou podobnosti.

- *checked*: Modul kontroly podobnosti právě vložil do databáze záznamy o podobnosti, ale ještě nepřepsal stav na *guilty* nebo *innocent*. To udělá během několika následujících vteřin.
- *guilty*: Pro toto řešení v databázi existuje alespoň jeden záznam o podezřelé podobnosti.
- *innocent*: Pro toto řešení v databázi žádný takový záznam neexistuje.

Rozlišení *checked/guilty/innocent* se může zdát zbytečné, protože tyto informace již v databázi jsou. Jejich rozlišení však umožní zobrazovat cvičícím upozornění na podezření již na stránce s řešeními (viz soubor *GetTeacherSubmissions.php*).

6.9.8 Automatické testy (*sooth.unittests*)

Modul obsahuje několik automatických testů v balíčku *sooth.unittests*. Testují korektnost naimplementovaných algoritmů. Většina ostatních funkcí provádí operace s databází a testovala by se tedy jen obtížně. Přesto je určitě možné – a vhodné – implementovat ještě nějaké další automatické testy.

6.9.9 Algoritmy (*sooth.similarity*)

Jsou implementovány čtyři algoritmy: identita, Levenshteinova vzdálenost, Greedy-String-Tiling a editační vzdálenost Zhang-Shasha. Třídy implementující tyto algoritmy jsou v balíčku *sooth.similarity* a obsahují metodu *compare*, která porovná dva dokumenty a vrátí výsledek algoritmu. Identita tak vrací pravdivostní hodnotu, zatímco ostatní algoritmy vrací celé číslo.

Rozhodli jsme se oddělit rozhodování o procentuální podobnosti od samotných algoritmů kvůli jednoduššímu testování.

Všechny tyto třídy umožňují vytvářet instance, a to proto, aby se algoritmy daly paralelizovat. Algoritmus *Greedy-String-Tiling* by sice mohl být statický, protože si žádá lokální data nepřechovává, ale Levenshteinův algoritmus i algoritmus Zhang-Shasha alokují velké množství paměti. Tuto paměť je možné použít v několika porovnáních za sebou bez jejího explicitního přemazání. Proto jsou proměnné v této paměti instančními proměnnými třídy algoritmu, a nikoliv lokálními proměnnými příslušné metody *compare*.

Pro snadné použití jsme implementovali u každé takové třídy statickou metodu *getInstance*, která využívá proměnnou typu *ThreadLocal* a zařizuje vytváření instance třídy sama.

6.9.10 Přístup k databázi a knihovna jOOQ

Pro přístup k databázi využíváme knihovnu jOOQ⁶. Pro tuto knihovnu jsme se rozhodli, protože jako jedna z mála považuje za primární zdroj struktury databázi a nikoliv kód Javy. Struktura databáze je již generována pomocí Doctrine2 z kódu PHP a nemůžeme ji generovat ze dvou různých zdrojů.

Čistý SQL přístup k databázi ovšem také nechceme použít, protože použití jOOQ a vygenerovaných tříd je jednodušší a méně náchylné k programátorským chybám.

6.9.11 SubmissionsByPlugin

Pro porovnávání je třeba stáhnout z databáze všechna řešení spolu s informacemi o pluginu, který je hodnotí, a s informacemi o dokumentech, které řešení obsahuje. Tyto informace jsou ovšem rozděleny ve třech různých tabulkách. Pro jejich načtení do paměti používáme datovou strukturu *SubmissionsByPlugin*, kterou naplňujeme pomocí tohoto SQL dotazu:

```
1 SELECT d.name AS dname,  
2 d.text AS dtext,  
3 d.type AS dtype,  
4 s.userId AS suser,  
5 s.id AS sid  
6 plg.identifier AS plgIdentifier,  
7 s.date AS sdate  
8 FROM documents AS d  
9 INNER JOIN submissions AS s ON d.submissionId = s.id  
10 INNER JOIN assignments AS a ON s.assignmentId = a.id  
11 INNER JOIN problems AS p ON a.problemId = p.id  
12 INNER JOIN plugins AS plg ON p.pluginId = plg.id  
13 WHERE s.status <> 'deleted'  
14 ORDER BY plg.identifier, s.date, s.id, d.type
```

Obzvláště důležitá je klauzule *ORDER BY*, která nám umožní výsledky procházet sekvenčně, a přitom tvořit seznam utříděný podle pluginů a chronologicky.

⁶Java Object Oriented Querying. Používáme verzi 3.5. <http://www.jooq.org/>

Dotaz vrací dokumenty, které vkládáme v kódu do seznamu. Jakmile dotaz vrátí dokument s jiným ID řešení, víme, že předcházející řešení je kompletně zpracované a již žádné další dokumenty neobsahuje.

Podrobnosti jsou v komentářích třídy *SubmissionsByPlugin*.

6.9.12 Zdůvodnění použití statických tříd a metod

V tomto modulu používáme velké množství statických metod. K tomu existuje několik důvodů:

- Některé metody operují nad řetězci nebo třídami v *Java Class Library*, které nelze rozšířit.
- Některé metody operují nad automaticky vygenerovanými třídami jOOQ, které rozšiřovat nechceme, protože se generují z databázové struktury, a změny jsou tedy přemazávány.
- Mnohé operace transformují jeden objekt na jiný, např. vytváří dokumenty z archivu ZIP nebo výsledek porovnání z páru dokumentů. Kdyby byly tyto operace instanční metody, nemuselo by být jasné, ve které třídě je hledat. Proto jsme je seskupili podle jejich funkce a nikoliv podle objektů, nad kterými operují.

Máme tedy statické funkce seskupené podle oblastí:

- Třída **ComparisonFunctions** obsahuje funkce na porovnávání dvou řešení nebo dokumentů.
- Třída **BatchActions** obsahuje funkce, které pracují nad všemi daty v databázi.
- Třída **DocumentExtractor** obsahuje funkce, které zpracovávají soubory ZIP.
- Třída **PreprocessingUtilities** obsahuje funkce, které předzpracovávají text dokumentů.

6.9.13 Vyžadovaná operační paměť

Aby bylo porovnávání co možná nejrychlejší, jsou všechna starší řešení stažena z databáze do operační paměti. To může být problém, pokud velikost řešení v databázi přesáhne velikost operační paměti.

V době psaní této práce byl systém XML Check používán po čtyři roky výuky a modul *similarity* během kontroly podobnosti zabírá 600 MB operační paměti. Počet řešení roste lineárně. Můžeme tedy odhadnout, že velikost dat se bude zvětšovat o 150 MB za rok. To by znamenalo, že za 10 let zabraná operační paměť přesáhne 2 GB, a za 16 let 3 GB. Tolik paměti systém nyní k dispozici má.

Pokud by byl systém XML Check v roce 2030 stále používán, bylo by možná vhodné ho upravit, aby nahrával jen část databáze, porovnal nová řešení s touto částí, a toto opakoval pro další části databáze, dokud se neprojde databáze celá.

6.10 Dokumentace některých dalších změn

- **Anglické pojmenování pro oznámkování řešení bylo přejmenováno z *rate* na *grade*.** Slovo *grade* lépe odpovídá významu oznámkování učitelem. Na některých místech v kódu (nikoliv v rozhraní) ovšem zůstává slovo *rate* (protože by bylo příliš náročné takový kód refaktorovat).
- **Systém byl přejmenován na *XML Check*.** Dříve byl rozlišován název *Assignment Manager*, který označoval samotnou aplikaci, a název *xmlcheck*, který označoval počítač, na kterém běžela instance této aplikace pro předmět Technologie XML. Učitelé i studenti se ovšem na systém odkazovali jako na *xmlcheck* a systém dosud není využíván nikde mimo tento předmět.

7. Pohled zpět a do budoucna

V této kapitole nejprve činíme několik pozorování, ke kterým jsme došli v průběhu práce nebo na jejím konci, a poté navrhuje několik způsobů, jak by bylo možné systém XML Check dále vylepšit.

7.1 Zpětný pohled

Měli jsme příležitost pozorovat reakce studentů a cvičících na změněný systém v průběhu dvou semestrů. Celkově považujeme projekt rozšíření systému XML Check za úspěšný, některých chyb jsme se ovšem mohli vyvarovat, a naopak některá rozhodnutí nám výrazně ulehčila práci. Museli jsme se také vypořádat s některými rozhodnutími z původního projektu.

7.1.1 Upravovat aplikaci během živého provozu je těžké.

Obzvláště během prvního semestru práce na systému, kdy systém již byl plně v provozu, jsme věnovali velké úsilí tomu, abychom zabránili vzniku nových chyb. Proces nahrávání nové verze na server jsme také neměli promyšlený, a někdy tak vznikaly chyby, které se na vývojovém počítači neprojevíly.

Nutnost nenahrávat nové verze programu v době před termíny odevzdání byla také velmi omezující. Termíny odevzdání totiž nejsou napříč cvičeními synchronizované a zdálo se, že téměř každý den je nevhodný k nahrání nové verze.

7.1.2 Nový DTD Parser není dobře použitelný jinde.

Vytvořený parser DTD souborů je sice obecný a dá se použít jako samostatný balíček, jeho funkcionality ovšem nejspíše nebude užitečná pro nikoho, kdo nemá velmi podobné požadavky jako XML Check. Soubory DTD totiž není většinou třeba parsovat nezávisle na XML datech, a většině uživatelů bude vyhovovat použití výchozích knihoven PHP.

Možná jsme nemuseli parser programovat jako samostatný publikovatelný balíček a místo toho naprogramovat jen funkce nezbytné pro kontrolu domácích

úkolů v systému XML Check.

7.1.3 Automatické testy pluginů jsou velmi užitečné.

Zavedení automatických testů kontrolujících pluginů odhalilo velké množství chyb dříve, než byly aktualizované pluginy nahrány na server. Některé chyby by bez těchto testů zůstaly dokonce neodhalené.

7.1.4 Úprava uživatelského rozhraní není snadná.

V návrhu původní aplikace je vystavěno celé uživatelské prostředí na tabulkách a formulářích. V původní verzi projektu skutečně nebyly nutně třeba žádné jiné prvky. Přesto je tento systém velmi omezující. Obzvláště při programování stránek pro kontrolu podobnosti by se hodilo použít jiné prvky než dynamické formuláře a tabulky původní verze.

Přidání nových prvků (mimo tabulek a formulářů) ovšem není snadné a komplikací je zde i fakt, že uživatelské prostředí není podrobně dokumentováno. Následkem toho jsme použili (modifikované) dynamické formuláře a tabulky, což se však negativně projevuje na přívětivosti rozhraní.

7.2 Možné pokračování

V této sekci popíšeme možné způsoby, jak by bylo možné systém XML Check dále vylepšit.

7.2.1 Hybridní uložení jazykového nastavení

V současnosti se jazykové nastavení (čeština nebo angličtina) ukládá v souboru cookie na uživatelské počítači.

Kdyby se místo toho ukládalo v uživatelském záznamu v databázi spolu s celým jménem a e-mailem, mělo by to několik výhod, konkrétně, že by bylo jazykové nastavení asociováno s uživatelem spíše než s počítačem a že by mohly být lokalizovány do uživatelského jazyka e-maily jemu posílané. Na druhou stranu by pak

nebyla lokalizována část před přihlášením, tj. přihlašovací obrazovka a registrační formulář.

Bylo by ale možné použít hybridní přístup: pomocí souboru cookie určit jazyk pro přihlašovací obrazovku a pomocí nastavení v databázi určit jazyk uvnitř aplikace po přihlášení.

7.2.2 Smazání starých uživatelů

Jak bylo uvedeno v sekci 3.2.9, celkem 591 uživatelů má nyní v systému heslo chráněné jen slabým hashovacím algoritmem MD5. Celkem 112 z těchto uživatelů sdílí své heslo alespoň s jedním dalším uživatelem, dvě hesla jsou dokonce používána každé pěti různými uživateli.

Možná by bylo vhodné smazat všechna tato hesla. To by způsobilo, že by se stávající uživatelé nedokázali do systému přihlásit. K opětovnému přihlášení by museli využít funkci pro obnovení zapomenutého hesla.

Výhodou by pak bylo, že v případě přístupu útočnicka k databázovému serveru by tento nebyl schopen zjistit hesla žádného uživatele.

7.2.3 Silnější bezpečnost aplikace

Ač jsme se bezpečností v této práci také zabývali, vykazuje aplikace stále některé bezpečnostní mezery. V tuto chvíli se spoléháme na to, že se uživatelé nebudou snažit úmyslně systém poškozovat. To nemusí být úplně správný předpoklad, vzhledem k tomu, že aplikace je veřejně přístupná z internetu a odkaz na ní je zveřejněn na stránkách předmětu.

System aktuálně běží ve virtuálním počítači na serverech MFF. Narušením systému by tedy útočnick mohl v nejhorším případě získat informace o všech řešeních domácích úkolů všech studentů, mohl by získat přístup k jejich e-mailovým adresám a zahashovaným heslům, změnit hodnocení cvičících nebo nějaké domácí úkoly smazat. Jiné fakultní servery by ovšem neovlivnil.

V současné době jsme si vědomi tří bezpečnostních chyb, kterými může uživatel server se systémem zastavit tak, že zahltí jeho pevný disk nebo procesor.

7.2.4 Náhled nahraných řešení přímo v prohlížeči

Tadeáš Palusga popisuje ve své analýze systému XML Check v [2, str. 8-9] ten nedostatek, že systém cvičícímu nedovoluje přímo ve webovém rozhraní prohlédnout obsah souborů nahraných studentem. Místo toho musí cvičící stáhnout archiv ZIP, rozbalit ho na svém počítači a prohlédnout si ho. Tím se však zvyšuje počet úkonů, které musí cvičící pro každé hodnocené řešení provést, čímž je zdržován.

Nemuselo by být zvláště obtížné přidat možnost zobrazení adresářové struktury nahraného souboru ZIP přímo v prohlížeči. Cvičící by tam pak mohl vybrat soubor, který by se taktéž v prohlížeči (se zvýrazňováním syntaxe) zobrazil. Tuto možnost by cvičící předmětu uvítali.

7.2.5 Umožnění používání systému v jiných předmětech

Systém XML Check se v předmětu Technologie XML osvědčil a usnadňuje práci studentům i cvičícím. Ovšem i vyučující jiných předmětů by mohli ocenit XML Check, protože jim může usnadnit práci s poloautomatickou kontrolou.

Palusga v [2] ukázal, že je možné alespoň pro první úkol z předmětu *Databázové systémy* na FEL vytvořit poloautomatickou kontrolu, na použití v tomto předmětu by tedy systém mohl být rozšířen. Na MFF by systém možná mohl najít uplatnění např. v předmětu *Principy překladačů*, kde je možné téměř automatickou kontrolu provést také. Převod domácích úkolů tohoto předmětu do systému XML Check by mohl ulehčit práci nejen cvičícím, ale také by dal studentům rychlejší zpětnou vazbu a umožnil by jim vyvarovat se zbytečných chyb.

7.2.6 Vyzkoušení dalších metod kontroly podobnosti

Neotestovali jsme zdaleka všechny metody kontroly podobnosti. V [3] lze najít několik dalších postupů. I pro postupy, které jsme zkusili, je možné změnit nastavení parametrů. Je možné, že bychom tak získali kvalitnější výsledky.

7.2.7 Shlukování stejných domácích úkolů

Na straně 55 popisujeme, že v několika případech vznikl „shluk“ studentů, kteří odevzdali téměř stejná řešení. Současný systém pro takový shluk vygeneruje velké

množství záznamů o podezřelé podobnosti a může být těžké se v nich vyznat.

Možná by proto bylo výhodné sloučit tyto domácí úkoly do jedné skupiny, a pokud se najde další domácí úkol, který do ní patří, cvičícímu by se mohla zobrazit přímo informace ve stylu „*Toto řešení je podobné této skupině čítající 7 jiných řešení.*“ místo sedmi jednotlivých porovnání.

K tomu by se dala použít metoda *document clustering*, popsaná například v [13, str. 349].

7.2.8 Vylepšení dokumentace tříd Doxygenu

Doxygen nyní generuje neúplný výstup. To má dvě příčiny.

Zprv je znak pro uvození klíčového slova doxygenu (zpětné lomítko) stejný jako znak pro uvození jmenného prostoru PHP, a Doxygen tak někdy mezi klíčovými slovy a jmennými prostory nedokáže rozlišit.

Zadruhé používáme knihovnu *Base.js*, čímž do Javascriptu přidáváme třídy a dědičnost, které tento jazyk obvykle nemá, a Doxygen s nimi neumí pracovat.

Tyto nedostatky by snad bylo možné odstranit lepšími skripty pro preprocessing zdrojového kódu před zpracováním Doxygenem.

7.2.9 Zrychlení kontroly podobnosti

Algoritmus *Greedy-String-Tiling* by možná poskytoval dobré výsledky, ale jeho současná implementace je nepoužitelně pomalá. Aplikace optimalizací popsanych v [16] by z algoritmu možná udělala použitelnou metodu.

7.2.10 Upgrade na vyšší verze jQuery a jQuery UI

Nové verze řeší některé chyby a zvyšují rychlost. Upgrade ovšem není jednoduchý, protože kód systému nějakým způsobem závisí na starších verzích jQuery. Autorovi této práce se nepodařilo upgrade provést.

Závěr

Naším cílem bylo opravit chyby v systému XML Check, přidat nové funkce žádané studenty a cvičícími a implementovat kontrolu podobnosti domácích úkolů. Všechny cíle se nám podařilo splnit.

Všechny nahlášené chyby byly opraveny. V letním semestru 2015, kdy byl plně upravený systém použit v předmětu poprvé, byly objeveny další menší chyby, které ovšem nezabraňovaly použití systému, a byly opraveny.

Přidali jsme do systému 27 funkcí navržených cvičícími nebo studenty. Další navržené funkce jsme ale zavrhlí, protože byly nad rámec práce nebo by systému neprospěly.

Implementovali jsme tři různé algoritmy pro srovnávání domácích úkolů – Levenshteinovu vzdálenost, algoritmus Greedy-String-Tiling a editační vzdálenost stromů Zhang-Shasha – a srovnali jsme je. Levenshteinova vzdálenost a editační vzdálenost Zhang-Shasha přinášely dobré výsledky, a proto se nyní aplikují na všechna nová řešení domácích úkolů. Během analýzy jsme objevili 14 opsaných domácích úkolů. Žádné z nich nebyly identické – vždy se studenti snažili opisování nějak zakrýt.

Upravený systém nyní funguje stabilně a je v plánu pokračovat v jeho používání v předmětu Technologie XML v dalších letech.

Seznam použité literatury

- [1] Jan Konopásek. Homework and Test Management System. Praha 2011.
- [2] Tadeáš Palusga. Systém pro kontrolu domácích úkolů a plagiátorství. Praha 2012.
- [3] Irena Mlýnková a Jaroslav Pokorný. Exploitation of Similarity and Pattern Matching in XML Technologies. Praha 2006.
- [4] W3C. Extensible Markup Language (XML) 1.0 (Fifth Edition). [Online, citace 26. 1. 2015] <http://www.w3.org/TR/REC-xml/>
- [5] XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999. [Online, citace 4. 2. 2015] <http://www.w3.org/TR/xpath/>
- [6] XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 November 1999. [Online, citace 4. 2. 2015] <http://www.w3.org/TR/xslt/>
- [7] Document Object Model (DOM) Technical Reports. [Online, citace 4. 2. 2015] <http://www.w3.org/DOM/DOMTR>
- [8] SAX. [Online, citace 4. 2. 2015] <http://www.saxproject.org/>
- [9] XQuery 1.0: An XML Query Language (Second Edition). W3C Recommendation 14 December 2010. [Online, citace 4. 2. 2015] <http://www.w3.org/TR/xquery/>
- [10] Bray, Tim. The Annotated XML Specification. [Online, datum vytvoření 2. 10. 1998, citace 29. 11. 2014] <http://www.xml.com/axml/testaxml.htm>
- [11] Ian S. Graham a Liam Quin. XML Specification Guide. John Wiley & Sons, Inc. 1999. ISBN 0-471-32753-0.
- [12] Irena Mlýnková a další. XML technologie - Principy a aplikace v praxi. Grada Publishing a. s. Praha 2008. ISBN 978-80-247-2725-7.
- [13] Christopher Manning, Prabhakar Raghavan a Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008. ISBN 0521865719.
- [14] XMLUnit - Unit Testing XML for Java and .NET. [Online, citace 4. 2. 2015] <http://www.xmlunit.org/>
- [15] PHP: Password Hashing. [Online, citace 4. 2. 2015] <http://php.net/manual/en/faq.passwords.php>
- [16] Michael J. Wise. String Similarity via Greedy String Tiling and Running Karp-Rabin Matching. University of Sydney, Austrálie. 1993.
- [17] Lutz Prechelt, Guido Malpohl a Michael Philippsen. JPlag: Finding plagiarisms among a set of programs. Technical Report. Karlsruhe 2000.
- [18] Conni Müller. Analyse verschiedener Werkzeuge zur automatisierten Plagiatsprüfung von Programmtexten. Bachelorarbeit. München 2013.
- [19] Kaizhong Zhang a Dennis Shasha. SimpleFast Algorithms for the Editing Distance between Trees and Related Problems. Society for Industrial and Applied Mathematics 1989.
- [20] Doctrine Project. [Online, citace 26. 1. 2015] <http://www.doctrine-project.org/>

- [21] webtoolkit.info [Online, citace 26. 1. 2015]
<http://www.webtoolkit.info/ajax-file-upload.html>
- [22] The Apache Software Foundation. Apache HTTP ServerProject. [Online, citace 29. 11. 2014] <http://httpd.apache.org/>
- [23] Oracle Corporation. MySQL. [Online, citace 29. 11. 2014]
<http://www.mysql.com/>
- [24] The PHP Group. PHP: Hypertext Preprocessor. [Online, citace 29. 11. 2014]
<http://php.net/>
- [25] Oracle Corporation. Java Software. [Online, citace 29. 11. 2014]
<https://www.oracle.com/java/index.html>
- [26] Jiří Kosek. PHP a XML. Grada Publishing a. s. Praha 2009. ISBN 978-80-247-1116-4.
- [27] CodEx - The Code Examiner. [Online, citace 4. 2. 2015]
<http://codex2.ms.mff.cuni.cz/project/>
- [28] ERUDIO s. r. o. Kde jsou instalace IS Studium. [Online, citace 15. 11. 2014]
<http://www.erudio.cz/?stranka=sw.reference>
- [29] Erudio s. r. o. IS STUDIUM. [Online, citace 4. 2. 2015]
<http://www.erudio.cz/?stranka=sw.isstudium>
- [30] Katedra distribuovaných a spolehlivých systémů. Doporučené postupy v programování. [Online, citace 27. 11. 2014]
http://d3s.mff.cuni.cz/teaching/programming_practices
- [31] José Paulo Leal. Mooshak. [Online, citace 28. 11. 2014]
<https://mooshak.dcc.fc.up.pt/>
- [32] Pietro Longo, Andrea Sterbini a Marco Temperini. TSW: A Web-Based Automatic Correction System for C Programming Exercises. Řím 2009.
- [33] The Web-CAT Community. [Online, citace 28. 11. 2014]
<http://web-cat.org/>
- [34] Moodle - Open-source learning platform. [Online, citace 28. 11. 2014]
<https://moodle.org/>
- [35] Juan Carlos Rodriguez-del-Pino. VPL - Virtual Programming Lab. [Online, citace 28. 11. 2014] <http://vpl.dis.ulpgc.es/index.php/en/>
- [36] Blackboard Inc. Blackboard Learn. [Online, citace 29. 11. 2014]
<http://uki.blackboard.com/>
- [37] Shehryar Nabi. 7 Blackboard competitors with online learning solutions. [Online, citace 29. 11. 2014] <http://www.educationdive.com/news/7-blackboard-competitors-with-online-learning-solutions/35847/>
- [38] Odevzdávání studentských prací. [UploadSystem, online, citace 4. 2. 2015]
<http://http://cw.felk.cvut.cz/upload/>
- [39] Jakub Čapský. Systém pro ověřování plagiátorství prostřednictvím vyhledávací služby Google. Diplomová práce. Pardubice 2010.
- [40] Plagiarism Detection Software | iThenticate. [Online, citace 4. 2. 2015]
<http://www.ithenticate.com/>

- [41] Skyline, LLC. What is Plagiarism Detector? [Online, citace 30. 11. 2014]
[http:
//www.plagiarism-detector.com/what-is-plagiarism-detector.php](http://www.plagiarism-detector.com/what-is-plagiarism-detector.php)
- [42] Masarykova univerzita. Theses.cz. [Online, citace 30. 11. 2014]
<http://www.theses.cz>

Seznam tabulek

1	Výsledky analýzy podobnosti nad odevzdanými úkoly	56
2	Dotazy XPath na ukázkovém dokumentu	130
3	Dotazy XQuery na ukázkovém dokumentu	131

Seznam obrázků

1	Aplikace Grupíček	13
2	Formulář pro odevzdání úkolu v systému pro Doporučené postupy v programování	16
3	Registrační formulář	59
4	Formulář pro zadání aktivačního kódu	60
5	Přihlašovací obrazovka	60
6	Žádost o obnovu hesla	61
7	Formulář pro určení nového hesla	61
8	Změna jazyka z přihlašovací obrazovky	62
9	Změna jazyka po přihlášení	62
10	Hlavní obrazovka systému XML Check	63
11	Hlavní nastavení	63
12	Nastavení e-mailových notifikací	64
13	Nastavení uživatelského rozhraní	65
14	Typická tabulka systému XML Check	65
15	Formulář na vytvoření nového filtru	67
16	Skupiny, ke kterým se lze připojit	68
17	Formulář k odevzdání řešení	69
18	Neoznámkovaná řešení (z pohledu studenta)	70
19	Formulář pro vytvoření nebo úpravu skupiny	71
20	Formulář pro vytvoření nebo úpravu úkolu	72
21	Seznam neoznámkovaných řešení	72
22	Formulář pro přidělení počtu bodů	73
23	Informace o podobnosti s jinými řešeními	74
24	Průběžné hodnocení všech studentů	74
25	Seznam přednášek	75
26	Formulář pro vytvoření nebo úpravu problému	76
27	Formulář pro nahrání pluginu	76
28	Seznam všech uživatelů	77
29	Formulář pro vytvoření nebo úpravu uživatelského účtu	77
30	Seznam všech typů uživatelů	82
31	Formulář pro vytvoření nebo úpravu typu uživatele	82
32	Příklad nesrozumitelné chybové hlášky	84
33	Strom DOM ukázkového dokumentu	129

Seznam použitých zkratek

- **MD5**. Message-digest algorithm 5.
- **SHA-1**. Secure hash algorithm 1.
- **PDF**. Portable Document Format.
- **INI**. Inicializační soubor Windows.
- **libxml2**. XML library 2.
- **jOOQ**. Java Object Oriented Querying.
- **PHPass**. PHP Password.
- **EXE**. Koncovka spustitelného souboru ve Windows.
- **TXT**. Koncovka textového souboru ve Windows.
- **FEL**. Fakulta elektrotechnická Českého vysokého učení technického v Praze.
- **MFF**. Matematicko-fyzikální fakulta Univerzity Karlovy v Praze.
- **ČVUT**. České vysoké učení technické v Praze.
- **UK**. Univerzita Karlova v Praze.
- **W3C**. World Wide Web Consortium.
- **FIT**. Fakulta informačních technologií Českého vysokého učení technického v Praze.
- **MML**. Minimum Match Length.
- **CDATA**. Character Data.
- **ID**. Identifikátor.
- **IDREF**. Reference na identifikátor.
- **ORM**. Object-relational mapper.
- **RAM**. Random Access Memory.
- **DNS**. Domain Name System.
- **IP**. Internet Protocol.
- **SSL**. Secure Sockets Layer.
- **TLS**. Transport Layer Security.
- **SMTP**. Simple Message Transfer Protocol.
- **AJAX**. Asynchronous Javascript and XML.
- **IDE**. Integrated Development Environment.
- **DOCTYPE**. Document type.
- **FLWOR**. From, let, where, order by, return.
- **CodeEx**. The Code Examiner.
- **IS Studium**. Informační systém Studium.
- **VPL**. Virtual Programming Lab.
- **HWChecker**. Homework Checker.
- **MoSS**. Measure of Software Similarity.
- **JPlag**. Plagiarism in Java.
- **AsM**. Assignment manager.
- **SIS**. Studijní informační systém.
- **KOS**. Komponenta Studium.
- **XML**. Extensible Markup Language.

- **DTD.** Document Type Definition.
- **XSLT.** Extensible Stylesheet Language Transformations.
- **XSD.** XML Schema Definition.
- **DOM.** Document Object Model.
- **SAX.** Simple API for XML.
- **PHP.** PHP: Hypertext Preprocessor.
- **CSS.** Cascading Style Sheets.
- **HTML.** Hypertext Markup Language.
- **HTTP.** Hypertext Transfer Protocol.
- **SQL.** Structured Query Language.
- **PEAR.** PHP Extension and Application Repository.
- **UTF.** Universal Character Set Transformation Format.
- **JRE.** Java Runtime Environment.
- **JDK.** Java Development Kit.
- **JSON.** Javascript Object Notation.

Přílohy

A Obsah přiloženého CD

- Archiv ZIP s instalací programu (složka *Instalační balíček*)
- Text bakalářské práce (soubor *xmlcheck-thesis.pdf*)
- Vygenerovaná programátorská dokumentace tříd (složka *Dokumentace tříd Doxygen*)
- Uživatelská dokumentace ve formě PDF souboru (soubor *xmlcheck-user-documentation.pdf*)
- Statistika kontroly podobnosti (složka *Statistiky z kontroly podobnosti*)
- Zdrojový kód jako projekt IntelliJ IDEA (složka *Zdrojový kód*)

B Podrobné zadání domácích úkolů

B.1 XML a DTD

Vaše řešení musí obsahovat právě jeden .xml soubor a právě jeden .dtd soubor. Oba soubory musí být v UTF-8 bez Byte Order Mark.

Používejte XML verze 1.0.

Deklarace DOCTYPE souboru XML se musí odkazovat na .dtd soubor v řešení. XML i DTD soubor musí být správně zformované a validní.

DTD musí obsahovat:

1. Prázdné elementy EMPTY.
2. Čistě textové elementy (#PCDATA).
3. Elementy s modelem (tedy non-mixed, non-text)
4. „mixed“ elementy (tedy s textem i vnitřními elementy)
5. Deklaraci entity
6. Deklaraci ATTLIST
7. Deklaraci atributu #REQUIRED
8. Deklaraci atributu CDATA
9. Deklaraci atributu výčtového typu
10. Deklaraci atributu ID
11. Deklaraci atributu IDREF

XML musí použít každou funkcionalitu, která je vyžadována 11 body v předešlém odstavci, např. tedy musí obsahovat čistě textový element, různé druhy atributů, ID atribut a IDREF atribut. XML dokument musí mít hloubku alespoň 5 („depth“). Alespoň jeden element musí obsahovat alespoň 10 synů („fan-out“).

Dohromady musí být v XML a DTD souboru alespoň jedno použití vlastní nadefinované entity (obecné nebo parametrické), jedna instrukce ke zpracování, jedna sekce CDATA a jeden komentář.

B.2 DOM/SAX

Vaše třída MyDomTransformer musí mít metodu `public void transform(org.w3c.dom.Document)`, která pozmění XML dokument v parametru.

Vaše třída `MySaxHandler` musí dědit od `org.xml.sax.helpers.DefaultHandler` a musí mít výchozí bezparametrický konstruktor. Její instance bude automaticky vytvořena a předána SAX parseru.

Vaše úkoly:

1. `MyDomTransformer` provede 2 úpravy XML dokumentu - např. přidání nového zaměstnance se všemi parametry, smazání všech zaměstnanců s platem menším než X, seřazení zaměstnanců podle abecedy, zvýšení platu určitých zaměstnanců apod.
2. `MySaxHandler` spočte a vypíše na konzoli tři zvolené charakteristiky vašich dat, a to tak, že:
 - (a) Jedna se bude týkat hodnot atributů (např. spočte průměrnou váhu výrobku)
 - (b) Jedna se bude týkat obsahu elementů (např. najde tři nejčastější příjmení zaměstnanců)
 - (c) Jedna bude využívat kontext (např. počet zaměstnanců, kteří jsou starší 60 let, ale nemají dovolenou) (Systém automaticky nekontroluje, ale snažte se, aby vaše řešení bylo minimálně tak složité, jako uvedené příklady.)

Nahrajte jen soubory relevantní k domácímu úkolu! Nenahrávejte celé projekty NetBeans/Eclipse, protože obsahují mnoho různých souborů XML a systém pak může vaše řešení mylně označit za podezřelé z opisování.

B.3 XPath

Vaše řešení musí obsahovat jeden validní XML soubor a pět legálních XPath dotazů, volitelně také DTD soubor.

Používejte jen XPath 1.0.

Každý XPath dotaz opatřete komentářem uzavřeným do (: komentář :), který bude dotaz vysvětlovat.

XPath dotazy musí mezi sebou obsahovat:

1. Použití osy `attribute` (nebo `@`)
2. Predikát testující existenci potomka (např. `otec[syn]`)
3. Predikát testující neexistenci potomka (např. `otec[not(syn)]`)
4. Predikát testující pozici prvku (tj. `[position()=...]` nebo `[...=position()]` nebo `[1]` nebo `[last()]` apod.)
5. Funkci `count()`
6. Explicitní nezkrácené použití nějaké osy (např. `following-sibling::vehicle`)

Dotazy by navíc měly dávat smysl ve vašem tématu.

B.4 Schéma XSD

Řešení musí obsahovat jeden soubor s příponou `.xml` a jeden soubor s příponou `.xsd`, volitelně také nějaký DTD soubor. Oba soubory musí být správně zformované XML soubory a XML soubor musí být validní oproti XSD souboru. XML soubor se musí na své schéma odkazovat pomocí atributu `xmlns:ns=URI` nebo `xmlns:ns=URI` ve jmenném prostoru `http://www.w3.org/2001/XMLSchema-instance`.

Používejte pouze prvky XMLSchema 1.0, jmenovitě tedy žádný assert.

Dále je třeba:

1. Použít jednoduché typy (tj. nějaký element nebo atribut musí mít předdefinovaný nebo jednoduchý typ)
2. Použít komplexní typy (tj. nějaký element musí používat vámi nadefinovaný komplexní typ)
3. Nadefinovat omezení na počet (tj. atribut `minOccurs` nebo `maxOccurs` na nějakém elementu)
4. Nadefinovat volitelný atribut (tj. `use="optional"` nebo `use="required"`)

5. Nadefinovat jednoduchý typ a použít ho v elementu nebo v atributu
6. Nadefinovat typ s atributy (tj. `simpleContent` uvnitř `complexType`, uvnitř nějž je definován atribut)
7. Nadefinovat a použít globálně definovaný element, atribut nebo typ
8. Nadefinovat lokální typ (tj. použít `simpleType` nebo `complexType` jinde, než na nejvyšší hladině)
9. Nadefinovat globální komplexní typ (tj. nějaký element se musí pomocí "type" odkazovat na nějaký `complexType` na nejvyšší hladině)
10. Použít reference (tj. atribut "ref" u elementu, skupiny, atributu nebo skupiny atributů)
11. Použít typovou dědičnost (tj. použít "extension" nebo "restriction")
12. Použít `unique`, `key` nebo `keyref`.

B.5 XQuery

Vaše řešení bude obsahovat jeden XML soubor a pět XQuery souborů.

Dotazy budou spouštěny oproti souboru "data.xml".

Dotazy musí mezi sebou dohromady použít následující konstrukce:

1. Funkci `min()`, `max()`, `avg()` anebo `sum()` v klauzuli "where"
2. "every ... satisfies" anebo "some ... satisfies"
3. "distinct-values"
4. "if (...) then ... else ..."

Následující požadavky nejsou automaticky kontrolovány, ale přesto je musíte splnit:

1. Jeden dotaz musí spojovat data ze 2 různých XML dokumentů (např. pro knihy ze seznamu knih hledá autory ze seznamu autorů, můžete využít např. vztah klíč-reference).
2. Jeden dotaz provádějící integraci heterogenních dat ze 2 různých XML dokumentů (podobně jako např. `employees.xml` a `employees2.xml`) - vytvořte si vlastní modifikovaný dokument
3. Jeden dotaz používající rekurzivní vámi definovanou funkci nebo jinak netriviální použití funkce
4. Jeden dotaz musí mít jako výstup XHTML
5. Použít konstrukci "order by"
6. Použít počítaný konstrukt

Kromě toho musí každý dotaz obsahovat komentář ve stylu (: komentář :), který popisuje, co dotaz dělá.

B.6 Transformace XSLT

Řešení musí obsahovat jeden .xml soubor a jeden .xsl soubor, volitelně také .dtd soubor. Oba soubory musí být správně zformované XML a XSL soubor musí být validní XSLT skript.

Používejte pouze XML 1.0 a XSLT 1.0. (Doporučení: Nastavte si ve vašem editoru, ať používá XSLT 1.0 a zabrání vám tak v použití pokročilých, v XMLChecku nedostupných, funkcí.)

XSLT soubor musí obsahovat:

1. Alespoň 5 šablon `xsl:template`
2. Alespoň 5 volání typu `xsl:apply-template` nebo `xsl:call-template`
3. `xsl:template/@name`
4. `xsl:template/@match`
5. `xsl:apply-templates/@mode`
6. Jedno z následujících: `xsl:choose` anebo `xsl:if`
7. `xsl:for-each`
8. `xsl:variable`
9. `xsl:param` a `xsl:with-param`
10. Jedno z následujících: `xsl:element`, `xsl:attribute`, `xsl:value-of` nebo `xsl:text`

11. Jedno z následujících: `xsl:copy` nebo `xsl:copy-of`

V XSL souboru používejte hojně komentáře, které budou vysvětlovat, co která operace znamená a co je jejím výsledkem.

Výstupním formátem musí být XML nebo HTML (nikoliv XHTML). Kromě toho by měla šablona být dostatečně komplexní.

C Seznam opravených chyb

Tento seznam obsahuje pouze chyby přítomné v původní verzi.

• Tabulky

- V tabulkách nejde označit text v prohlížeči Mozilla Firefox.
- Načítání tabulek je velmi pomalé, konstrukce HTML trvá velmi dlouho.
- V tabulkách se zobrazuje tlačítko *stáhnout výstup* i tehdy, když žádný výstup neexistuje.
- Řešení a úkoly si nelze seřadit podle data a času.
- V určitých skinech (např. *Le Frog*) není vidět obsah určitých polí, protože se zobrazuje bílý text na bílém pozadí.
- Problémy se nedají upravovat.
- Nefunguje řazení uživatelů podle počtu získaných bodů.
- Pokud uživatel do políčka pro počet řádků zadá text, který není číslo, tabulka přestane fungovat.

• Formuláře

- Při editaci úkolu/uživatele/skupiny nejsou vždy předvyplněná pole starými hodnotami.

• Úkoly

- Nezobrazují se úkoly, ke kterým není asociován kontrolující plugin.

• Přílohy, otázky a testy

- Při smazání přílohy se příloha neodstranila z otázek, které se na ní odkazovaly.

• Všechny pluginy

- Když uvnitř pluginu vypadne výjimka, zůstane nastavení aktuálního adresáře v nekonzistentním stavu, kvůli čemuž mohou selhávat automatické testy.
- Když plugin spadne, uživateli se nezobrazí žádná chybová hláška, a systém se tváří, že plugin stále ještě běží (do nekonečna).
- Kontrolující plugin spadne, pokud se během kontroly najde více než jedna chyba.
- Veškerý text v popisu chyb, který je určen pro uživatele, je oříznut tak, že se zobrazí až část po první dvojtečce.
- Některé chyby *libxml* jsou ignorovány.
- V určitých situacích produkuje systém nevalidní archivy ZIP.
- Popis formálních požadavků u mnoha úkolů není dostatečně podrobný.

• Úkol 1: XML a DTD

- Kvůli chybě v knihovně *libxml2*, kterou používá tento plugin, nelze uvnitř hodnoty notace mít znak procenta.
- Nefungují podmíněné sekce v DTD souboru.
- Notace nejsou správně parsovány.
- Entity nejsou vůbec parsovány.
- Parametrické entity nejsou rozvíjeny.
- Pokud uživatel použije kódování UTF-16, plugin spadne nebo zobrazí nesprávný výsledek.
- Detekce prvků *mixed* nefunguje.
- Soubor DTD, který není správně zformovaný, projde v některých případech kontrolou.
- Definice uvedené v interní podmnožině dokumentu XML se ignorují. Během opravování této chyby byl nahlášen bug <https://bugs.php.net/bug.php?id=67081>. Oprava chyby byla podmíněna jeho opravou.
- Některé chybové hlášky *libxml2* jsou nesrozumitelné. Systém XML Check k takovým hláškám nyní přidává pravděpodobnou příčinu chyby.
- **Úkol 2: DOM/SAX**
 - Pokud uživatel vypíše ze svého transformeru DOM nějaký text na standartní výstup, plugin spadne.
 - Uživatel musí používat Javu 5, nikoliv aktuální verzi Javy, tj. Javu 7.
 - Místo chybových hlášek se uživateli zobrazuje slovo *null*.
- **Úkol 3: XPath**
 - Test na použití `position()` selhává, když se `position()` objevilo na pravé straně výrazu nebo pokud byla použita zjednodušená varianta `[číslo]` nebo `[last()]`.
 - Bílé znaky uvnitř XPath výrazu, ač povolené standardem [5], způsobí, že řešení bude označeno za chybné.
- **Úkol 4: Schéma XSD**
 - Plugin spadne, pokud soubor XSD není správně zformovaný soubor XML.
 - Použití atributů *minOccurs* a *maxOccurs* není v některých kontextech akceptováno.
 - Dědění z typu, který není ve jmenném prostoru *xs*, se nepovažuje za dědění, ač by se za něj považovat mělo.
 - V určitých situacích není detekováno použití jednoduchého typu.
 - Použití nepovinných atributů bez explicitního stanovení pomocí *use* není detekováno.
- **Úkol 6: Transformace XSLT**
 - Pokud je XSLT soubor validní, ale XML soubor není, nezobrazí se žádná chybová hláška.
 - Je vyžadován prefix *xml*, i pokud uživatel legálně nadefinuje použití jiného prefixu.
- **Dokumentace**
 - Opraveny pravopisné a gramatické chyby.

- **Jiné chyby**

- Funkce *getTriggeredErrors* maže ze seznamu *libxml* chyb všechny chyby, i když je má jen vrátit v návratové hodnotě.
- Některé „činnosti vyžadující uživatelskou pozornost“ se na hlavní stránce nezobrazují. Toto se stane náhodně, podle toho, v jakém pořadí se vrátí AJAX požadavky ze serveru.
- Funkce *filterByKey* v určitých situacích vrátí nesprávné hodnoty.
- Funkce *Filter::isNonNegativeInteger* nic nekontroluje.
- Funkce *Filter::isDate* vyprodukuje PHP chybu na úrovni *notice*, pokud dostane jako argument něco, co není datum.
- Funkce *Filter::isRegex* nefunguje a není používána.
- Funkce *Filter::isName* vrátí vždy *true*.
- Na mnoha místech byla opravena programátorská dokumentace tříd a metod.
- Používá se nedokumentovaná funkce *jQuery 1.4.2*, která byla odstraněna v *jQuery 1.4.3* a při upgradu tedy přestane aplikace fungovat.

D Seznam přidanych nebo zmenenych funkcí

- **Kontrola podobnosti**

- Všechna nahraná řešení se nyní kontrolují na podobnost s dříve nahranými řešeními. Výsledek kontroly je k dispozici cvičícím.

- **Uživatelské rozhraní**

- Systém nyní umí zobrazovat rozhraní ve více jazycích. Všechny texty jsou nyní v angličtině i v češtině. Uživatel může mezi jazyky přepínat jak před přihlášením, tak v menu po přihlášení. Vybraný jazyk se udržuje jako cookie.
- Systém zobrazuje text „Načítání...“, když provádí déle trvající operaci.
- Systém zobrazuje v titulku „Loading...“, dokud není načtený natolik, aby dokázal zobrazit přihlašovací obrazovku.
- Uživatelé si mohou zobrazit seznam změn v systému (changelog).
- Pokud se stane chyba na straně serveru, je zobrazena uživateli místo toho, aby byla tiše ignorována.
- Bylo provedeno množství změn v zobrazovaném textu.
- Administrátor může nastavit zprávu (message of the day), která se zobrazí všem uživatelům na hlavní stránce.
- Na hlavní stránce jsou „činnosti vyžadující uživatelskou pozornost“ změněny na odkazy, kterými se lze prokliknout k obrazovce, kde lze tyto činnosti provést.
- Pokud uživatel použije odkaz na přístup k nějaké konkrétní stránce aplikace, ale není přihlášen nebo jeho sezení vypršelo, vrátí se na přihlašovací obrazovku, ale aplikace si zapamatuje, kam se chtěl dostat, a po přihlášení ho tam vrátí.

- **Uživatelské rozhraní: Tabulky**
 - Systém si nyní pamatuje uživatelem nastavenou délku tabulky v souboru cookie.
 - Studentovi se nyní v tabulce řešení zobrazuje datum a čas, kdy řešení nahrál.
- **Odevzdávání úkolů**
 - Student nyní může stáhnout úkol, který sám odevzdal.
 - Systém potvrzování úkolů byl úplně přepsán. Studenti nyní nepotvrzují řešení finálně, místo toho se řešení, které odevzdali nejpozději, počítá jako správné. Podrobnější popis je v hlavním textu práce (sekce 3.2.3).
 - Student nyní může odevzdat řešení i po termínu. Cvičící je na to ovšem upozorněn.
- **Známkování úkolů**
 - Cvičící může nyní kromě počtu bodů studentovi poslat i textové ohodnocení jeho řešení.
- **Úkol 1: XML a DTD**
 - Celý parser DTD byl přepracován. Místo balíčku PEAR pro parsování DTD se nyní používá vlastní parser Soothsilver DTD Parser.
 - Uživatel nyní musí definovat a použít nějakou entitu, a toto je kontrolováno.
 - Uživatel musí nyní správně použít deklaraci DOCTYPE v dokumentu XML a toto je kontrolováno.
- **Úkol 4: Schéma XSD**
 - Nyní se vyžaduje, aby se XML soubor správně odkazoval na soubor XSD pomocí atributů *schemaLocation* nebo *noNamespaceSchemaLocation*.
- **E-mail**
 - Po provedení některých akcí se uživateli pošle e-mail, pokud to tak mají nastaveno v uživatelských preferencích. Konkrétně se e-mail pošle po zadání nového úkolu všem studentům ve skupině tohoto úkolu, po oznámkování řešení se pošle e-mail řešiteli, a pokud student požádá o předčasné oznámkování, pošle se e-mail cvičícímu. V e-mailu je odkaz na stránku, kde se dá na spouštěcí událost zareagovat.
 - E-mail posílaný při registraci může administrátor také upravit.
 - E-mail lze také poslat přes SSL.
 - E-mail se posílá přes SwiftMailer, který nahrazuje původní PEAR modul Mail, který byl v aplikaci kvůli posílání prvotního e-mailu při založení účtu (to byl jediný druh e-mailu, který se posílal v původní verzi). Tímto již přestává být aplikace závislá na modulech PEAR.
- **Správa uživatelských údajů**
 - Nyní je možné změnit svoje jméno a svůj e-mail bez toho, že by uživatel zároveň změnil své heslo. Obdobně může nyní administrátor změnit jméno, e-mail a roli uživatele bez toho, že by změnil jeho heslo.
 - Uživatelé mají nyní možnost nechat si poslat na e-mail odkaz pro obnovení hesla. Pomocí tohoto odkazu si mohou nastavit nové heslo,

pokud zapomenou své staré.

- **Jiné**

- Pro ukládání hesel se nyní používá bezpečnější algoritmus *Blowfish* oproti původnímu MD5. Stávající uživatelé ovšem mají stále hesla zašifrovaná pomocí MD5, dokud si je nezmění.
- Když je aplikace aktualizována na novou verzi, všichni uživatelé jsou automaticky odhlášeni. Tím se zabrání zvláštním chybám vzniklým kvůli nekompatibilitě verzí.
- Nic již nelze smazat, lze to jen „schovat“, takže uživatel si myslí, že je daný objekt smazán, ale v databázi se stále nachází. Lze ho tedy použít např. k udržení konzistence databáze a ke zjišťování statistik a testování starých řešení.
- Zdrojové kódy aplikace byly změněny natolik, že dovolily upgrade systému na PHP 5.6, MySQL 5.5 a Javu 8.
- Přidány automatické jednotkové testy a automatické testy správnosti kontrolujících pluginů.
- Délky všech polí nyní nejsou omezeny nebo jsou omezeny až hodnotou 255.
- Administrátoři nyní mohou upravit popis pluginů v souborech na disku a v uživatelském rozhraní znovu nahrát informace do databáze z těchto souborů.

- **Všechny pluginy**

- Všechny hodnotící pluginy se nyní spouští synchronně, aby se k uživateli dostal výsledek rychleji.
- Když uživatel nahraje řešení, které není souborem ZIP, zobrazí se mu srozumitelná chybová hláška.
- Když uživatel nahraje soubor ZIP, který obsahuje jen jednu složku a nic jiného mimo ni, rozbalí se obsah této složky místo celého ZIP souboru, protože pravděpodobně chtěl uživatel zabalit jen tento obsah.
- Velikost písmen u koncovky není důležitá. Budou se tedy akceptovat jak soubory `.xml`, tak soubory `.XML`.
- Uživateli se nyní zobrazují všechny chyby zjištěné během kontroly, ne jen ta první.
- Soubory se nyní nemusí jmenovat přesně `data.xml`, `data.xsd` a podobně. Systém soubory v rámci archivu ZIP najde pomocí jejich koncovky.

E Seznam refaktorizací

Velká část kódu byla změněna. Zde jsou vypsané nejdůležitější změny.

- **Databáze**

- Všechny tabulky nyní používají formát uložení dat *InnoDB* místo *MyISAM*, to umožňuje použití cizích klíčů.
- Tabulky jsou nyní provázány pomocí cizích klíčů.

- Pro přístup k databázi se používá framework *Doctrine2*.
- Role *STUDENT* má nyní v databázi identifikační číslo 1, nikoliv 0. Tím pádem přestala být problematická funkce automatického inkrementování a importování dat funguje.
- **Autoload**
 - Původní systém automatického načítání tříd byl nahrazen autoloaderem Composeru.
- **Konfigurační soubory**
 - Soubor *.htaccess* byl silně zjednodušen. Nyní již není třeba měnit ho při přesunu/kopírování aplikace na jiný počítač nebo do jiné složky nebo ho generovat při instalaci.
 - Soubor *config.ini* byl rozdělen na *config.ini* a *internal.ini* a byl zjednodušen. Soubor *config.ini* obsahuje přístupové údaje k databázi a k SMTP serveru a informace o webovém serveru, na kterém systém běží. Soubor *internal.ini* obsahuje cesty interně používané systémem, které se nemění při kopírování systému na jiný počítač, a také obsahuje číslo verze.
 - Administrátor již v souboru *config.ini* nemusí zadávat cestu ke kořenovému adresáři.
- **PHP**
 - Již se nevyužívá možnost *call-by-reference*, která byla odstraněna v PHP 5.5.
 - Aplikace je nyní kompatibilní s PHP 5.6, a dokonce PHP 5.6 vyžaduje.
 - Aplikace nyní používá *Composer* pro správu závislostí.
 - Aplikace již nepoužívá žádné balíčky PEAR, který tedy není na počítači vůbec potřeba.
 - Proběhla velká refaktorizace kódu na nízké úrovni.
- **Javascript**
 - Některé Javascript soubory se nyní vkládají automaticky do hlavního HTML souboru bez toho, že by bylo nutné přidat je ručně. Protože na sobě ovšem tyto soubory v určitých případech závisí, nelze takto automaticky vložit všechny Javascript soubory.
- **Statická analýza**
 - Byla provedena statická analýza kódu v prostředí IntelliJ IDEA. Následkem toho bylo opraveno několik chyb a přidána nebo aktualizována dokumentace.
- **Adresářová struktura**
 - Byla změněna adresářová struktura, aby bylo možné upravovat celý projekt (samotný systém, pluginy, parser DTD, jednotkové testy a modul na kontrolu podobnosti) v rámci jednoho prostředí.

F Popis technologií, které se zkouší v předmětu Technologie XML

F.1 XML: Extensible Markup Language

XML (eXtensible Markup Language) je značkovací jazyk, který dává textu strukturu, a přitom není určen pro jednu konkrétní úlohu jako např. jazyk HTML pro tvorbu webových stránek. Dokument XML je textový dokument, který může vypadat např. takto:

```
1 <?xml version="1.0" encoding="utf-8">
2 <kalendář>
3   <měsíc jméno="leden">
4     <svátek den="1">Nový rok</svátek>
5     <svátek den="30">Pololetní prázdniny</svátek>
6   </měsíc>
7   <měsíc jméno="říjen">
8     <svátek den="28">Vznik Československa</svátek>
9   </měsíc>
10 </kalendář>
```

První řádka je povinná a nazýváme ji **prologem XML**. Zbytek dokumentu v tomto případě popisuje svátky a volné dny v roce. Výrazy `<kalendář>`, `<měsíc>` a podobně nazýváme **značkami** (anglicky **tagy**), které obklopují **obsah** (anglicky **content**). Všimněme si, že kromě prologu je každá značka uzavřená, tj. existuje k ní párová značka obsahující na začátku znak lomítka. Všechno mezi značkou a její párovou uzavírací značkou je obsahem **elementu** určeného těmito značkami.

Element typu svátek v tomto dokumentu má **atribut** („den“). Atributy se značí dovnitř značky a mají vždy hodnotu uzavřenou do uvozovek a rovnítkem oddělenou od jména atributu (např. „leden“).

Důležité je, že „kalendář“, „měsíc“ ani „jméno“ nejsou klíčová slova jazyka XML. Jako značku lze použít libovolné slovo.

Dokument, který jsme představili, používá jen nejjzákladnější funkce jazyka XML. O dalších strukturách jazyka XML jako např. o entitách se můžete dozvědět v [12] nebo přímo ve specifikaci jazyka [4]. Pro důkladnější porozumění jazyku doporučujeme anotované specifikace [10] nebo [11].

V době psaní této práce je specifikace jazyka XML v páté verzi, podle které také kontrolujeme řešení studentů.

Samotná specifikace jazyka XML také stanovuje syntaxi jazyka nazvaného **DTD** (Document Type Definition), který omezuje, jaké elementy a s jakými atributy se mohou vyskytnout v dokumentu XML, který danému DTD souboru odpovídá.

Výše uvedený dokument by odpovídal například tomuto DTD souboru:

```
1 <!ELEMENT kalendář (měsíc+)>
2 <!ELEMENT měsíc (svátek+)>
3 <!ELEMENT svátek #PCDATA>
```

```
4 <!ATTLIST měsíc jméno CDATA #IMPLIED>
5 <!ATTLIST svátek den CDATA #IMPLIED>
```

První řádka znamená, že dokument může obsahovat element *kalendář*, který obsahuje jednoho nebo více synů se jménem *měsíc*, ale nic jiného. Třetí řádka říká, že element *svátek* obsahuje pouze text, žádné značky. Poslední dvě řádky pak říkají, jaké atributy mohou jednotlivé elementy mít.

K dokumentu nelze jeho DTD definici přiřadit jednoznačně. Výše uvedený dokument odpovídá například i této DTD definici:

```
1 <!ELEMENT kalendář ANY>
2 <!ELEMENT měsíc ANY>
3 <!ELEMENT svátek ANY>
4 <!ATTLIST měsíc jméno CDATA #IMPLIED>
5 <!ATTLIST svátek den CDATA #IMPLIED>
```

Zde první řádka znamená, že *kalendář* může obsahovat libovolné elementy jako své syny, a dokonce je může míchat s textem.

Specifikace XML popisuje i jazyk DTD podrobněji.

F.2 DOM: Document Object Model

DOM (Document Object Model) je specifikace rozhraní pro přístup k dokumentům XML. Parser XML může programátorovi, který parser používá, zpřístupnit načtený dokument ve formě stromu DOM. To je strom, jehož uzly jsou označovány podle přesně daných instrukcí ve specifikaci DOM.

Model DOM byl nejprve aplikován v jazyce Java, nyní ovšem existuje implementace parseru DOM pro velké množství programovacích jazyků. Některé parsery XML také zpřístupňují dokument ve formě stromu, který se stromu DOM hodně podobá, ale uzly se mohou jmenovat jinak nebo mohou zpřístupňovat jiné vlastnosti. Pak se nejedná přísně vzato o parsery DOM, občas se tak ale označují kvůli jednoduchosti.

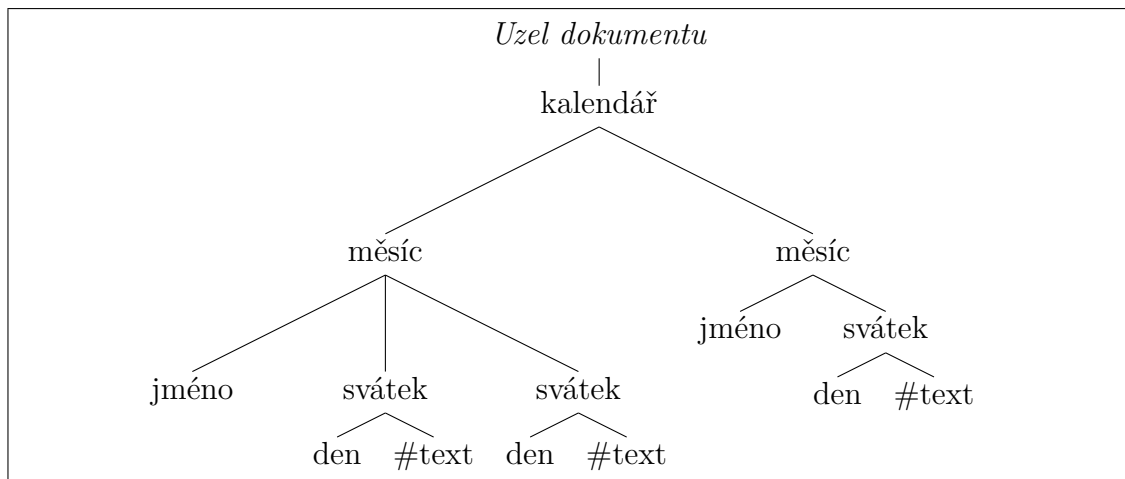
Strom DOM pro výše uvedený příklad souboru XML by vypadal jako na obrázku 33. U každého uzlu je uloženo více informací, než je zobrazeno na obrázku.

Pro stručnost byly navíc vynechány ze stromu uzly s bílými znaky, které se v něm také mají nacházet. Další informace o této technologii najdete v [12] nebo ve specifikacích [7].

F.3 SAX: Simple API for XML

SAX (Simple API for XML) je knihovna pro načítání souborů XML pro Javu. Od doby, kdy byla prvotní knihovna vytvořena, vydali různí autoři varianty SAXu i pro jiné programovací jazyky.

Načítání pomocí SAXu se zřetelně liší od načítání pomocí DOMu: parser typu SAX prochází dokument od začátku do konce a hlásí uživateli události, které



Obrázek 33: Strom DOM ukázkového dokumentu

při procházení spustí. Jakmile ovšem nějaký element projde, rovnou ho vymaže z paměti. Dá se proto výhodně použít pro parsování extrémně dlouhých dokumentů.

Ukázkový dokument výše by vygeneroval v pořadí tyto události, na které může programátor aplikace reagovat:

- DocumentStart
- ElementStart: kalendář
- ElementStart: měsíc (s atributem jméno=leden)
- ElementStart: svátek (s atributem den=1)
- Characters: Nový rok
- ElementEnd: svátek
- ElementStart: svátek (s atributem den=30)
- Characters: Pololetní prázdniny
- ElementEnd: svátek
- ElementEnd: měsíc
- ElementStart: měsíc (s atributem jméno=říjen)
- ElementStart: svátek (s atributem den=28)
- Characters: Vznik Československa
- ElementEnd: svátek
- ElementEnd: měsíc
- ElementEnd: kalendář
- DocumentEnd

Další informace o tomto druhu parsování je možné najít v [12] nebo na oficiální stránce SAX [8].

F.4 XPath

XPath je dotazovací jazyk, jehož specifikaci vydalo v roce 1999 konsorcium W3C. Jazyk XPath je široce implementován v editorech XML a v knihovnách a uživatel se s jeho pomocí může dotazovat na různá data z dokumentů XML.

Dotaz XPath se skládá z kroků oddělených znakem lomítka, kde každý krok reprezentuje sestup do hloubky ve stromu dokumentu. Dotaz vrací buď jeden uzel dokumentu, seřazenou množinu takových uzlů, číslo, logický příznak nebo text.

Několik příkladů dotazů na výše uvedeném dokumentu je v tabulce 2.

Dotaz XPath	Výsledek
/kalendář/měsíc/svátek/text()	„Vznik Československa“ (text)
/kalendář/měsíc/attribute::Jméno	(„leden“, „říjen“)
//svátek	Tři objekty typu Element, každý z nich obsahuje informaci o atributu den a o textovém obsahu
//měsíc[jméno="leden"]/svátek/text()	(„Nový rok“, „Pololetní prázdniny“)

Tabulka 2: Dotazy XPath na ukázkovém dokumentu

Více informací o jazyce XPath lze najít v [12] nebo ve specifikaci [5].

F.5 XSD: XML Schema Definition Language

XSD (XML Schema Definition Language) je jazyk popisující strukturu nějaké třídy dokumentů XML. Schéma XSD je samo napsáno formou XML souboru. Programem je možné zkontrolovat, zda daný dokument XML schématu „vyhovuje“.

Jako příklad uvedeme následující schéma, kterému vyhovuje výše uvedený dokument:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="kalendář">
4     <xs:complexType>
5       <xs:sequence>
6         <xs:element name="měsíc">
7           <xs:complexType>
8             <xs:sequence>
9               <xs:element name="svátek">
10                <xs:complexType>
11                  <xs:simpleContent>
12                    <xs:extension base="xs:string">
13                      <xs:attribute name="den" type="xs:positiveInteger" />
14                    </xs:extension>
15                  </xs:simpleContent>
16                </xs:complexType>
17              </xs:element>
18            </xs:sequence>
19            <xs:attribute name="jméno" type="xs:string" use="required" />
20          </xs:complexType>

```



```

21     </xs:element>
22   </xs:sequence>
23 </xs:complexType>
24 </xs:element>
25 </xs:schema>

```

Toto schéma říká, že dokument má kořenový element „kalendář“, který obsahuje alespoň jeden element typu „měsíc“ s povinným atributem „jméno“. Každý takový element pak musí obsahovat alespoň jeden element typu „svátek“ s volitelným atributem „jméno“ a textovým obsahem.

Jak je vidět, může existovat více schémat, kterým odpovídá ten stejný dokument XML.

Více informací o schématech je možné získat z [12] nebo ze specifikace [6].

F.6 XQuery

XQuery je dotazovací jazyk založený na jazyce XPath, je ovšem mnohem složitější a umožňuje data nejen z dokumentu získávat, ale i transformovat. Navíc dokáže pracovat s několika dokumenty současně.

Jádrem jazyka jsou takzvané výrazy FLWOR - je to zkratka z prvních písmen klíčových slov *for*, *let*, *where*, *order by* a *return*, které se v takovém výrazu v tomto pořadí vyskytují.

Tabulka 3 ukazuje základní možnosti jazyka XQuery na výše uvedeném zdrojovém dokumentu.

Dotaz XQuery	Výsledek
<pre> for \$m in kalendář/měsíc/svátek let \$d := \$m order by \$d/attribute::den return \$d </pre>	<pre> <svátek den="1">Nový rok</svátek> <svátek den="28">Vznik Československa</svátek> <svátek den="30">Polololetní prázdniny</svátek> </pre>
<pre> sum (//svátek/attribute::den) </pre>	<pre> 59 </pre>
<pre> for \$m in kalendář/měsíc/svátek let \$d := element { "významný-den"} { attribute { "den-v-týdnu" } { \$m/attribute::den }, attribute { "jméno-dne" } { \$m/text() } } return \$d </pre>	<pre> <významný-den den-v-týdnu="1" jméno-dne="Nový rok"/> <významný-den den-v-týdnu="30" jméno-dne = "Polololetní prázdniny"/> <významný-den den-v-týdnu="28" jméno-dne = "Vznik Československa"/> </pre>

Tabulka 3: Dotazy XQuery na ukázkovém dokumentu

Více informací lze najít v [12] nebo ve specifikaci jazyka XQuery [9].

F.7 XSLT

XSLT (Extensible Stylesheet Language Transformations) je skriptovací jazyk založený na XML určený k transformaci XML dokumentů. Transformátor přijme skript a dokument a vyprodukuje transformovaný dokument.

Výše uvedený dokument bychom mohli transformovat k zobrazení do formátu HTML například tímto skriptem:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:output method="html" indent="yes"/>
4   <xsl:template match="svátek">
5     <li>
6       Den <xsl:value-of select="attribute::den"/>: <xsl:value-of select="text()"
7         </li>
8   </xsl:template>
9   <xsl:template match="měsíc">
10    <li>
11      <xsl:value-of select="attribute::jméno"/>
12      <xsl:text>: </xsl:text>
13      <br>
14      <ul>
15        <xsl:apply-templates select="node()" />
16      </ul>
17    </li>
18  </xsl:template>
19  <xsl:template match="kalendář">
20    <html>
21      <head>
22        <title>Kalendář</title>
23      </head>
24      <body>
25        <ul>
26          <xsl:apply-templates select="node()" />
27        </ul>
28      </body>
29    </html>
30  </xsl:template>
31 </xsl:stylesheet>
```

Výsledkem po spuštění transformátoru bude tato stránka HTML:

```
1 <html>
2   <head><title>Kalendář</title></head>
3   <body>
4     <ul>
5       <li>leden: <br>
6         <ul>
7           <li>Den 1: Nový rok</li>
8           <li>Den 30: Pololetní prázdniny</li>
9         </ul>
```

```
10     </li>
11     <li>říjen:<br>
12         <ul>
13             <li>Den 28: Vznik Československa</li>
14         </ul>
15     </li>
16 </ul>
17 </body>
18 </html>
```

Více informací lze najít v [12] nebo ve specifikaci jazyka XSLT [6].