VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZPOZNÁNÍ PLAGIÁTŮ ZDROJOVÉHO KÓDU V JAZYCE PHP

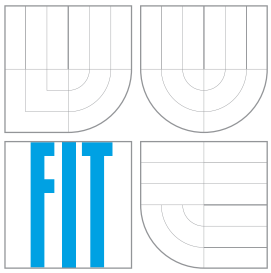
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ KRPEC

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZPOZNÁNÍ PLAGIÁTŮ ZDROJOVÉHO KÓDU V JAZYCE PHP

PLAGIARISM RECOGNIZER IN PHP SOURCE CODE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ KRPEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZBYNĚK KŘIVKA, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Cílem práce je vytvořit aplikaci, která rozpozná plagiáty projektů napsaných v jazyce PHP. Za plagiátorství lze považovat úmyslné kopírování cizího kódu, případně jeho transformací a jeho vydávání za vlastní.

Abstract

Main goal of this thesis is to create application, which can detect plagiarism in source code written in PHP language. Plagiarism is viewed as a form of code obfuscation where plagiarists deliberately perform semantics preserving transformations of original version to pass it off as their own.

Klíčová slova

PHP, plagiát, odhalování plagiátů, Halsteadova metrika

Keywords

PHP, plagiarism, plagiarism detection, Halstead metric

Citace

Ondřej Krpec: Rozpoznání plagiátů zdrojového kódu v jazyce PHP, bakalářská práce, Brno, FIT VUT v Brně, 2015

Rozpoznání plagiátů zdrojového kódu v jazyce PHP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zbyňka Křivky, Ph.D.

.....
Ondřej Krpec
22. prosince 2014

Poděkování

Velmi rád bych poděkoval vedoucímu mé bakalářské práce Ing. Zbyňku Křivkovy, Ph.D. za jeho připomínky, odborné rady, čas a ochotu při konzultacích.

Obsah

Kapitola 1

Úvod

Plagiátorství je závažným problémem nejen ve vzdělávacích a vědeckých institucích a má velmi dlouhou a bohatou historii. Mezinárodní norma ČSN ISO 5127-2003 jej popisuje jako představení duševního díla jiného autora, půjčeného nebo napodobeného vcelku nebo zčásti, jako svého vlastního. Nicméně nejvíce případů plagiátorství se stále nachází v akademických institucích, kde studenti kopírují své práce mezi sebou, v přesvědčení, že zahladili všechny stopy vedoucí k odhalení plagiátu.

Pokud se zaměříme čistě na plagiáty ve zdrojových kódech, můžeme je definovat jako program, který byl vytvořený z jiného programu tak, aby na první pohled nebylo možné rozeznat originál od kopie. Mezi základní transformace zdrojového kódu, tedy ty, na které se tato práce zaměřuje můžeme označit změnu komentářů, identifikátorů, řídících struktur, restrukturalizaci zdrojového kódu nebo změnu toku programu (výměna podmínek ve vyhodnocení `if-else`).

V rámci této bakalářské práce se pokusím ukázat způsoby, jak odhalit právě plagiáty studentských prací napsaných v jazyce PHP a poukázat na problémy, ke kterým může při automatické detekci plagiátorství docházet.

TODO: strucne popsany obsah tz

1.1 Plagiátorství

Jak již bylo uvedeno výše, plagiátorství je na akademické půdě závažným problémem a tudíž by mělo být i příslušně potrestáno. Nicméně plagiátorství zdrojových kódu sebou přináší několik problému, které značně stěžují schopnost odlišit plagiát od originálu. Tyto problémy můžeme rozlišit do šesti kategorií (Figure 1.1), z nichž pouze jednu můžeme označit jako plagiát.

- **Zdrojový kód třetích stran**, kterým jsou myšleny různé open-source kódy případně různé knihovny.
- **Nástroje na automatické generování kódu**, kde jako příklad můžeme uvést vývojové prostředí Netbeans IDE, které automaticky vytváří některé metody.
- **Obvykle používané identifikátory** jako například proměnné *result* nebo *i*.
- **Obvykle používané algoritmy** budou ve většině připadech implementované stejným způsobem. Jako příklad zde lze uvést téměř libovolný řadící algoritmus.

- **Společný autor** jednoho nebo více programů může vytvořit více různých verzí programů, které se mohou jevit jako plagiáty, protože autor má tendenci psát kód svým naučeným způsobem.
- **Opsaný kód**, který jako jediný může být označet jako plagiát, protože zde došlo ke kopírování nebo transformaci cizího kódu bez patřičného uvedení jeho autora.

Figure 1.1: Zeidmanův seznam šesti kategorií podobností kódu [odkaz na Bob Zeidman. What, exactly, is software plagiarism? Intellectual Property Today, 2007.]

Detekce plagiátorství na akademické úrovni sebou bohužel přináší ještě další problémy. Zadávané úkoly, hlavně ty v začatecnických kurzech programování bývají standardizované a velmi striktně zadávané, což může vyústit v podobně napsané programy, přestože studenti vypracovávali zadaný úkol samostatně.

1.2 Typy plagiátorství

Jako plagiátorství označujeme nejen jednoduché zkopírování zdrojového kódu, ale také jeho transformace. Tyto mohou být velice jednoduché jako pouhá změna, odebrání nebo přidávání komentářů, případně přejmenováním proměnných, ale mohou být také komplikovanější jako třeba změna struktury kódu tj. různé vnořování funkcí nebo přepsání `for` cyklů na `do-while` cykly. Páni J. A. W. Faidhi a S. K. Robinson podle těchto předpokladů definovali šest úrovní plagiátorství zdrojového kódu (Figure 1.2) od nejjednodušších technik až po ty nejsložitější.

- **Úroveň 1** - změna komentářů ve zdrojovém kódu
- **Úroveň 2** - změna názvu identifikátorů
- **Úroveň 3** - změna pozice proměnných ve zdrojovém kódu
- **Úroveň 4** - změna konstant a funkcí
- **Úroveň 5** - změna cyklů
- **Úroveň 6** - změna struktur určených pro kontrolu toku programu

Figure 1.2: Faidhi a Robinson definovali šest úrovní plagiátorství zdrojového kódu.[J. A. Faidhi and S. K. Robinson. An empirical approach for detecting program similarity and plagiarism within a university programming environment. 1987.]

Tato práce je zaměřena na odhalení všech šesti úrovní plagiátorství, přičemž hlavní pozornost je věnována prvním čtyřem úrovním, které by měly být odhaleny vždy již při povrchním porovnávání zdrojových kódů.

1.3 Jazyk PHP

Jazyk PHP se ve své první formě objevil již v roce 1994, kdy se pan Rasmus Lerdorf rozhodl, že vytvoří jednoduchý systém, který bude započítávat přístup na webové stránky. První verze byla napsána v jazyce PERL, nicméně vzhledem ke značnému zatížení serveru

bylo poté PHP přepsáno do jazyka C. V průběhu let následovalo vydání několika dalších verzí, až nakonec v roce 2003 byla oficiálně vydána beta verze PHP5, která přinesla největší změnu v podobě přidání objektového modelu.

Jazyk získal velké uplatnění zejména z důvodu, že je nezávislý na platformě a rozdíly v různých operačních systémech se omezují pouze na několik systémově závislých funkcí. Nicméně stejně jako ostatní jazyky, má i PHP nevýhody. Největší nevýhodou je fakt, že se jedná o jazyk interpretovaný, což znamená, že při jakémkoliv spuštění i toho nejmenšího skriptu, je potřeba soubor s tímto skriptem znovu kompilovat.

1.3.1 Specifika jazyka PHP

Jak již bylo uvedeno výše, tak objektový model byl do jazyka přidán až později, což nyní programátorům umožňuje vybrat si, jestli budou své programy psát využívajíc imperativního nebo objektově orientovaného paradigmatu. Tento výběr ovšem také napomáhá šíření plagiátorství, protože lze přepsáním originálního zdrojového kódu do jiného programovacího paradigmatu vytvořit na první pohled odlišný kód.

1.3.2 Kvalita kódu

Jazyk PHP byl dlouho definován pouze svou implementací a oficiální specifikace jazyka byla oznámena teprve na konci července 2014. Pokud k tomuto připočteme fakt, že v jazyku panuje nekonzistentní pojmenování funkcí, případně nejednotné pořadí parametrů ve funkcích a vezmeme v potaz i to, že jazyk samotný není obtížné se naučit, dostaneme jako výsledek mnoho nekvalitního kódu.

Kvalitu zdrojového kódu lze podle Dr. Billa Curtise definovat jako pětici požadovaných vlastností softwaru, nutných k zajištění jeho obchodní hodnoty. Tento model byl později uveden jako CISQ model kvality. (Figure 1.3)

- **Spolehlivost** - Určuje míru rizika a pravděpodobnosti případných selhání aplikace.
- **Efektivita** - Zdrojový kód a jeho struktura jsou hlavními faktory, které určují rychlost běhu aplikace.
- **Bezpečnost** - Metrika na určení náchylnosti programu na prolomení bezpečnosti zapříčiněné špatnými praktikami v návrhu a naprogramování aplikace.
- **Udržovatelnost** - Zahrnuje srozumitelnost a přenositelnost aplikace, jak mezi jednotlivými operačními systémy, tak mezi jednotlivými vývojovými týmy.
- **Velikost** - Ačkoliv velikost přímo neovlivňuje kvalitu zdrojového kódu, tak stále má velký vliv na udržovatelnost kódu a tedy nepřímo i na kvalitu výsledného kódu.

Figure 1.3: Definice kvality zdrojového kódu podle CISQ modelu[link na cisq]

Kapitola 2

Typografické a jazykové zásady

Při tisku odborného textu typu *technická zpráva* (anglicky *technical report*), ke kterému patří například i text kvalifikačních prací, se často volí formát A4 a často se tiskne pouze po jedné straně papíru. V takovém případě volte levý okraj všech stránek o něco větší než pravý – v tomto místě budou papíry svázány a technologie vazby si tento požadavek vynucuje. Při vazbě s pevným hřbetem by se levý okraj měl dělat o něco širší pro tlusté svazky, protože se stránky budou hůře rozevírat a levý okraj se tak bude oku méně odhalovat.

Horní a spodní okraj volte stejně veliký, případně potištěnou část posuňte mírně nahoru (horní okraj menší než dolní). Počítejte s tím, že při vazbě budou okraje mírně oříznuty.

Pro sazbu na stránku formátu A4 je vhodné používat pro základní text písmo stupně (velikosti) 11 bodů. Volte šířku sazby 15 až 16 centimetrů a výšku 22 až 23 centimetrů (včetně případných hlaviček a patiček). Proklad mezi řádky se volí 120 procent stupně použitého základního písma, což je optimální hodnota pro rychlost čtení souvislého textu. V případě použití systému LaTeX ponecháme implicitní nastavení. Při psaní kvalifikační práce se řiďte příslušnými závaznými požadavky.

Stupeň písma u nadpisů různé úrovně volíme podle standardních typografických pravidel. Pro všechny uvedené druhy nadpisů se obvykle používá polotučné nebo tučné písmo (jednotně buď všude polotučné nebo všude tučné). Proklad se volí tak, aby se následující text běžných odstavců sázel pokud možno na *pevný rejstřík*, to znamená jakoby na linky s předem definovanou a pevnou roztečí.

Uspořádání jednotlivých částí textu musí být přehledné a logické. Je třeba odlišit názvy kapitol a podkapitol – píšeme je malými písmeny kromě velkých začátečních písmen. U jednotlivých odstavců textu odsazujeme první řádek odstavce asi o jeden až dva čtverčíky (vždy o stejnou, předem zvolenou hodnotu), tedy přibližně o dvě šířky velkého písmene M základního textu. Poslední řádek předchozího odstavce a první řádek následujícího odstavce se v takovém případě neoddělují svislou mezerou. Proklad mezi těmito řádky je stejný jako proklad mezi řádky uvnitř odstavce.

Při vkládání obrázků volte jejich rozměry tak, aby nepřesáhly oblast, do které se tiskne text (tj. okraje textu ze všech stran). Pro velké obrázky vyčleňte samostatnou stránku. Obrázky nebo tabulky o rozměrech větších než A4 umístěte do písemné zprávy formou skládanky všité do přílohy nebo vložené do záložek na zadní desce.

Obrázky i tabulky musí být pořadově očíslovány. Číslování se volí buď průběžné v rámci celého textu, nebo – což bývá praktičtější – průběžné v rámci kapitoly. V druhém případě se číslo tabulky nebo obrázku skládá z čísla kapitoly a čísla obrázku/tabulky v rámci kapitoly – čísla jsou oddělena tečkou. Čísla podkapitol nemají na číslování obrázků a tabulek žádný vliv.

Tabulky a obrázky používají své vlastní, nezávislé číselné řady. Z toho vyplývá, že v odkazech uvnitř textu musíme kromě čísla udat i informaci o tom, zda se jedná o obrázek či tabulku (například "... *viz tabulka 2.7* ..."). Dodržování této zásady je ostatně velmi přirozené.

Pro odkazy na stránky, na čísla kapitol a podkapitol, na čísla obrázků a tabulek a v dalších podobných příkladech využíváme speciálních prostředků DTP programu, které zajistí vygenerování správného čísla i v případě, že se text posune díky změnám samotného textu nebo díky úpravě parametrů sazby. Příkladem takového prostředku v systému LaTeX je odkaz na číslo odpovídající umístění značky v textu, například návěští (`\ref{navesti}` – podle umístění návěští se bude jednat o číslo kapitoly, podkapitoly, obrázku, tabulky nebo podobného číslovaného prvku), na stránku, která obsahuje danou značku (`\pageref{navesti}`), nebo na literární odkaz (`\cite{identifikator}`).

Rovnice, na které se budeme v textu odvolávat, opatříme pořadovými čísly při pravém okraji příslušného řádku. Tato pořadová čísla se píší v kulatých závorkách. Číslování rovnic může být průběžné v textu nebo v jednotlivých kapitolách.

Jste-li na pochybách při sazbě matematického textu, snažte se dodržet způsob sazby definovaný systémem LaTeX. Obsahuje-li vaše práce velké množství matematických formulí, doporučujeme dát přednost použití systému LaTeX.

Mezeru neděláme tam, kde se spojují číslice s písmeny v jedno slovo nebo v jeden znak – například *25krát*.

Členicí (interpunkční) znaménka tečka, čárka, středník, dvojtečka, otazník a vykřičník, jakož i uzavírací závorky a uvozovky se přimykají k předcházejícímu slovu bez mezery. Mezera se dělá až za nimi. To se ovšem netýká desetinné čárky (nebo desetinné tečky). Otevírací závorka a přední uvozovky se přimykají k následujícímu slovu a mezera se vynechává před nimi – (takto) a "takto".

Pro spojovací a rozdělovací čárku a pomlčku nepoužíváme stejný znak. Pro pomlčku je vyhrazen jiný znak (delší). V systému TeX (LaTeX) se spojovací čárka zapisuje jako jeden znak "pomlčka" (například "Brno-město"), pro sázení textu ve smyslu intervalu nebo dvojic, soupeřů a podobně se ve zdrojovém textu používá dvojice znaků "pomlčka" (například "zápas Sparta – Slavie"; "cena 23–25 korun"), pro výrazné oddělení části věty, pro výrazné oddělení vložené věty, pro vyjádření nevyslovené myšlenky a v dalších situacích (viz Pravidla českého pravopisu) se používá nejdelší typ pomlčky, která se ve zdrojovém textu zapisuje jako trojice znaků "pomlčka" (například "Další pojem — jakkoliv se může zdát nevýznamný — bude neformálně definován v následujícím odstavci."). Při sazbě matematického mínus se při sazbě používá rovněž odlišný znak. V systému TeX je ve zdrojovém textu zapsán jako normální mínus (tj. znak "pomlčka"). Sazba v matematickém prostředí, kdy se vzoreček uzavírá mezi dolary, zajistí vygenerování správného výstupu.

Lomítko se píše bez mezer. Například školní rok 2008/2009.

Pravidla pro psaní zkratek jsou uvedena v Pravidlech českého pravopisu [1]. I z jiných důvodů je vhodné, abyste tuto knihu měli po ruce.

2.1 Co to je normovaná stránka?

Pojem *normovaná stránka* se vztahuje k posuzování objemu práce, nikoliv k počtu vytištěných listů. Z historického hlediska jde o počet stránek rukopisu, který se psal psacím strojem na speciální předtištěné formuláře při dodržení průměrné délky řádku 60 znaků a při 30 řádcích na stránku rukopisu. Vzhledem k zápisu korekturních značek se používalo řádkování 2 (ob jeden řádek). Tyto údaje (počet znaků na řádek, počet řádků a proklad

mezi nimi) se nijak nevztahují ke konečnému vytištěnému výsledku. Používají se pouze pro posouzení rozsahu. Jednou normovanou stránkou se tedy rozumí 60*30 = 1800 znaků. Obrázky zařazené do textu se započítávají do rozsahu písemné práce odhadem jako množství textu, které by ve výsledném dokumentu potisklo stejně velkou plochu.

Orientační rozsah práce v normostranách lze v programu Microsoft Word zjistit pomocí funkce *Počet slov* v menu *Nástroje*, když hodnotu *Znaky (včetně mezer)* vydělíte konstantou 1800. Do rozsahu práce se započítává pouze text uvedený v jádru práce. Části jako abstrakt, klíčová slova, prohlášení, obsah, literatura nebo přílohy se do rozsahu práce nepočítají. Je proto nutné nejdříve označit jádro práce a teprve pak si nechat spočítat počet znaků. Přibližný rozsah obrázků odhadnete ručně. Podobně lze postupovat i při použití OpenOffice. Při použití systému LaTeX pro sazbu je situace trochu složitější. Pro hrubý odhad počtu normostran lze využít součet velikostí zdrojových souborů práce podělený konstantou cca 2000 (normálně bychom dělili konstantou 1800, jenže ve zdrojových souborech jsou i vyznačovací příkazy, které se do rozsahu nepočítají). Pro přesnější odhad lze pak vyextrahovat holý text z PDF (např. metodou cut-and-paste nebo *Save as Text...*) a jeho velikost podělit konstantou 1800.

Kapitola 3

Závěr

Závěrečná kapitola obsahuje zhodnocení dosažených výsledků se zvlášť vyznačeným vlastním přínosem studenta. Povinně se zde objeví i zhodnocení z pohledu dalšího vývoje projektu, student uvede náměty vycházející ze zkušeností s řešeným projektem a uvede rovněž návaznosti na právě dokončené projekty.

Literatura

[1] Kolektiv autorů: *Pravidla českého pravopisu*. Academia, 2005, iSBN 80-200-1327-X.