

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

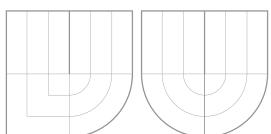
**NÁSTROJ PRO PODPORU VÝVOJE SOFTWAROVÝCH
SYSTÉMŮ**

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

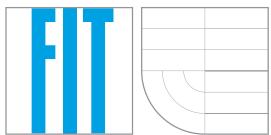
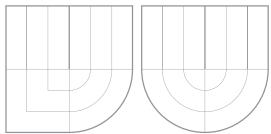
AUTOR PRÁCE
AUTHOR

KAREL HALA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NÁSTROJ PRO PODPORU VÝVOJE SOFTWAROVÝCH SYSTÉMŮ

TOOL FOR SOFTWARE SYSTEMS DESIGN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VEDOUCÍ PRÁCE
SUPERVISOR

KAREL HALA

Ing. RADEK KOČÍ, Ph.D.

BRNO 2013

Abstrakt

Nástroj pro podporu vývoje softwarových systémů je aplikace sloužící pro vizuální znázorňení průběhu vývoje aplikace. Slouží k porozumění zákazníkovým požadavkům, připravení návrhu a rozvězení jednotlivých tříd. Aplikace má za cíl ulehčit rozvržení práce před její implementací a pomocí uživateli odhalit některé chyby, které by mohly nastat během implementace.

Abstract

Tool for software systems design is application for visualization of development application. It's main goal is to achieve connection between developer and customer. It should be used to understand customer's needs, prepare workflow of project and understanding each class. This application should make easier to understand some flaws, that might occur when creating software.

Klíčová slova

Případ užití, Diagram tříd, Nástroj, Softwarový vývoj

Keywords

Use Case, Class diagram, Tool, Software development

Citace

Karel Hala: Nástroj pro podporu vývoje softwarových systémů, bakalářská práce, Brno, FIT VUT v Brně, 2013

Nástroj pro podporu vývoje softwarových systémů

Prohlášení

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

.....
Karel Hala
6. března 2014

Poděkování

Předem bych rád poděkoval panu doktoru Kočímu, za odbornou pomoc a vysvětlení propojení mezi jednotlivými částmi aplikace.

© Karel Hala, 2013.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1	Úvod	2
1.1	Seznámení s nástrojem	2
2	Zhondocení konkurence	3
2.1	Umbrello	3
2.1.1	Popis aplikace	3
2.1.2	Výhody a nevýhody	3
2.1.3	Zhodnocení a převzaté vlastnosti	4
2.2	Enterprise Architect	5
2.2.1	Popis aplikace	5
2.2.2	Výhody a nevýhody	5
2.2.3	Zhodnocení a převzaté vlastnosti	5
3	Grafy pro popis částí	6
3.1	Případ žití	6
3.1.1	Technický popis případu užití	6
3.1.2	Přidané prvky	6
3.2	Diagram tříd	6
3.2.1	Technický popis diagramu tříd	6
3.2.2	Přidané prvky	6
3.3	Objektově orientované petriho sítě	6
3.3.1	Technický popis objektově orientovaných petriho sítí	6
3.3.2	Přidané prvky	6
4	Popis aplikace	7
4.1	Editace a chování prvků	7
4.2	Editace a chování spojů	7
5	Nikdy to nebude naprostě dokonalé	9
6	Typografické a jazykové zásady	10
6.1	Co to je normovaná stránka?	11
7	Závěr	13
A	Obsah CD	15
B	Manual	16

Kapitola 1

Úvod

Při implementaci jakéholidv nástroje se často naráží na bariéru mezi zákazníkem a programátorem. Zákazník často netuší co chce a programátor často zákazníka špatně pochopí. Tento nástroj bohužel tyto problémy nedokáže odstranit, nicméně měl by napomoci odstranit některé bariéry a pomoci snažší vizualizace problémů.

Nástroj tohoto docílí za použití několika částí. Případu užití [5] pro jednotlivé znázornění zákazníkových potřeb, diagramu tříd pro vlastní vizualizaci a pochopení vyvýjeného softwaru a nakonec objektově orientované petriho síťe, sloužící pro porozumění celého systému do hloubky.

1.1 Seznámení s nástrojem

Jak bylo již řečeno pro porozumění požadavků zákazníka nám poslouží diagram případu užití, odsud se předgenerují jednotlivé třídy a ty následně bude možné upravit a popsat v části objektově orientovaných petriho sítí.

Kapitola 2

Zhondocení konkurence

Z hlediska snažšího porozumění byl brán zřetel na konkurentní nástroje. Převážně byl brán zřetel na jejich jednoduchost a co je dělá tak hojně využívané a snadno použitelné. Z toho důvodu byly prozkoumány dva nástroje Umbrello a komerční Enterprise Architect. Detailnější popis jednotlivých aplikací, jejich výhod, nedostatků a převzatých vlastností je popsán v sekci 2.1 a 2.2.

2.1 Umbrello

Aplikace celým názvem **Umbrello UML Modeler** [3] aplikace primárně určena pro Unix based systémy, nicméně je možné jej bez problémů nainstalovat také na windows. Za pomoci KDE installera a vybrání požadovaného balíčku. Ovládání tohoto installera, je velice jednoduché a intuitivní. Mezi její hlavní výhody patří refactoring pro některé jazyky. Nicméně graficky zaostává za konkurencí ostatních nástrojů pro tvorbu UML diagramů.

2.1.1 Popis aplikace

Aplikace je velice robustní a poskytuje mnoho lastností. Při ohlédnutí na to, že je volně ke stažení a volně k používání, aplikace používá licenci GNU general public license [2]. Což umožňuje komunitu být zapojenou při vývoji, odchyťávat chyby a sami je opravovat. Pokud vývojář nicméně chce být zahrnut do vývoje, musí být členem KDE komunity. Umbrello poskytuje mnoho UML grafů a práce s touto aplikací je poměrně jednoduchá, ale její jednoduchý design má mnoho špatných návrhů. Pro menší aplikace a pro začínající firmy je naprosto dokonalá, ale absence pokročilého editoru a ne příliš jednoduché práce s editorem je opravdu žalostná.

2.1.2 Výhody a nevýhody

Umbrello aplikace poskytuje několikero UML grafů. **Class Diagram**, pro popis tříd a provázanosti mezi nimi. **Sequence Diagram**, pro popis chodu aplikace, komunikace mezi vlákny a komunikace mezi aplikacemi. **Use Case Diagram** pro popis akcí v systému, aplikaci. **State Diagram** pro popis stavu systému a průběhu jednotlivých operací, převážně zachování v určitých momentech aplikace na dané podmínky. **Activity Diagram** popisuje akce a průběh celého programu graficky. **Component Diagram** pro popis spojení a komunikace mezi jednotlivými komponentami. **Deployment Diagram** popisuje fyzické uložení

a mašiny, na kterých program bude pracovat. A v neposlední řadě **Entity Relationship Diagram** pro popis vztahu dat mezi sebou.

Jak je vidno Umbrello poskytuje velké množství editorů a možností, jak znázornit chod aplikace. Aplikace je také velice intuitivní, ať již díky klávesovým zkratkám, tak také díky jednoduchému grafickému designu. Umbrello, také podporuje generaci a refactoring zdrojového kódu pro některé jazyky. Stačí pouze importovat jednu již vytvořenou třídu a Umbrello vás provede jednoduchým nastavením, výběrem které třídy namodelované v aplikaci se mají vygenerovat a nabídne také možnost upravit mírně zvolený kód. Celkově funkce pro generování zdrojového kódu a jeho úprava je velice potěšující a s ohlédnutím na to, že Umbrello je volně šířitelné, je tato vlastnost velice dobrá.

Umbrello dále poskytuje možnost generování částečné dokumentace, i když nejsou poskytnutu komentáře. Při generování tříd je možnost, také importovat zdrojový kód a nechat umbrello vše vygenerovat samo. Tato vlastnost je, ale podporována pouze pro jazyky **ActionScript**, **Ada**, **C++**, **C#**, **D IDL**, **Java**, **Javascript**, **MySQL**. Výběr je tedy více než bohatý.

Dále Umbrello poskytuje možnost exportování objektů jako PNG obrázky. Tato funkčnost je velice příjemná při psaní dokumentů stačí část aplikace jednoduše okopírovat a vložit jako obrázek. Exportovat jako obrázek, je také možno celý diagram. A také tisknout daný diagram.

Hlavní nevýhoda aplikace je pravděpodobně ne příliš dobře udělaný editor. Editor je sice jednoduchý na používání a intuitivní, nicméně dlouhodobější práce je zdlouhavá a příliš repetitivní. Nutnost vytvoření objektu na pracovním plátně, poté výběru nástroje pro přesun a editaci daného objektu je zdlouhavé a špatné. Jednodušší pro práci by jistě bylo vytvoření objektu a po kliknutí na již vytvořený objekt jej pouze přesunout. Vytvoření nového atributu třídy je opět velice náročné a neintuitivní.

Co je ale největší slabinou Umbrello aplikace, je nevalidace daného diagramu. Je čistě na uživateli hlídat si veškeré chyby a validitu modelované aplikace.

2.1.3 Zhodnocení a převzaté vlastnosti

Aplikace Umbrello je špičkou ve své třídě. Je zdarma a poskytuje mnoho nástrojů při návrhu aplikace. Mezi jeho přednosti jistě patří možnost refactoringu a také rozsáhlé možnosti ve variabilitě grafických návrhů.

Aplikace Umbrello poskytla mnoho informací o vývoji editační aplikace. Jakým chybám se vyvarovat a co se naopak hodí. Z aplikace Umbrello bylo především použita jednoduchost a intuitivní ovládání. Rozdělení aplikace na jednotlivé části a poté jednoduchý grafický design. Není za potřebí ve výsledné aplikaci mnoho náročných grafických elementů. Tyto elemnty by byly rušivé, tudíž na základě Umbrello aplikace byl zvolen jednoduchý design.

Na základě chyb a nedostatků, převážně tomu, že Umbrello neposkytuje jakoukoliv kontrolu daného modelu, bylo v aplikaci Nástroj pro podporu vývoje softwarových systémů implementována možnost kontroly editovaného grafu. A to přesněji kontrola spojů, zda je daný spoj mezi dvěma objekty na pracovním plátně validní.

V Umbrello nelze dané spoje nikterat ohýbat, což dělá výdledný diagram velice neprehledný. Dále není možno v Umbrello nikterak editovat dané spoje, tomuto bylo vyhnuto a přidána editace spojů s možností jednak změny typu, dále také změny objektu na který daná hrana ukazuje a také možnost jednoduchého odebrání hrany.

V Umbrello není nikde vidět, zda daný objekt má hrany, které vedou do a nebo z objektu. Toto bylo ve výsledné aplikaci přidáno, což následně změnilo grafické čtení modelu

a jednoduchou identifikaci objektů.

Ve výsledné aplikaci, bylo také vyhnuto vlastnosti automatického vyskakování okna při přidání nového objektu, toto bylo nahrazeno pozdější úpravou daného objektu. Bylo vzáno v potaz, že uživatel bude chtít v jeden moment vytvořit více objektů a až poté je pojmenovat a pospojovat.

2.2 Enterprise Architect

Aplikace [1] je tvořena firmou sparx system <http://www.sparxsystems.com>, tato firma se soustřeďuje na tvorbu nástrojů pro tvorbu UML grafů. Bohužel firmou nabízené produkty jsou placené, ale za dané služby člověk obdrží opravdu robustní systém pro tvorbu UML grafů a možnost refactoringu pro několik jazyků.

2.2.1 Popis aplikace

Enterprise Architect je velice roustní systém, již nemá takové nedostatky jako aplikace Umbrello a nabízí několikrát možností pro refactoring projektu. Aplikace rozděluje UML grafy do dvou skupin, skupina **Structural Diagrams** pro diagramy na popis vyvýjené aplikace a popis její struktury. A skupinu **Behavior Diagrams** pro popis chování aplikace a komunikace mezi jednotlivými částmi aplikace. Aplikace poskytuje velkou škálu grafů a diagramů od klasických diagramů pro popis chování dat mezi sebou, přes diagramy pro návrh Win32 UI až po návrh testování výsledné aplikace.

2.2.2 Výhody a nevýhody

Aplikace zastává velké množství grafů a diagramů, pro jednodušší ovládání aplikace dostane uživatel možnost popisu všech částí aplikace a vysvětlení všech diagramů.

Grafické rozhraní již není tak jednoduché, jako tomu je u aplikace Umbrello, ale je intuitivní a jednoduché pro používání. Většina pokročilých funkcí je bohužel schována a je náročné ji najít. Přidávání atributů pro jednotlivé třídy je opět zdlouhavé a neintuitivní. Aplikace Umbrello má vyřešenu tuto vlastnost lépe, ale ne dokonale.

Výhodou této aplikace jistě je možnost vytváření diagramů pro jednotlivé jazyky, celkem jedenáct jazyků je podporovaných. **ActionScript**, **C**, **C#**, **C++**, **Delphi**, **Java**, **PHP**, **Python**, **VBNet**, **Visual Basic**, **WorkFlow Script**.

Největší výhodou je jistě možnost importovat celý obsah složky se zdrojovými kódy a tento projekt poté zobrazit v diagramu tříd. Import nicméně probíhá poněkud složitěji, než v aplikaci Umbrello.

2.2.3 Zhodnocení a převzaté vlastnosti

Kapitola 3

Grafy pro popis částí

Pro grafické znázornění jednotlivých částí byly vybrány z UML grafů případ užití 3.1 pro popis jednotlivých akcí v systému, diagram tříd 3.2 pro grafické znázornění tříd a operace mezi nimi. A jako poslední část, která nespadá pod UML bylo vybráno objektově orientovaných petriho sítí 3.3 pro popis chování jednotlivých tříd.

3.1 Případ žití

[5]

3.1.1 Technický popis případu užití

3.1.2 Přidané prvky

3.2 Diagram tříd

[4]

3.2.1 Technický popis diagramu tříd

3.2.2 Přidané prvky

3.3 Objektově orientované petriho sítě

[7]

3.3.1 Technický popis objektově orientovaných petriho sítí

3.3.2 Přidané prvky

Kapitola 4

Popis aplikace

Jak již bylo řečeno v předešlé části, bylo rozhodnuto o jednoduchosti a intuitivním ovládání aplikace. Tohoto bylo docíleno sjednocením základního designu aplikace, sjednocení klávesových zkratky a porozumění a prozkoumání moderních aplikací.

Hlavní okno se skládá z několika panelů. Horní část obsahuje otevřené projekty, jejich částí a ovládací prvky. Celý hlavní obsah aplikace se nachází ve zbylých dvou třetinách okna. V levé části se dále nachází tlačítka pro možnost přepínání jednotlivých vkládaných objektů. Dolní část obsahuje informační a editační možnosti zvoleného prvku a pro hlavní část, je vyhrazen zbytek pracovní plochy, takzvané editační plátno.

Na toto plátno je možné přidávat jednotlivé prvky, spojovat je, přesouvat je a mazat je. Po vybrání některého tlačítka v levém menu je možné začít přidávat prvky. Bylo zvoleno jednoduché a intuitivní chování aplikace, po kliknutí se vytvoří objekt, po tažení se přesune objekt a pro spojení dvou objektů hranou je zapotřebí kliknout na první objekt a poté na druhý, přičemž spoj je viditelný od kurzoru po první objekt. Pro zahnutí spoje je možno kliknout pravým tlačítkem myši, což vyvolá vytvoření ohybu spoje.

4.1 Editace a chování prvků

4.2 Editace a chování spojů

Naším cílem je vytvořit jasný a srozumitelný text. Vyjadřujeme se proto přesně, píšeme dobrou češtinou (nebo zpravidla angličtinou) a dobrým slohem podle obecně přijatých zvyklostí. Text má upravit čtenáři cestu k rychlému pochopení problému, předvídat jeho obtíže a předcházet jim. Dobrý sloh předpokládá bezvadnou gramatiku, správnou interpunkci a vhodnou volbu slov. Snažíme se, aby náš text nepůsobil příliš jednotvárně používáním malého výběru slov a tím, že některá zvlášť oblíbená slova používáme příliš často. Pokud používáme cizích slov, je samozřejmým předpokladem, že známe jejich přesný význam. Ale i českých slov musíme používat ve správném smyslu. Např. platí jistá pravidla při používání slova *zřejmě*. Je *zřejmě* opravdu *zřejmé*? A přesvědčili jsme se, zda to, co je *zřejmě* opravdu platí? Pozor bychom si měli dát i na příliš časté používání zvratného se. Například obratu *dokázalo se*, že... zásadně nepoužíváme. Není špatné používat autorského *my*, tím předpokládáme, že něco řešíme, nebo například zobecňujeme spolu se čtenářem. V kvalifikačních pracích použijeme autorského *já* (například když vymezujeme podíl vlastní práce vůči prevzatému textu), ale v běžném textu se nadměrné používání první osoby jednotného čísla nedoporučuje.

Za pečlivý výběr stojí i symbolika, kterou používáme ke *značení*. Máme tím na mysli volbu zkratek a symbolů používaných například pro vyjádření typů součástek, pro označení hlavních činností programu, pro pojmenování ovládacích kláves na klávesnici, pro pojmenování proměnných v matematických formulích a podobně. Výstižné a důsledné značení může čtenáři při četbě textu velmi pomoci. Je vhodné uvést seznam značení na začátku textu. Nejen ve značení, ale i v odkazech a v celkové tiskové úpravě je důležitá důslednost.

S tím souvisí i pojem z typografie nazývaný *vyznačování*. Zde máme na mysli způsob sazby textu pro jeho zvýraznění. Pro zvolené značení by měl být zvolen i způsob vyznačování v textu. Tak například klávesy mohou být umístěny do obdélníčku, identifikátory ze zdrojového textu mohou být vypisovány **písmem typu psací stroj** a podobně.

Uvádíme-li některá fakta, neskrýváme jejich původ a náš vztah k nim. Když něco tvrdíme, vždycky výslovně uvedeme, co z toho bylo dokázáno, co teprve bude dokázáno v našem textu a co přebíráme z literatury s uvedením odkazu na příslušný zdroj. V tomto směru nenecháváme čtenáře nikdy na pochybách, zda jde o myšlenku naší nebo převzatou z literatury.

Nikdy neplýtváme čtenářovým časem výkladem triviálních a nepodstatných informací. Neuvádíme rovněž několikrát totéž jen jinými slovy. Při pozdějších úpravách textu se nám může některá dříve napsaná pasáž jevit jako zbytečně podrobná nebo dokonce zcela zbytečná. Vypuštění takové pasáže nebo alespoň její zestručnění přispěje k lepší čitelnosti práce! Tento krok ale vyžaduje odvahu zahodit čas, který jsme jejímu vytvoření věnovali.

Kapitola 5

Nikdy to nebude naprosto dokonalé

Když jsme už napsali vše, o čem jsme přemýšleli, uděláme si den nebo dva dny volna a pak si přečteme sami rukopis znovu. Uděláme ještě poslední úpravy a skončíme. Jsme si vědomi toho, že vždy zůstane něco nedokončeno, vždy existuje lepší způsob, jak něco vysvětlit, ale každá etapa úprav musí být konečná.

Kapitola 6

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 rychlosť čtení souvislého textu. V případě použití systému LaTeX ponecháme implicitní nastavení. Při psaní kvalifikační práce se řídte 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 písma 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 rozdílných větších než A4 umístěte do písemné zprávy formou skládanky všitě 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 návštětí (`\ref{navesti}`) – podle umístění návštětí 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 rádku. 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 mínsus se při sazbě používá rovněž odlišný znak. V systému TeX je ve zdrojovém textu zapsán jako normální mínsus (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 [6]. I z jiných důvodů je vhodné, abyste tuto knihu měli po ruce.

6.1 Co to je normovaná stránka?

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

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

Orienteční 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 vyextrahouvat holý text z PDF (např. metodou cut-and-paste nebo *Save as Text...*) a jeho velikost podělit konstantou 1800.

Kapitola 7

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] Enterprise architect. <http://www.sparxsystems.com/products/ea/>, accessed: 22.03.2013.
- [2] Gnu General public license v 2.0.
<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>, accessed: 23.03.2013.
- [3] Umbrello Project. <http://umbrello.kde.org/>, accessed: 22.03.2013.
- [4] Kettenis, J.: *Getting Started With UML Class Modeling*. Oracle, May 2007, accessed: 22.03.2013.
- [5] Kettenis, J.: *Getting Started With Use Case Modeling*. Oracle, May 2007, accessed: 22.03.2013.
- [6] Kolektiv autorů: *Pravidla Českého pravopisu*. Academia, 2005,
ISBN 80-200-1327-X.
- [7] Radek Kočí, Vladimír Janoušek, and František Zbořil, jr. : Object Oriented Petri Nets – Modelling Techniques Case Study. In *International Journal of Simulation Systems, Science and Technology*, 3, ročník 10, May 2009.

Příloha A

Obsah CD

Příloha B

Manual