Technická univerzita v Liberci

FAKULTA PEDAGOGICKÁ

Katedra: Aplikované Informatiky

Studijní program: 3. stupeň

Kombinace: Informatika – Anglický jazyk

KOREKTOR PÍSEMNÝCH PRACÍ V ANGLICKÉM JAZYCE

CORRECTOR OF WRITTEN ASSIGNMENTS IN ENGLISH

Autor:

Petr Koutek

Podpis:

.....

Adresa:

Tulipánová 1051

463 11, Liberec 30

Vedoucí práce: RNDr. Petr Kolář

Konzultant: PeadDr. Zuzana Šaffková, M.A., CSc., KAJ TU v Liberci

V Liberci dne: 30. 4. 2006

TU v Liberci, FAKULTA PEDAGOGICKÁ
461 17 LIBEREC 1, Hálkova 6 Tel.: 485 352 515 Fax: 485 352 332

Katedra: Aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(pro magisterský studijní program)

diplomant	Petr Koutek
adresa:	Tulipánová 1051, Liberec 30
obor (kombinace):	Inf-Aj
Název DP:	**Korektor písemných prací v angličtině**
Název DP v angličtině:	Corrector of Written Assignments in English
Vedoucí práce:	RNDr. Petr Kolář, KAI TU v Liberci
Konzultant:	PeaDr. Zuzana Šaffková, M. A., CSc., KAJ TU v Liberci
Termín odevzdání:	Květen 2006

Pozn. Podmínky pro zadání práce jsou k nahlédnutí na katedrách. Katedry rovněž formulují podrobnosti zadání. Zásady pro zpracování DP jsou k dispozici ve dvou verzích (stručné resp. metodické pokyny) na katedrách a na Děkanátě Fakulty pedagogické TU v Liberci.

……. …….

děkan **vedoucí katedry**

Převzal (diplomant): Petr Koutek
Datum: 22. 3. 2005

Podpis: ..……

Cíl: Cílem práce je vytvořit program pro podporu opravování studentských prací v anglickém jazyce. Program bude zpracovávat soubory formátu RTF, bude umožňovat vkládání komentářů a korekčních značek a měl by být schopný vyhledat chyby v textu (lexikální, popřípadě i gramatické a stylistické).

Doporučená literatura:

Cantu, M.: Myslíme v jazyku Delphi 6 (1. a 2. díl). Grada Publishing, Praha, 2002

Cantu M.: Myslíme v jazyku Delphi 7. Grada Publishing, Praha 2003

Holan, T.: Delphi v příkladech, BEN, Praha, 2001

Písek, S.: Delphi – praktický průvodce. Grada, Praha, 2001

Svoboda, L. a kol.: 1001 tipů a triků pro Delphi. Computer Press, Brno, 2001

Greenbaum, S.; Quirk, R.: A Student's Grammar of English Language. Longman., Harlow, 1990

Leki, I.: Academic Writing: Techniques and Tasks. St. Martin's Press, N.Y., 1990

Smalley, R.: Reuthen, M.: Refining Composition Skills: Rhetoric and Grammar for ESL students. Heinle & Heinle Publishers, University of New Orleans, 190

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci dne: 30. 4. 2006 Petr Koutek

.....

Poděkování:

Děkuji všem, bez nichž bych asi nikdy svoji práci nedokončil. Dík náleží mé přítelkyni a rodině za trpělivost a podporu. Vedoucímu práce RNDr. Petru Kolářovi a PeadDr. Zuzaně Šaffkové, M.A., CSc. děkuji za zasvěcené vysvětlování problémů tvorby diplomové práce a za podnětné rady. V neposlední řadě bych rád tou cestou poděkoval Ing. Jakubu Lankovi za pomoc při testování aplikace.

Korektor písemných prací v anglickém jazyce

KOUTEK Petr **DP–2006** **Vedoucí DP:** RNDr. Petr Kolář

Resumé

Cílem diplomové práce bylo navrhnout a posléze naprogramovat aplikaci, která by usnadnila a zrychlila opravování a vyhodnocování písemných prací studentů v anglickém jazyce. Práce se zabývá problematikou psaní a hodnocení písemných prací a využívá tuto teorii pro návrh a realizaci aplikace. Nedílnou částí práce jsou i kapitoly zabývající se jednotlivými prostředky, které byly využity k sestavení této aplikace. Značná část diplomové práce je věnována i popisu samotné aplikace.

Corrector of Written Assignments in English

Summary

The aim of the diploma thesis is to design and programme an application that would speed the correction and evaluation of students' written assignments in English language. The thesis deals with the problematic of essay writing and evaluation and uses this theory for the project and implementation of the application. A portion of the thesis focuses on chapters dealing with individual means used for the creation of the application. A considerable part of the thesis is dedicated to actual description of the application itself.

Obsah

Seznam obrázků

Úvod

Opravování písemných prací – téměř každodenní rutina učitele. Luštit nečitelné písmo studentů a donekonečna psát obdobné, ne-li stejné, vysvětlivky a komentáře studentům, které objasňují, kde a proč udělali chybu. Právě tato úmorná práce mě přiměla položit si otázku: „Je možné zjednodušit opravování písemných prací?“.

Cílem této diplomové práce bylo vypracovat a realizovat návrh, který by dal jasnou, v praxi použitelnou a především kladnou odpověď na tuto otázku. Výsledkem celého tohoto úsilí je aplikace *Essay Correction Assistant.* V následujících kapitolách seznámím čtenáře s:

- úskalím a zásadami opravování písemných prací
- požadavky kladenými na program a konceptem programu
- popisem programu a jeho použitím
- prostředky využitými při tvorbě aplikace

1. Písemné práce v anglickém jazyce

Psaní, jedna ze základních komunikačních dovedností. Je vůbec důležité v dnešní moderní době technologických zázraků a inovací? Bohužel někteří si myslí, že nikoliv. Jak upozorňuje CHASTIAN, role psaní je v jazykových hodinách podceňována, obzvláště od vzniku audio-jazykových přístupů, které prosazují mluvení nad psaním jakožto hlavní produktivní jazykovou dovednost. Záhy však dodává, že by učitelé měli pečlivě zvážit, zdali obětují psaní ve prospěch ostatních jazykových dovedností (poslech, čtení, mluvení). [2, s. 245 – 247]

Psaní je s jazykem spjato již od pradávna a příchodem moderních technologií se jeho význam ještě umocnil. Díky e-mailu, chatu a jiným technologiím již není pro studenty problém komunikovat s celým světem pomocí psaného slova. Je proto důležité, aby studenti uměli tuto dovednost v anglickém jazyce využít. Jedním z prostředků při výuce psaní v anglickém jazyce jsou písemné práce.

1.1. Vyčlenění termínu písemná práce

Pro potřeby této diplomové práce budeme chápat termín písemná práce jako libovolný literární útvar. Může to být i jeden malý odstavec, nebo celá slohová práce. Rozhodně však tento termín nebudeme vnímat jako testy gramatiky, nebo jim cokoliv podobného.

1.2. Vyhodnocování písemných prací

Jedna z nejdůležitějších povinností učitele je poskytování zpětné vazby studentům, a tak jim pomoci při vyhodnocování jejich úspěchu a pokroku. Pojem zpětná vazba může být popsán jako interakce mezi učitelem a studentem, tedy sekvence akcí a reakcí jak z učitelovy strany tak i ze studentovy, založená na vývoji studentových dovedností v anglickém jazyce. Cílem zpětné vazby je monitorovat a posléze kladně ovlivnit studentův vývoj a pokrok v těchto dovednostech. Zpětná vazba může mít mnoho forem. Například se může jednat o pochvalu, povzbuzení nebo opravu.

Podoba zpětné vazby se liší v závislosti na následujících faktorech: typ aktivity, fáze hodiny a individualita studenta. [4, s. 163]

Zpětná vazba na písemnou práci je odlišná v časovém aspektu v porovnání se zpětnou vazbou na jinou jazykovou dovednost – mluvení, jak HARMER tvrdí: „Pisatel také strádá tím, že neobdrží okamžitou zpětnou vazbu od čtenáře …“ [5, s. 53]. Navíc, jelikož je tato zpětná vazba obvykle psaná, se může stát, že si ji studenti ani nepřečtou, protože o ni nebudou mít zájem nebo důvod proč ji číst. HARMMER proto zdůrazňuje: „Učitelé se musí ujistit, že studenti rozumí problému (chybě, stylistické chybě atd.), a že danou pasáž textu správně přepíší.“ [6, s. 84].

Tento akt přepsání je zásadní, protože studenty nutí přečíst si zpětnou vazbu a studenti ji tak budou vnímat jako nástroj pro zlepšení jejich práce a jazyka. Psaná podstata zpětné vazby na písemnou práci ji také činí trvalou v časovém horizontu. Student má příležitost se znovu podívat na psanou zpětnou vazbu kolikrát touží či potřebuje.

Zpětná vazba na písemnou práci má ve většině případů podobu opravy. Pro učitele je totiž mnohem snazší soustředit se na aspekt jazyka (gramatiku) než na styl a obsah. Avšak většina studentů považuje za odrazující, pokud se jejich opravená práce topí v červeném inkoustu, jak se učitel pokusil opravit každou chybu. Přestože tento způsob opravování je mocným nástrojem, jak studentovi sdělit, že jeho angličtina je žalostná, je to také velice demotivující prostředek. Učitel proto musí najít rovnováhu mezi přesností oprav a opatrným přístupem ke studentovi, aby jej neodradil. [6, s. 84] Jako jeden z nástrojů k udržení této rovnováhy mohou posloužit delší komentáře, které by vždy měly v počátku zmínit pozitiva studentovy práce před kritikou.

1.3. Teorie psaní konceptů

Jedna z metod, která je aplikovaná při výuce psaní v anglickém jazyce je psaní konceptů (pracovních verzí). Na začátku student učitelovi odevzdá první hrubý návrh výsledné písemné práce. Ten je učitelem patřičně okomentován a opraven. Student pak využije těchto komentářů a oprav ke zdokonalení původního návrhu. Tento opravený návrh, který je teď již finální verzí, student odevzdá ke konečné kontrole.

Výhoda této metody spočívá v tom, že nutí studenta pracovat s opraveným textem. Učitel má tak jistotu, že student věnuje jeho poznámkám patřičnou pozornost. Navíc se tak student učí z vlastních chyb. Využití této metody bylo zohledněno při sestavování požadavků kladených na aplikaci.

2. Požadavky kladené na aplikaci

Požadavky kladené na program vychází z potřeb vyučujícího i studenta v souladu s obecně platnými metodickými zásadami psaní a opravování písemných prací. Z pohledu učitele se jedná především o zjednodušení práce – tj. snížení počtu úkonů, které je učitel nucen při hodnocení provést. Konkrétně jde o opakované psaní vysvětlivek a komentářů a počítaní chyb. Program by měl také zabránit nepřehlednosti opravené písemné práce, která nastává v případech, kdy učitel omylem opraví to, co je ve skutečnosti správně, a snaží se své značení napravit.

Z hlediska studenta byl návrh koncipován tak, aby zpřehlednil a prohloubil zpětnou vazbu poskytnutou učitelem. Zpřehledněním se míní konzistentnost a jednoznačná symbolika použitého značení.

S ohledem na potřeby učitele i studenta byly tedy stanoveny následující požadavky:

- Program by měl být schopen zpracovávat texty ve formátech produkovaných nejrozšířenějšími textovými editory a procesory.
- Program musí uživateli umožnit vkládání korekčních značek, komentářů a jejich snadné odstranění. Zároveň by měl umožnit i jejich snadné vytvoření a nastavení.
- Výstupem programu by měl být dokument obsahující i přehled týkající se počtu chyb, jejich závažnosti a počet slov v původním dokumentu.
- Program by měl umět sám vyhledat a označit lexikální chyby.
- Výstup obsahující značky musí být přehledný a zobrazitelný běžně dostupnými programy.

3. Koncepce řešení

Pro splnění podmínek uvedených v předchozí kapitole bylo použito toto řešení: Program je schopen načítat dokumenty ve formátu RichText (*.rtf), *Microsoft Word* Dokument (*.doc) a obyčejné textové soubory. Text v jiných formátech lze do programu přenést pomocí schránky operačního systému *Microsoft Windows* – pro tento případ program umožňuje zahájit práci s prázdným dokumentem. Pro výstup programu byl použit jazyk HTML a kaskádové šablony stylů, jelikož umožňuje jak vkládání značek a komentářů, tak i strukturování textu.

Použitím HTML a kaskádových šablon stylů se z pohledu studenta do značné míry zjednodušila manipulace s opraveným dokumentem – prohlížeč webových stránek je standardní součástí každého systému *Microsoft Windows*. Navíc se tím otevřela i cesta k použití interaktivních prvků, které mají na svědomí zjednodušení prohlížení opraveného dokumentu. Z pohledu učitele byla tímto způsobem zajištěna i snadná publikace zkontrolované práce na internetu s ohledem na moderní trendy v pojetí výuky – e-learning.

Opravování písemné práce probíhá vizuální formou. Uživatel převede text písemné práce studenta do aplikace, kde přímo vkládá korekční značky a komentáře. To znamená, že učitel ihned vidí, jak bude výsledná úprava vypadat, až si opravený dokument otevře student.

Pro opravování jsou k dispozici následující prostředky:

- **Korekční značky** (zjednodušeně značky) – Označování vybraného textu, kdy program za vybraný text přidá ještě uživatelem zvolenou zkratku názvu značky (symbol). Každá značka je po ukončení oprav opatřena patřičným textem (vysvětlivkou), který si opět uživatel nadefinuje. Značky lze využít jak k označení nedostatků a chyb, tak i k vyzdvihnutí pozitiv studentovy písemné práce.
- **Komentáře** – Opět vkládáme symboly do textu, ale na rozdíl od značek již bez označení textu. Jedná se tak o odkazy na samotné komentáře, které jsou rozsáhlejší než vysvětlivky u značek a jejich

použití směřuje k okomentování větších částí textu a závěrečnému zhodnocení studentovy písemné práce.

- **Statistika** – Je pouhý přehled počtu označených chyb a počtu slov v původní práci. Poskytuje zjednodušený přehled, jak si student vedl při plnění zadaného úkolu.
- **Kontrola pravopisu** – Je automatické vyhledávání pravopisných chyb a jejich následné označení značkou, kterou si uživatel sám k tomuto účelu zvolí.

Výsledný opravený dokument je možno exportovat jako webovou stránku, kdy jsou zachovány interaktivní prvky aplikované při opravě, nebo jako dokument PDF. Webové stránky může student otevřít libovolným internetovým prohlížečem – např. *Microsoft Internet Explorer* nebo *Mozilla Firefox.* V případě dokumentu PDF může student použít *Adobe Acrobat Reader*, který je možno zdarma získat na stránkách společnosti *Adobe* (www.adobe.com).

4. Aplikace „Essay Correction Assistant“

Program *Essay Correction Assistant* (ECA) je nástrojem, jenž si klade za cíl usnadnit opravování písemných prací v anglickém jazyce. Jeho vstupem jsou textové dokumenty, písemné práce studentů, které učitel za pomoci tohoto programu opraví. Posléze je opravené uloží jako interaktivní webové stránky nebo jako dokumenty PDF. Přestože program není editorem textů, je možno opravovaný text upravovat po obsahové stránce, nikoliv však po vzhledové – to pro účel aplikace není nutné.

Aplikace poskytuje širokou paletu možností nastavení značek i komentářů, jejímž cílem je nechat učitelovi volnou ruku při vytváření vlastního způsobu značení oprav. Pro usnadnění opravování program obsahuje lexikální kontrolu, tj. zdali dané slovo patří nebo nepatří do anglického jazyka, a přehled počtu chyb a počtu slov studentovy práce.

4.1. Využití

Program najde uplatnění při opravování libovolného textu v anglickém jazyce. Ať se již jedná o krátký odstavec nebo o delší písemnou stať. Podporuje interaktivní způsob výuky s využitím informačních technologií. Oprava dokumentu v programu ECA je rychlá v porovnání s tradiční metodou, kdy učitel vpisuje komentáře ručně přímo do studentovy práce. Učitel například může od studentů požadovat zaslání domácího úkolu elektronickou poštou a posléze opravené práce studentů publikovat na internetu. Studenti tak mají okamžitý přístup k opravené práci a nemusí čekat na další hodinu anglického jazyka. Učitel se takto také vyvaruje náročnému luštění škrabopisu studentů. Studenti zase získávají interaktivní opravený dokument bez zbytečného škrtání a s jasným popisem nedostatků a případných doporučení.

4.2. Požadavky pro spuštění

Pro svou činnost aplikace vyžaduje PC s operačním systém *Microsoft Windows 2000, XP* nebo *98*. Dále pak aplikaci *Internet Explorer 5.5* nebo vyšší (je standardně součástí operačního systému *Microsoft Windows*) a program *Microsoft Word*. Ten

není nutnou podmínkou, avšak bez něj není možné importovat do aplikace soubory typu dokument *Microsoft Word* `(*.doc)`.

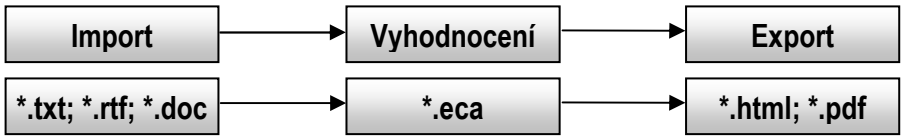
4.3. Instalace

K instalaci programu postačí spustit instalační soubor z instalačního CD a zbytkem instalačního procesu uživatele provede instalační průvodce. Během této procedury bude vyzván k zadání cílové složky instalace. Instalační CD obsahuje instalační soubor samotné aplikace a navíc i další dva instalační soubory aplikací *GhostScript* a *CutePDF Writer*, jež je nutné nainstalovat v případě, že uživatel požaduje, aby aplikace uměla export opravených dokumentů do souborů PDF. Je však možné použít i jiné virtuální tiskárny umožňující export do PDF a tím pádem instalaci těchto dvou aplikací vynechat. Obě doplňkové aplikace pro export do PDF je možno zdarma získat na adrese www.cutePDF.com.

4.4. Logika práce s programem

Aby uživatel mohl s aplikací začít pracovat, je více než vhodné seznámit jej se základními principy práce s tímto programem. Vstupem programu může být textový dokument formátu *RichText* `(*.rtf)`, *Microsoft Word Dokument* `(*.doc)`, nebo obyčejný textový dokument `(*.txt)`. Tento soubor je nutné nejdříve převést (importovat) na formát dokumentu ECA `(*.eca)` – tj. formát, se kterým aplikace dále pracuje. Je možné také vytvořit prázdný dokument ECA a následně do něj text vložit pomocí schránky operačního systému *Microsoft Windows*.

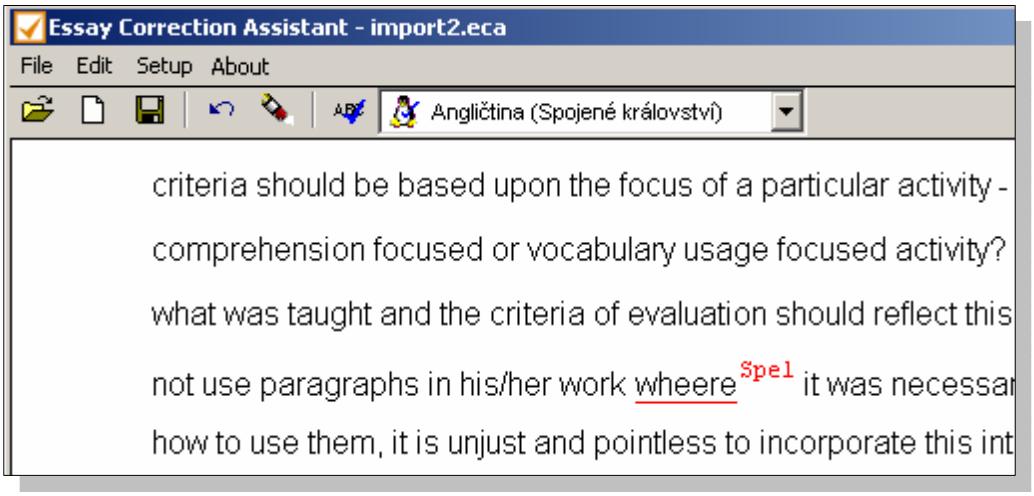
Následuje fáze vyhodnocování. Učitel provede opravu textu a opatří text svými komentáři. V této fázi je možno do dokumentu libovolně zasahovat. Rozpracovaný dokument může uživatel i uložit a dokončit opravu později. Nakonec uživatel opravený dokument vyexportuje buď jako webovou stránku `(*.html)`, nebo jako dokument PDF `(*.pdf)`. Po exportu již není možné nově vytvořený soubor nikterak upravit (míněno pomocí této aplikace). Je však možné provést změny v původním dokumentu ECA a ten pak opětovně vyexportovat. Celý proces názorně zachycuje následující diagram.



Obrázek 4.1: Diagram logiky práce s programem

4.5. Spuštění a ovládání programu

Program lze spustit buď pomocí zástupné ikony, kterou je možno při instalaci vytvořit na pracovní ploše, nebo otevřením souboru ECA. Po spuštění se uživateli zobrazí okno programu obsahující hlavní nabídku, panel nástrojů a editační okno, kde probíhá samotné opravování textu. Práce s editačním oknem je velice snadná. Pohybuje se v něm jako v okně běžného textového editoru. Je zde však jedna zvláštnost. Pokud chce uživatel ukončit řádek a začít psát na jiném, bez vytvoření nového odstavce, je nutné stisknout klávesy Shift a Enter najednou. Klávesa Enter totiž slouží k vytvoření nového odstavce. Editační okno je aktivní, pouze pokud je otevřen libovolný dokument ECA.



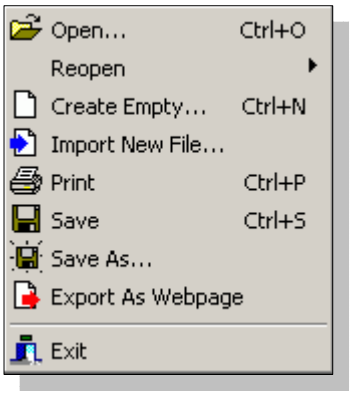
Obrázek 4.2: Aplikace Essay Correction Assistant

Hlavní nabídka se skládá z položek *File*, *Edit*, *Setup* a *About*. Pod položkou *File* lze nalézt volby pro manipulaci se soubory, položka *Edit* ukrývá jednoduché editační

nástroje, *Setup* pak umožňuje přístup k nastavení vlastností aplikace a konečně *About* zobrazí dialog obsahující informace o aplikaci.

Jednotlivé nabídky detailněji:

- **File** (soubor)
  - **Open** – Otevřít existující dokument ECA
  - **Reopen** – Obsahuje seznam odkazů posledních až pěti otevřených dokumentů ECA. Kliknutím na jeden z těchto odkazů se dokument otevře.
  - **Create Empty** – Vytvořit prázdný dokument ECA
  - **Import New File** – Importovat nový textový dokument jako dokument ECA
  - **Print** – Vytisknout dokument ECA[i]
  - **Save / Save As** – Uložit změny v aktuálním dokumentu ECA.
  - **Export As Webpage** – Exportovat dokument ECA jako webovou stránku
  - **Exit** – Ukončit aplikaci



Obrázek 4.3: Nabídka *File*

- **Edit** (anglicky úpravy)
  - **Cut** – Vyjmout text
  - **Copy** – Kopírovat text
  - **Paste** – Vložit text
  - **Find** – Najít řetězec

- **Setup** (anglicky nastavení)
  - **Marks** – Nastavení značek
  - **Comments** – Nastavení komentářů
  - **Application** – Další nastavení

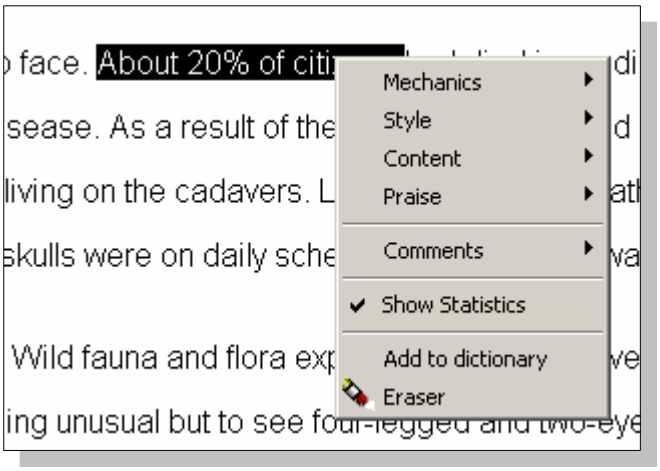
[i] V případě tisku dokumentu je text automaticky doplněn o legendu obsahující značky a odpovídající vysvětlivky.

Další částí hlavního okna aplikace je panel nástrojů, který je složen z tlačítek, jež umožňují snadnější přístup k některým položkám hlavní nabídky (konkrétně *Open, Create empty, Save*). Zároveň obsahuje i tlačítka s dalšími funkcemi.

Obrázek 4.4: Panel Nástrojů

První novinkou je tlačítko *Undo* (anulovat), jenž vrací poslední akci učiněnou uživatelem o jeden krok zpět. Obdobně funguje i tlačítko *Erase* (smazat), které vymaže značky a komentáře ve výběru. Následuje sekce pro kontrolu pravopisu, která obsahuje tlačítko *Spellchecker* (kontrola pravopisu) a menu pro výběr slovníku (viz obrázek 4.4), jenž se má při kontrole použít.

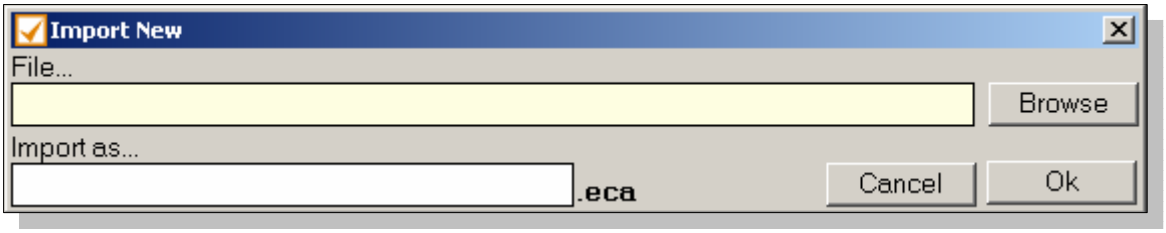
Nejdůležitější částí celého okna aplikace je však roletové menu, které se zobrazí po stisku pravého tlačítka myši. Obsahuje seznam všech značek roztříděných do kategorií, komentáře, volbu pro zobrazování statistiky, volbu přidat vybrané slovo do aktuálně zvoleného slovníku kontroly pravopisu a položku pro vymazání značení. Pomocí tohoto menu se provádí celá oprava písemné práce. Lze jej zobrazit pouze, pokud je otevřen libovolný dokument ECA.



Obrázek 4.5: Roletové menu

4.6. Import písemné práce

Prvním krokem opravy je import souboru, který chceme opravit. To je možno provést přes nabídku *File* v hlavním menu stisknutím tlačítka *Import New File.* V zobrazeném dialogu (viz obrázek 4.6) zvolíme importovaný soubor pomocí tlačítka *Browse* (procházet). Popřípadě ještě upravíme název souboru v editačním poli (*Import As*) podle toho, jak chceme pojmenovat nově vzniklý soubor. Poté výběr souboru potvrdíme. Výsledný soubor s příponou souborového typu aplikace ECA `(*.eca)` se uloží do implicitního adresáře pro import, který může být změněn v nastavení pod záložkou *Application.*



Obrázek 4.6: Dialog importu písemné práce

V případě, že uživatel chce vložit text z jiného typu dokumentu, než který je program schopen přímo importovat, lze využít volby *Create New* v nabídce *File* hlavního menu a do nově vzniklého dokumentu ECA text vložit pomocí schránky operačního systému *Microsoft Windows*. Text je možno vložit buď pomocí klávesové zkratky Ctrl + V, nebo zvolením volby *Paste* (vložit), která je v nabídce *Edit* v hlavním menu.

4.7. Práce s dokumentem ECA

Dokumenty ECA představují již importované písemné práce studentů. Tedy dokumenty, na které učitel aplikuje opravné nástroje. Tyto dokumenty je možno uložit a znovu otevřít. Učitel tedy nemusí dokončit opravu písemné práce okamžitě, ale má možnost dokument uložit a pokračovat v opravě později. Pro uložení dokumentu stačí, aby uživatel zvolil volbu *Save* nebo *Save As,* přeje-li si uložit dokument pod jiným názvem, v nabídce *File* hlavního menu. Uložený dokument je možné otevřít v téže nabídce zvolením volby *Open*, nebo použitím seznamu odkazů

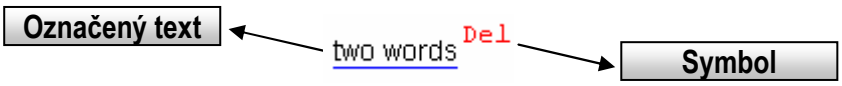
obsahujícím až pět posledních otevřených, nebo importovaných, dokumentů, který je možno nalézt pod volbou *Reopen*.

4.8. Prostředky pro vyhodnocování a jejich aplikace

To nejdůležitější, co program po funkční stránce nabízí, jsou prostředky pro vyhodnocování písemných prací studentů. Uživatel má k dispozici celkem čtyři nástroje: značky pro korekci, komentáře, statistiku a kontrolu pravopisu.

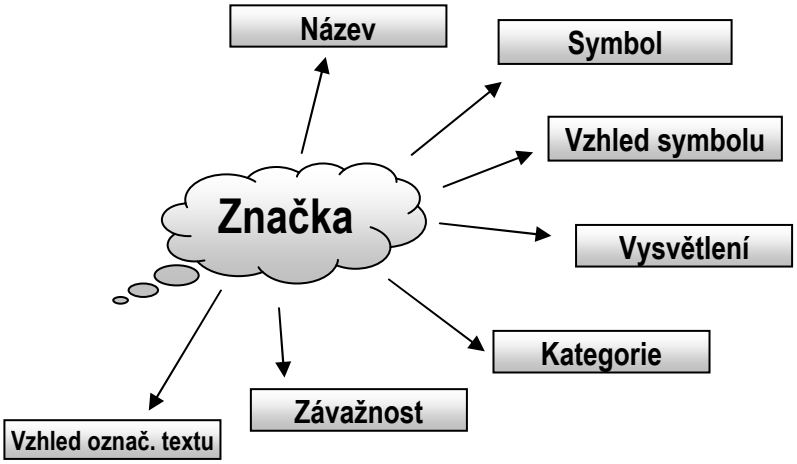
4.8.1. Značky a jejich použití

Základním prostředkem pro opravování prací v programu ECA jsou značky. Značka určuje vzhled vybraného textu a symbolu značky, který je k vybranému textu připojen. Značky slouží k vyznačování chyb a pozitiv písemné práce a lze je aplikovat na libovolnou část textu, například slovo, slovní spojení nebo větu. Po aplikaci značky bude příslušný úsek textu zvýrazněn (podtržením, přeškrtnutím, odlišnou barvou nebo stylem písma) a na jeho konci bude zobrazen symbol značky.



Obrázek 4.7: Názorný příklad použití značky

Pro rozlišení různých druhů chyb jsou k dispozici různé značky, které lze upravit. Učitel si také může vytvářet značky nové. Záměrem bylo vytvořit variabilní strukturu, která by umožnila učiteli výběr vzhledu značení a typu chyby.



Obrázek 4.8: Diagram struktury značky

Základem celé struktury je **název značky**, který musí být unikátní (tj. jediný svého druhu). Maximální délka názvu byla stanovena na dvacet znaků. Název značky se používá pro pojmenování příslušné položky v roletovém menu, přes které se značky aplikují. Druhým nejdůležitějším atributem značky je **symbol**, neboli řetězec, který by měl být zkratkou jména značky, a jenž se posléze přidá na konec označeného výběru – tedy místa, kde se má značka aplikovat. Symbol může mít nanejvýš čtyři znaky, aby se zabránilo přebujele dlouhým zkratkám. K volbě symbolu a názvu značky se váže i **vysvětlení**. Jinými slovy popis, který studentovi osvětluje, proč byla daná část textu označena. Tento popisek se aplikuje až při exportu dokumentu, aby zbytečně nenarušoval strukturu dokumentu během opravování.

Dalšími atributy značky jsou:
Kategorie, tedy jaké povahy je značka.

- Mechanismus (*Mechanics*) – Interpunkce, gramatika a větná skladba
- Styl (*Style*) – Stylistika a rozsah slovní zásoby
- Obsah (*Content*) – Logika, podpora tvrzení a koherence
- Pochvala (*Praise*) – Pozitiva studentovy práce

Závažnost značky

- Žádná (*None*) – Jedná se o pochvalu, nebo je nedostatek v textu studentovy práce bezvýznamný. Tato značka se neprojeví ve statistice[i].
- Méně závažná (*Minor*)
- Závažná (*Major*)

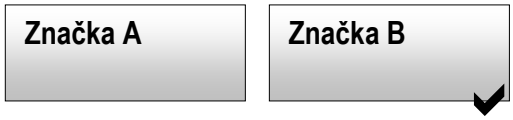
Důležitou částí značky je i **vzhled**, neboli jak se vizuálně projeví aplikace značky v textu. Ten lze rozdělit na dvě části: **vzhled označeného textu** a **vzhled symbolu**.

V případě **označeného textu** lze měnit barvu, styl (tučný, podtržený, přeškrtnutý, kurzívou.), barvu podtržení a styl podtržení (celá nebo přerušovaná čára). U **symbolu** může učitel ovlivnit následující: barvu, styl a formát (horní/dolní index, normální text).

Značky aplikujeme na označený text. U jednoho slova není nutné text značit. Postačí, pokud je pozice kurzoru v textu na slově, na které má být značka použita. Pro označení si již stačí vybrat požadovanou značku z roletového menu a program ji použije na vybraný text. Aplikace značek však podléhá určitým pravidlům.

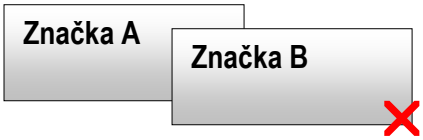
Je možné vnořit značku do značky, či i několik značek do jedné značky (viz obrázek 4.9). Samozřejmostí je i možnost použít v textu značky, které se sebou vůbec nekolidují (viz obrázek 4.10), avšak nesmíme značky vzájemně překřížit. (viz obrázek 4.11).

Obrázek 4.9: Vnořené značky



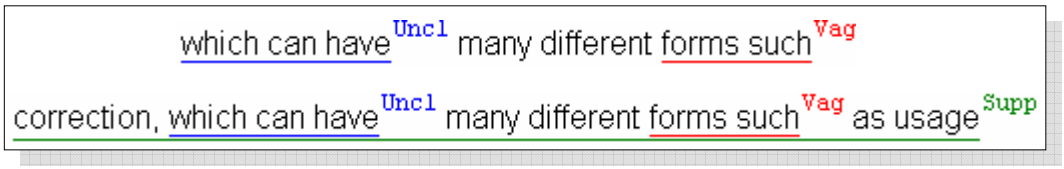
Obrázek 4.10: Nekolidující značky

[i] Více v části 4.8.3 „Statistika“



Obrázek 4.11: Překřížení značek

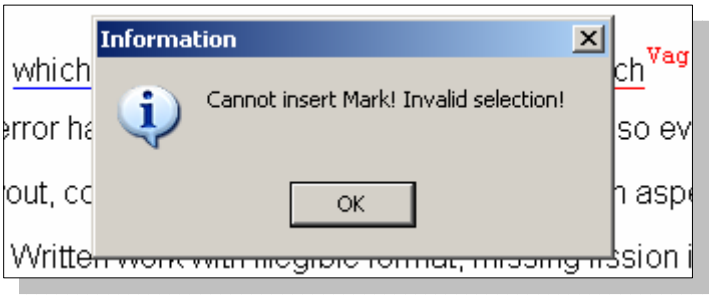
V praxi je tedy povoleno, aby se značky neprolínaly, nebo aby byly celé, tedy jak označený text tak i symbol, vnořené uvnitř jiné značky (viz obrázek 4.12).



Obrázek 4.12: Správný způsob značení

Pokud se uživatel pokusí vložit značku tak, že by tím značky překřížil (viz obrázek 4.13), program jej varuje (viz obrázek 4.14). Obdobným způsobem je uživatel varován, pokud překříží výběr při vymazávání značek – např. neoznačí celou značku.

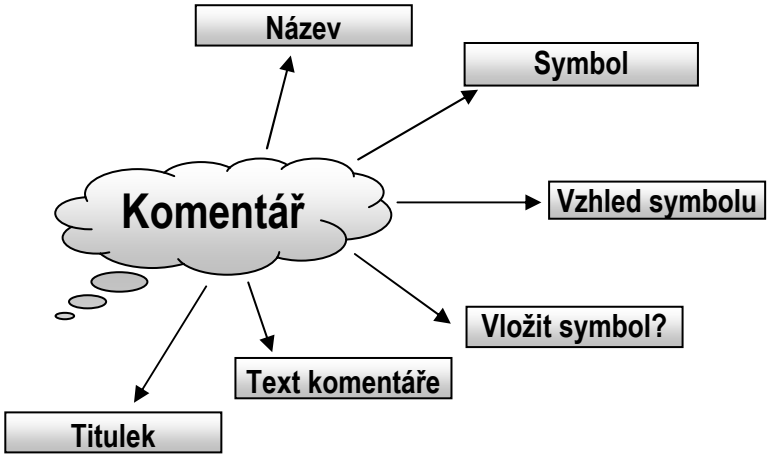
Obrázek 4.13: Nesprávně vybraný text pro vložení značky



Obrázek 4.14: Varování při překřížení značek

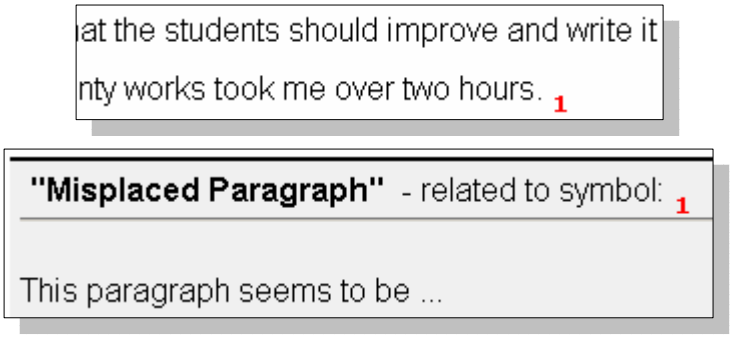
4.8.2. Komentáře a jejich použití

Komentáře jsou vlastně speciálním druhem značek, které nemají vliv na statistiku a vzhled označeného textu. Jedná se o pouhé vkládání symbolů – návěští, která odkazují na samotný komentář pod písemnou prací studenta. Komentáře slouží k závěrečnému zhodnocení, dodatečným doporučením či upozorněním. V případě jejich aplikace se za vybraný text přidá pouze symbol. Pokud není vybrán žádný text, symbol se vloží na aktuální pozici kurzoru v textu. Komentáře je možno vložit kamkoliv do textu. Struktura komentářů je následující:



Obrázek 4.15: Diagram struktury komentáře

Stejně jako u značky, i zde je klíčovým prvkem **název** komentáře, jenž musí být v rámci komentářů unikátní. Tento název se opět používá k pojmenování dané položky v roletovém menu. **Symbol** se vládá na dané místo v textu a odkazuje na samotný text, respektive na **titulek** (nadpis) komentáře. Pokud uživatel ponechá titulek prázdný, program vygeneruje automatický titulek. Na **vzhledu symbolu** lze upravit barvu, styl (tučný, podtržený, kurzívou, přeškrtnutý) a formát (horní/dolní index, normální text).



Obrázek 4.16: Ukázka použití komentáře v textu

Není nutné symbol do textu **vkládat**. Lze jej vynechat – tj. nevložit do textu. V tom případě se za text písemné práce přidá pouze text komentáře opatřený titulkem. Tento způsob je vhodný k závěrečnému zhodnocení práce studenta, kdy již vkládání symbolu není zapotřebí, jelikož se komentář vztahuje k celému textu a ne pouze k jeho části. **Text komentáře** lze upravit v mezích barevnosti, stylu a zarovnání (vlevo, vpravo a na střed).

4.8.3. Statistika

Dalším nástrojem, který usnadňuje hodnocení písemných prací je statistika. Jedná se o automaticky generovanou tabulku, která v sobě zahrnuje celkový počet slov studentovy práce, přehled počtu značek dle kategorií a závažnosti, a součet značek dle závažnosti. Je důležité mít na paměti, že počet slov je sečten pouze jednou, a to při importu dokumentu do aplikace ECA, u nově vytvořených dokumentů ECA před první akci značení (přidání značky, komentáře) nebo při prvním zobrazení statistiky. Zobrazení statistiky lze nastavit v roletovém menu, které se zobrazí po stisku pravého tlačítka myši. V tomto menu stačí kliknout na volbu *Show statistics*. Pokud je statistika zobrazená, tj. tato volba je v menu zatržená, program ji skryje. V opačném případě program statistiku zobrazí. U nově vytvořených prázdných dokumentů ECA, myšleno neimportovaných, je statistika automaticky skryta. Očekává se, že uživatel do takto vytvořeného dokumentu nejdříve vloží text. Toto opatření je nutné, aby program správně sečetl počet slov v dokumentu.

Jak již bylo řečeno, počty jednotlivých značek, které slouží k označení nedostatků a chyb, jsou shrnuty dle jejich kategorií – tj. mechanismus, styl, obsah (pochvaly se do statistiky nezahrnují). Přehled značek je také rozčleněn dle jejich závažnosti – tj. méně závažné, závažné. Značka se závažností nastavenou na *none* (žádná) se do statistiky nepromítne, rovněž tak se nezapočítávají ani komentáře. Výsledkem je tedy tabulka (viz obrázek 4.17), kde v prvním řádku jsou závažnosti značek a v prvním sloupci jsou položky:

- **Mechanics, Style, Content** – kategorie značek
- **Total** – celkový počet chyb dle závažnosti
- **Number of Words** – počet slov písemné práce

Importance and Number of Mistakes		
	Minor	**Major**
Mechanics	2	0
Style	0	0
Content	0	0
Total	**2**	**0**
Number of Words	**508**	

Obrázek 4.17: Ukázka výsledné statistiky

Samotná statistika rozhodně nemá být jediným prostředkem hodnocení. Její vypovídací hodnota je čistě orientačního rázu. Přesto však může poskytnout stručný, ale výstižný přehled, ve které hodnocené kategorii je práce studenta slabší a naopak, ve které kategorii není nutné závažných změn.

4.8.4. Kontrola pravopisu

Posledním prostředkem, jenž má usnadnit opravování, je kontrola pravopisu. Vzhledem ke složitosti a časové náročnosti vytváření modulu pro komplexní kontrolu pravopisu se program omezuje pouze na lexikální analýzu slov, tedy zdali použitý slovní tvar patří do anglického jazyka. Učitel může zvolit, zda se při kontrole použije slovník americké nebo britské angličtiny. Uživatel musí mít na paměti, že

slovníky dodané s aplikací nejsou dokonalé a ani nemohou být. Jazyk prochází neustálým vývojem, vznikají nová slova a spojení, tudíž se může stát, že slovník označí takováto slova za nesprávná, protože je neobsahuje. Je však možné je do slovníku přidat, a to buď označením daného slova a zvolením volby *Add to Dictionary* v roletovém menu, nebo v nastavení na záložce *Application* v části *Add Word to Dictionary.*

Kontrola pravopisu je jedna z prvních akcí, kterou by měl uživatel u nové písemné práce provést. Lze tak učinit stisknutím tlačítka *Check Spelling* umístěného na panelu nástrojů. Program provádí kontrolu od aktuální pozice kurzoru v textu níže, aby se zamezilo zbytečné kontrole již zkontrolovaného textu. Slova, která obsahují pravopisnou chybu, označí program příslušnou značkou, již si uživatel může zvolit v nastavení programu opět na záložce *Application* v části *Spellchecker*. Parametry této značky lze nastavit na záložce *Marks*. V případě, že není tato značka zvolena, není možné kontrolu provést – nebylo by možné text nikterak označit, tudíž by byla kontrola pravopisu zbytečná.

4.9. Nastavení programu

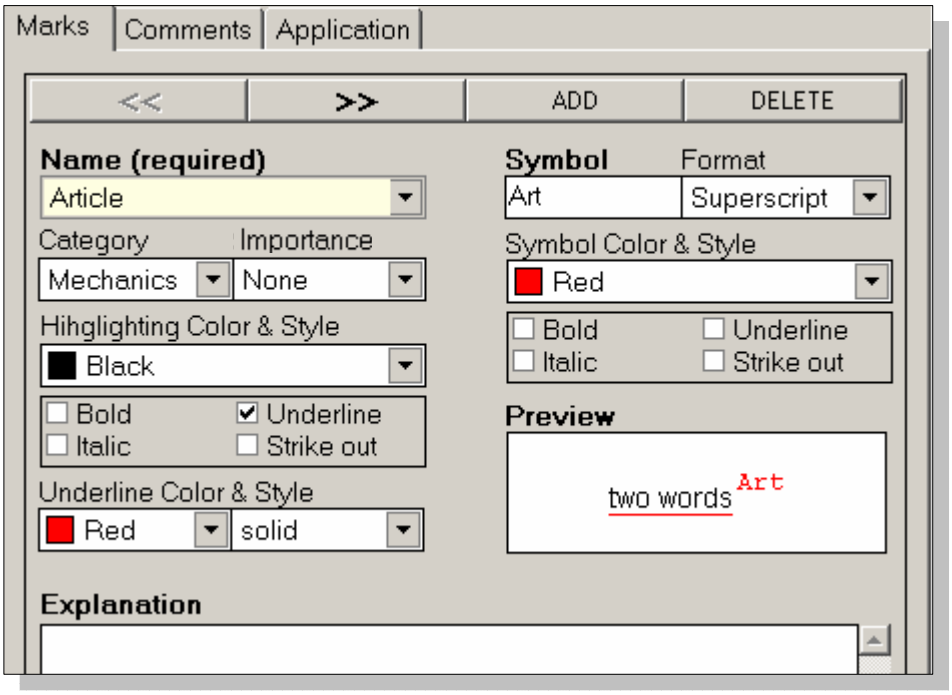
Veškeré úpravy parametrů programu je možno najít v hlavní nabídce pod záložkou *Setup* (nastavení). Tyto úpravy jsou rozděleny do třech oblastí:

- **Značky** – Tlačítko *Marks*
- **Komentáře** – Tlačítko *Comments*
- **Další nastavení** – Tlačítko *Application*

Po kliknutí na libovolné z těchto tlačítek se zobrazí patřičná záložka s panelem, který obsahuje příslušné prostředky pro konfiguraci dané oblasti.

4.9.1. Možnosti nastavení značek

Značky je možno přidávat (tlačítko *Add*) i odebírat (tlačítko *Delete*). Program je automaticky řadí dle abecedy. Základním prvkem každé značky je její název (*Name*), který může být o maximální délce dvaceti znaků. Zároveň se smí skládat pouze z písmen anglické abecedy a čísel. Každou značku musí uživatel přiřadit do jedné ze čtyř kategorií[i] (položka *Category*) a je nutné nastavit její závažnost (položka *Importance*). U značek z kategorie *Praise* (pochvala) je závažnost automaticky nastavena na žádnou (*none*) a nelze ji změnit. Značky s žádnou závažností se nezapočítávají do výsledné statistiky. Kategorie určuje, v jaké skupině značek se bude námi zvolená značka zobrazovat, a to jak v roletovém menu tak i ve statistice.



Obrázek 4.18: Panel nastavení značek

Dále je možné zvolit vzhled textu, na který je značka použita (*Highlighting Color & Style*). Konkrétně se jedná o barvu označeného textu, styl (tučný, kurzívou atd.) a

[i] Více v kapitole 4.8.1 „Značky a jejich použití“

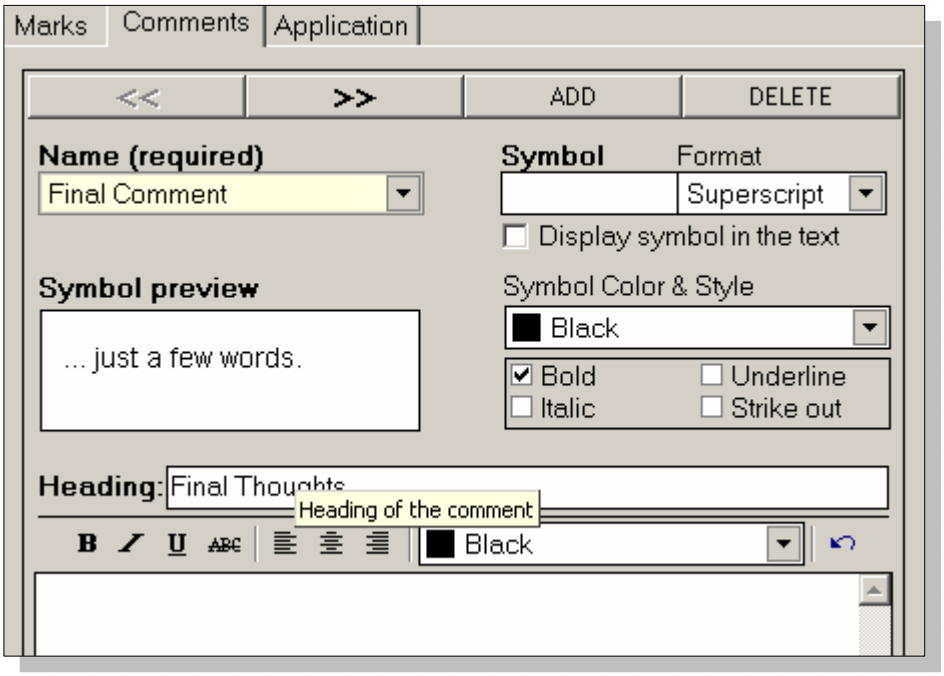
podtržení. U podtržení je možné nastavit jeho barvu, a zdali má být čára přerušovaná či plná (*Underline Color & Style*). Dalším určujícím prvkem je symbol, který se může skládat až ze čtyř znaků. Měl by být nejlépe zkratkou názvu značky. Tento symbol se pak při aplikaci značky vkládá do dokumentu. Změna symbolu se automaticky projeví i v opravovaném textu.

Podobně jako u změny vzhledu textu, na který byla aplikována značka, lze i u symbolu nastavit barvu a styl. Navíc je možné zvolit, zdali se má zobrazovat jako horní/dolní index, nebo jako normální text – tj. bez indexu. Výsledný vzhled celé značky je možno vidět v ukázce (okénko *Preview*). Poslední součástí každé značky je vysvětlení (*Explanation*). Jedná se o krátký popisek, který studentovi objasní význam značky. Je možné jej upravit pouze po obsahové stránce, nikoliv však po vzhledové. Popisek je aplikován až po exportu opravovaného dokumentu.

4.9.2. Možnosti nastavení komentářů

Panel komentářů je podobný panelu značek. Opět je možno komentáře přidávat a odebírat a program je také řadí abecedně. Parametry nastavení názvu komentáře a symbolu jsou téměř totožné. Podstatnou změnou je položka *Display symbol in the text* (Zobraz symbol v textu). V případě komentářů totiž není nutné vyplnit či použít, údaj symbol. Tato volba tak vytvoří komentář, který se nebude vztahovat k žádné části textu, ale k textu celému. U komentáře již není možné nastavit vzhled označeného textu, jelikož komentář se skládá pouze ze symbolu a komentáře, který se pak přidává na konec vybraného textu. U komentářů navíc není nutné vybírat text, symbol se přidá vždy na aktuální pozici kurzoru v textu.

Vlastní komentář má dvě části – titulek (*Heading*) a text komentáře (*Comment*). Obojí lze libovolně měnit po obsahové stránce, avšak pouze text komentáře lze též upravovat po vzhledové stránce – tj. styl, zarovnání a barva.



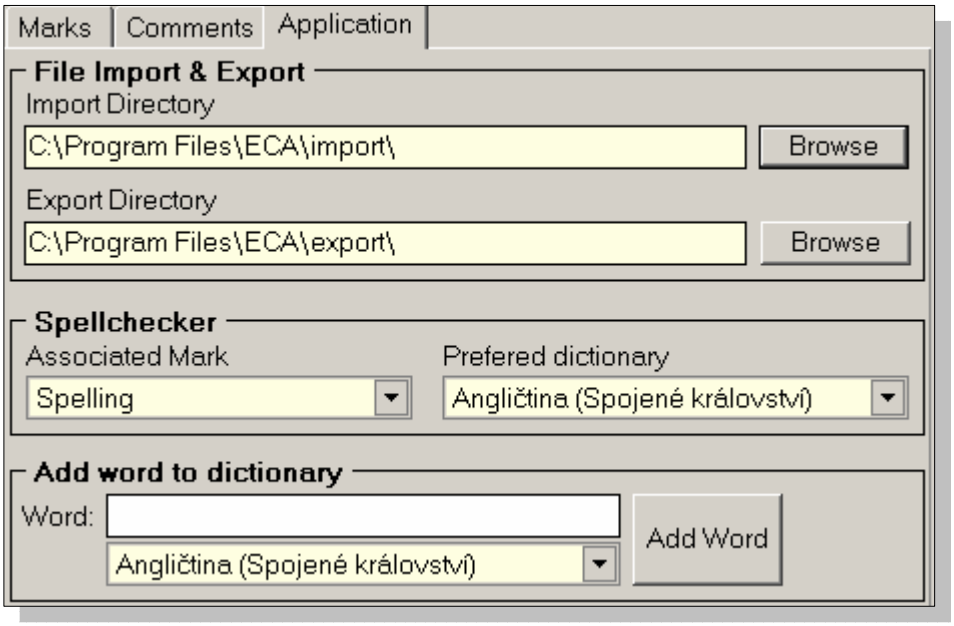
Obrázek 4.19: Panel nastavení komentářů

4.9.3. Ostatní nastavení

Pod záložkou *Application* se nachází poslední panel sloužící k nastavení dalších parametrů programu, který je rozčleněn na tři části. První část je věnována implicitním adresářům pro import a export souborů (*Import/Export Directory*). Nastavení *Import Directory* ovlivňuje cílový adresář, kam se mají ukládat soubory po importu. Podobně pak nastavení *Export Directory* určuje cílový adresář pro export souborů. Výběr těchto adresářů lze změnit stiskem tlačítka *Browse*.

V případě sekce věnované kontrole pravopisu (hláskování) lze nastavit dva atributy. Prvním je značka, která má být použita při této kontrole – *Associated Mark*. Tedy značka, kterou si uživatel vytvořil v nastavení *Marks* a nyní si přeje, aby byla použita k označení slova, které tato kontrola shledá chybným (nepatřícím do slovníku). Pokud není nastavena žádná značka pro kontrolu pravopisu, pak tato kontrola není možná. Druhé nastavení, *Preferred Dictionary*, pak slouží k nastavení slovníku, která uživatel upřednostňuje. Toto nastavení pouze ovlivňuje, jaký slovník bude nastaven ve výběru slovníku v panelu nástrojů po spuštění aplikace. Poslední sekce (*Add word to Dictionary*), přidání slov do slovníků, obsahuje tři části: pole

kam napsat přidávané slovo, výběr slovníku a tlačítko pro přidání slova do slovníku. Je velmi důležité, aby se uživatel ujistil, že slovo, které chce do slovníku přidat, je správě hláskováno. Akce přidání slov do slovníku je totiž nevratná!



Obrázek 4.20: Panel pro ostatní nastavení

4.10. Export písemné práce

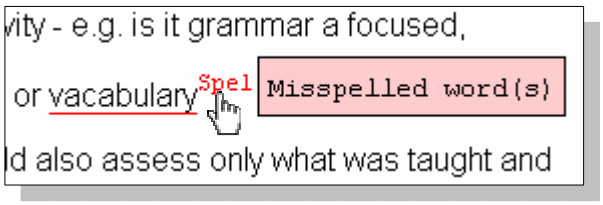
Export dokumentu ECA je poslední fáze opravy písemné práce. V této fázi je dokument ECA uložen jako soubor takového typu, aby si jej mohl snadno otevřít a prohlédnout i student. Uživatel má na výběr dva typy souborů. Prvním typem souboru je webová stránka. Tento formát souboru zaručuje snadnou prezentaci na internetu. Zároveň umožňuje aplikaci interaktivních prvků v dokumentu, jež usnadňují manipulaci s dokumentem ze strany studenta a zvyšují přehlednost dokumentu jako takového. Soubor tohoto formát si může student otevřít v libovolném prohlížeči webových stránek – např. *Internet Explorer* nebo *Mozilla Firefox*.

Druhým typem je dokument PDF (*Portable Document Format* – formát přenositelného dokumentu), tedy formát, který zaručuje stejný výsledek při zobrazení dokumentu na libovolném počítači. K jeho zobrazení je potřeba aplikace *Adobe*

Acrobat Reader, kterou je možné zdarma stáhnout z webových stránek společnosti *Adobe* (http://www.adobe.com).

4.10.1. Formát webové stránky

Pro export do webové stránky stačí zvolit položku *Export as Webpage* v nabídce *File* hlavního menu. Pak již zbývá jen vybrat, jak se má nový soubor jmenovat a kam se má uložit. Při exportu jsou veškeré značky opatřeny popisy (vysvětleními), které se automaticky objeví, pokud se kurzor myši nachází nad symbolem dané značky (viz obrázek 4.21). Zároveň se z veškerých symbolů komentářů stanou návěští odkazující na texty odpovídajících komentářů. Studentovi pak stačí na symbol kliknout, aby se k textu komentáře dostal. Poslední předností formátu webové stránky je jeho snadné publikování na internetu.



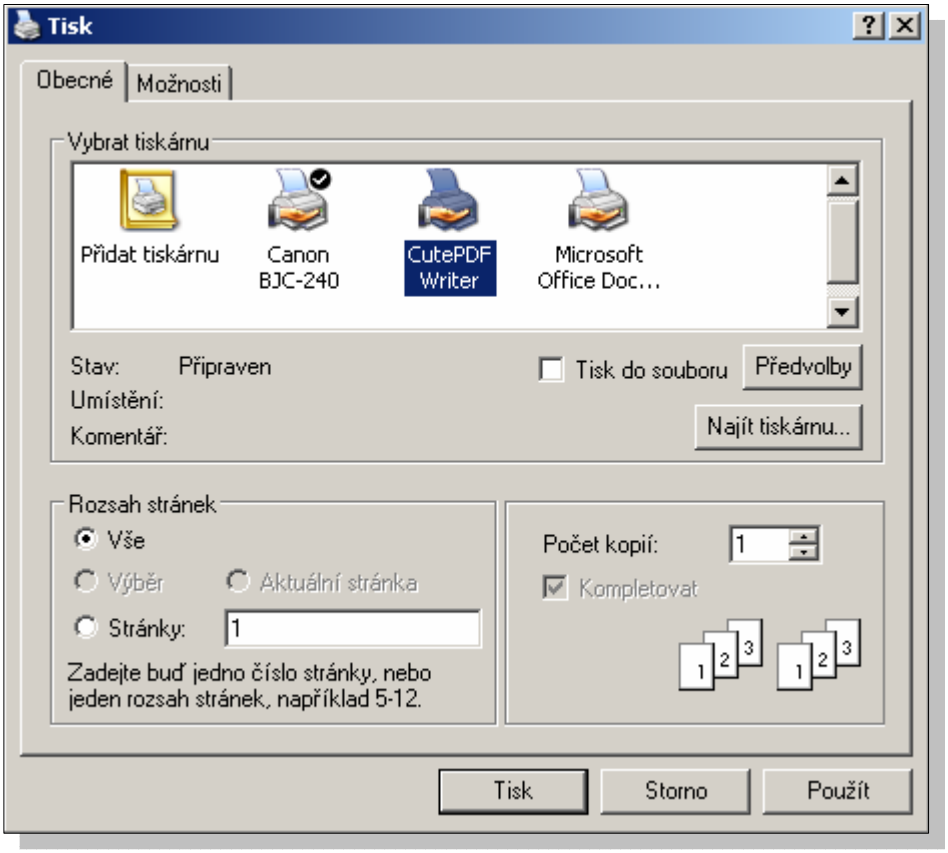
Obrázek 4.21: Ukázka popisu značky

4.10.2. Dokument PDF

Pokud si uživatel přeje opravený dokument uložit jako dokumentu PDF, měl by mít nainstalované aplikace *GhostScript* a *CutePDF Writer*, obě jsou k dispozici na instalačním CD. Tyto aplikace nainstalují virtuální PDF tiskárnu, kterou pak uživatel může používat i v jiných aplikacích. Je samozřejmě možné použít i jinou virtuální tiskárnu.

Export se spustí zvolením položky *Print* (tiskni) v nabídce *File* hlavního menu. V zobrazeném dialogu stačí zvolit tiskárnu *CutePDF Writer* (viz Obrázek 4.22) a stisknout tlačítko Tisk. Posléze se objeví standardní nabídka pro uložení souboru, ve které si uživatel může zvolit název souboru a místo kam se má uložit.

Výhodou formátu PDF je záruka stejného zobrazení dokumentu na všech PC, což v případě webových prohlížečů bývá někdy problém. Bohužel dokument přichází o interaktivní prvky – tj. automatické zobrazování popisu chyby a interaktivní návěští komentářů. Popisky jednotlivých značek jsou před uložením souboru připojeny ke konci dokumentu formou tabulky obsahující symboly a odpovídající vysvětlení.

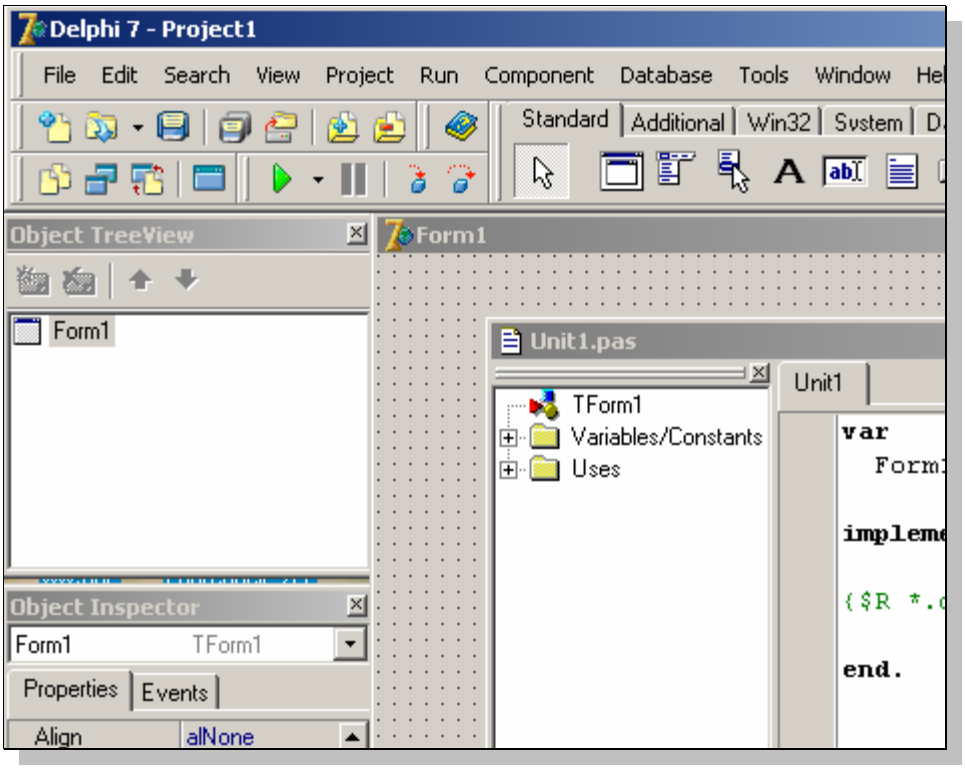


Obrázek 4.22: Dialog tisku

5. Vývojové prostředí Borland Delphi 7

Aplikace *Essay Correction Assistant* byla navržena a naprogramována v prostředí *Delphi 7 Personal Edition*. Jedná se objektově orientovaný nástroj pro tvorbu aplikací pro platformu *Microsoft Windows* založený na jazyce *Object Pascal*. O tomto prostředí by se daly napsat knihy, avšak pro potřeby této diplomové práce postačí zmínit pouze na některé jeho aspekty.

První verze prostředí *Delphi* byla uvedena na trh v roce 1995 a zaznamenala velký úspěch. Oproti ostatním vývojovým prostředím té doby, nabízelo prostředí *Delphi* maximálně zjednodušený návrh vzhledu programu. Navíc je zjednodušeno i psaní zdrojového kódu, tím že za programátora některé části kódu vyplňuje vývojové prostředí. V důsledku se pak programátor může více věnovat algoritmizaci řešení problému.



Obrázek 5.1: Vývojové prostředí *Delphi 7*

Programové prostředí *Delphi* tedy zahrnuje jednoduchou syntaxi jazyka *Object Pascal* a zároveň je vybaveno pokročilými nástroji pro vývoj komplexních aplikací.

Za všechny jmenujme alespoň mechanismus ošetření výjimek a jednoduché propojení s aplikacemi společnosti *Microsoft* – obojí bylo využito při tvorbě programu.

5.1. Programování v Delphi

Jak již bylo řečeno, programování v prostředí *Delphi* je po vzhledové stránce navrhované aplikace velmi snadné. Ruku v ruce s návrhem vzhledu jde i řešení programové části, která je zjednodušena širokou nabídkou předpřipravených komponent roztříděných dle typu do palet. Například k vytvoření tlačítka stačí zvolit komponentu *Button* (tlačítko) z palety *Standard* a umístit ji na vytvořený formulář.

Formulář je vlastně jakýmsi plátnem, na které programátor nakreslí celý vzhled aplikace, aniž by musel zasahovat do zdrojového kódu. Každou takto umístěnou komponentu lze dále upravovat v *Object Inspektoru,* a to ne jen po vzhledové a parametrové stránce, ale také po programové.

Práce s jednotlivými komponentami je založena na modelu událostí a vlastností. Událostmi se míní, co má program provést, když nastane určitá událost. Vlastnosti pak nastavují parametry komponenty – například velikost, umístění a jiné.

Vraťme se k příkladu tlačítka. Programátor umístí tlačítko na formulář a nastaví jeho vlastnost. Jednoznačně nejdůležitější událostí pro tuto komponentu je stisk tlačítka. V *Object Inspektoru* v záložce *Events* (události) zvolíme událost *OnClick* – po kliknutí. Delphi automaticky vloží prázdnou předpřipravenou proceduru do zdrojového kódu programu a provedou veškeré nutné změny v programu po syntaktické stránce. Programátor tak již pouze doplní samotnou akci, která se má provést.

Následující kód názorně ukazuje ošetření této události. Po stisku tlačítka se provede příkaz `showmessage` (viz řádek 3) a zobrazí se dialog se zprávou „Ahoj lidi!“ – programátorem jediný napsaný řádek.

```
1  procedure TForm1.Button1Click(Sender: TObject);
2  begin
3    Showmessage('Ahoj lidi!');
4  end;
```

Zdrojový kód 5.1: Ukázka obsluhy události

5.2. Objektově orientované programování

Jedná se o moderní, v současné době poměrně prosazovaný, způsob programování, jenž je inspirován skutečným světem. Objektem se rozumí datová struktura podobná záznamu – tedy obsahuje datové položky popisující objekt. Objekt je však ještě navíc opatřen metodami (funkcemi a procedurami), které nám umožňují s ním dále manipulovat. Za základy objektově orientovaného programování se považují následující tři principy: zapouzdření, dědičnost a polymorfismus.

5.2.1. Zapouzdření

Téměř každý předmět v reálném světě má předem určený způsob, jak s ním zacházet. Například automobil, pokud jej chceme nastartovat, otočíme klíčkem v zapalování. Rozhodně nelezeme pod kapotu se sirkami v ruce ve snaze učinit totéž. Jinými slovy – je zde jakýsi obal pravidel a návodů, jak to provést.

U objektů v programování je to podobné. Jsou zde nadefinované datové struktury, ke kterým máme určitá přístupová práva a metody, jež nám umožňují s objektem manipulovat – souhrnně rozhraní. Příkladem takového rozhraní u automobilu je funkce `Nastartuj` a atribut `PocetKm`. Pokud objektu automobil řekneme „`Nastartuj`“, objekt příkaz provede. Podobně nám atribut `PocetKm` sdělí, kolik kilometrů objekt automobil ujel. Objekt automobil sám určuje, kdy, a zda vůbec, je možné tento atribut změnit. Programátor tedy nikterak do objektu přímo nezasahuje, jen využívá objektem nabízené rozhraní. Program se tak stává nezávislý na vnitřní podobě daného objektu.

5.2.2. Dědičnost

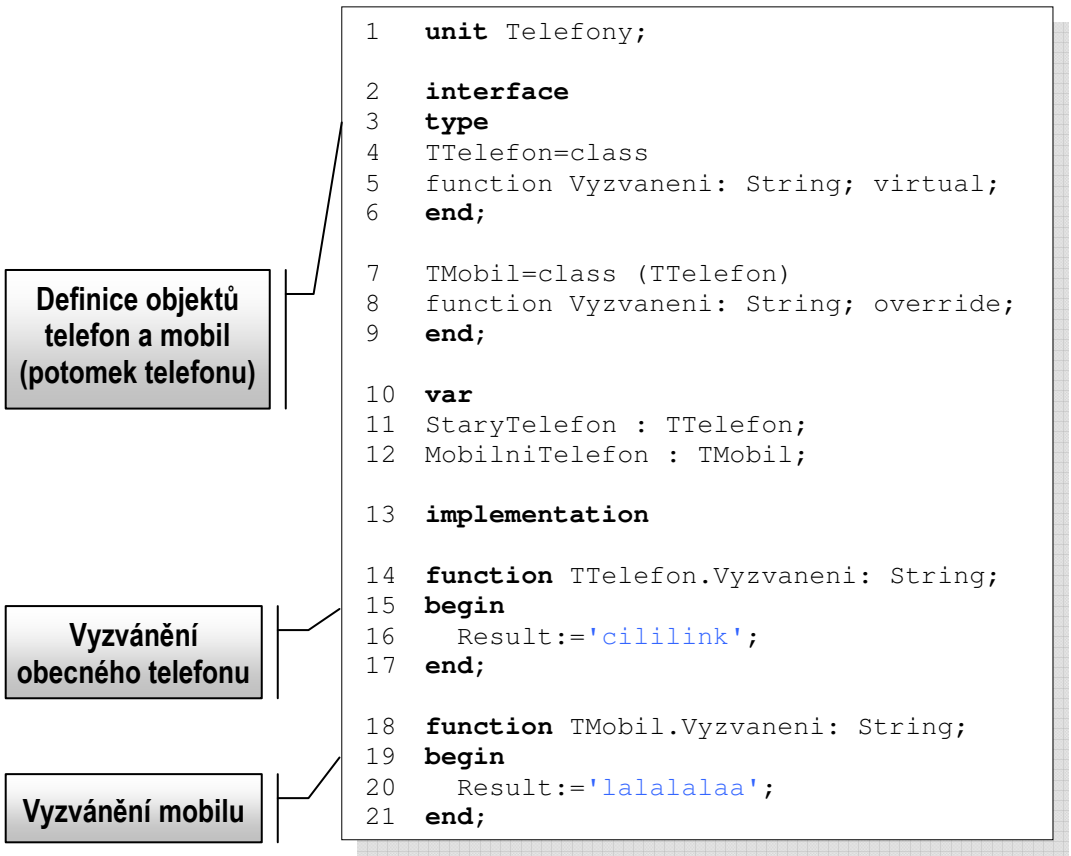
Jeden ze základních principů rozmnožování, ne-li ten nejdůležitější. Je běžné, že potomci dědí některé své vlastnosti po rodičích. Podobné je to i u objektů. Pokud vytvoříme nový objekt, jenž je potomkem jiného, zdědí potomek vlastnosti (tj. datové struktury a metody) po rodičovském objektu.

Máme-li tedy objekt `Savec`, jenž má nadefinovanou metodu `SajMléko`, pak potomek tohoto objektu s názvem `Opice` zdědí tuto metodu po něm. Navíc může mít

proceduru `SkákejPoVětvích`, která bude specifická pouze pro objekt Opice a potomky tohoto objektu.

5.2.3. Polymorfismus

Posledním ze základů objektově orientovaného programování je polymorfismus. Tento princip je úzce spjat s dědičností. V praxi je často potřeba nedefinovat metodu, jenž bude pro určitou skupinu objektů společná, ale způsob jejího provedení se bude lišit dle objektu. Nejdříve vše ukáži na příkladu zdrojového kódu v jazyce *Delphi*, který podrobněji popíši.




```
1   unit Telefony;

2   interface
3   type
4   TTelefon=class
5   function Vyzvaneni: String; virtual;
6   end;

7   TMobil=class (TTelefon)
8   function Vyzvaneni: String; override;
9   end;

10  var
11  StaryTelefon : TTelefon;
12  MobilniTelefon : TMobil;

13  implementation

14  function TTelefon.Vyzvaneni: String;
15  begin
16    Result:='cililink';
17  end;

18  function TMobil.Vyzvaneni: String;
19  begin
20    Result:='lalalalaa';
21  end;
```

Zdrojový kód 5.2: Ukázka polymorfismu

Jak je patrné, byly vytvořeny dvě třídy objektů[i], `TTelefon` a `Tmobil` (viz řádky 4 a7). Třída `TMobil` je potomek třídy `TTelefon`. Oběma třídám byla přiřazena funkce `Vyzvaneni`. Posléze byly deklarovány i samotné objekty, respektive proměnné odkazující na objekt: `StaryTelefon`, kterému byla přiřazena třída `TTelefon` a `MobilniTelefon` s třídou `TMobil`. Samotné objekty je nutno ještě zinicializovat pomocí konstruktoru. Po řádku `implementation` následuje popis funkce `Vyzvaneni`, který se pro `TTelefon` a `Tmobil` liší. Přestože je `Tmobil` potomkem `TTelefon`, tedy měl by zdědit tu samou funkci `Vyzvaneni`, je výsledek jeho funkce `Vyzvaneni` odlišný, i přes stejné jméno.

V praxi by pak volání funkce `Vyzvaneni` vypadalo následovně:

```
1   StaryTelefon:=TTelefon.Create;
2   MobilniTelefon:=Tmobil.Create;

3   showmessage('Telefon zvoní ', StaryTelefon.Vyzvaneni);
4   showmessage('Mobil zvoní ', MobilniTelefon.Vyzvaneni);

5   StaryTelefon.Free;
6   MobilniTelefon.Free;
```

Zdrojový kód 5.3: Ukázka použití objektů

Nejdříve jsou objekty přivedeny k životu metodou `create` (viz řádky 1 a 2). Tato metoda slouží jako konstruktor v *Delphi*. Jejím cílem je alokovat místo v paměti a zinicializovat objekt. V následujících dvou řádcích jsou volány funkce `Vyzvaneni` jednotlivých objektů a jejich výsledky jsou zobrazeny uživateli ve formě textové zprávy. Pro objekt `StaryTelefon` bude tato zpráva vypadat následovně: „Telefon zvoní cililink". Obdobné je znění této zprávy i pro `MobilniTelefon`: „Mobil zvoní lalalalaa". Posledním krokem je uvolnění objektů pomocí metody `free`, která slouží jako destruktor (viz řádky 5 a 6).

[i] V Delphi se nejdříve definuje třída. Objekt je až posléze instancí třídy. Tedy vztah třída – objekt je analogií vztahu datový typ – proměnná.

5.3. Ošetření výjimek a běhových chyb v Delphi

Jedno ze zlatých pravidel programování říká, že každý program obsahuje alespoň jednu chybu. Na většinu z nich upozorní překladač, jsou však chyby, které se objeví teprve za běhu aplikace. Pokud se jedná o fatální chybu, pak se běh aplikace velmi rychle změní v pád, s patřičným hlášením systému, uživateli většinou nic neříkajícím, jenž obsahuje stručné vysvětlení, k jaké chybě došlo. Samozřejmě, že je možné pokusit se odladit chyby pomocí podmínek, či jiných programových struktur, ale ne vždy tato snaha přináší ovoce. Navíc podmínky mohou kód programu velmi zkomplikovat. Naštěstí *Delphi* mají komplexní mechanismus, jak tyto běhové chyby ošetřit a zabránit tak celkovému kolapsu aplikace. Jedná se o velice jednoduchý mechanismus inspirovaný metodou „pokus-omyl“. Základem je slůvko `try` (zkusit), tedy program něco zkusí, a když to nevyjde, udělá něco jiného. Komplementem tohoto slůvka jsou výrazy `finally`, `except on` a `raise`.

- `try` – uvozuje chráněný blok kódu, tedy tu část, kterou chceme ošetřit mechanismem výjimek
- `except on` – je ukončení chráněného bloku, jenž je zároveň počátkem kódu, který má chybu ošetřit
- `finally` – uvozuje blok kódu po `try`, který se má provést i v případě běhové chyby. Tento blok kódu je tedy proveden vždy.
- `raise` – slouží k vyvolání výjimky

Následuje příklad ošetření chyby při dělení nulou:

```
1   function VydelDveCisla (a, b: integer): integer;
2   begin
3     try
4       Result := a div b;
5     except on EDivByZero do
6       begin
7         Result := 0;
8         Showmessage('Nulou dělit nelze!')
9       end;
10    end;
11  end;
```

Zdrojový kód 5.4: Příklad ošetření chyby

Jak je patrné, jedná se o jednoduchou funkci, jenž má vydělit dvě čísla a vrátit výsledek. Příkazem `try` začíná chráněný blok programu, kde se provede pokus o dělení. V případě chyby dělení nulou (viz řádek č. 5) se provede blok příkazů na řádcích sedm a osm. Podobným způsobem se využívá příkaz `finally`, který také následuje po `try`. Je možné použít jak `except on`, tak i `finally` při ošetřování stejného bloku kódu.

6. Programová realizace aplikace

Tato kapitola se věnuje programovému řešení aplikace, čtenář v ní však nebude zatěžován konkrétními pasážemi zdrojového kódu a popisem, k čemu to či ono slouží. Spíše se seznámí s podstatou programu, tedy co bylo použito k dosažení cíle a jak. Zdrojový kód programu je přiložen na instalačním CD, takže i zvědavci dychtící vědět, jak vše funguje, si přijdou na své.

Návrh programu vycházel z myšlenky použití HTML dokumentů a kaskádových šablon stylů jakožto prostředků pro prezentaci a editaci písemné práce studenta. Jazyk HTML určuje strukturu dokumentu a kaskádové styly jeho vizuální prezentaci. Bylo proto nezbytné využít nástroje, který by umožnil přístup k těmto technologiím a jejich integraci. Zároveň bylo nutné zaručit bezproblémovou konverzi textového dokumentu na dokument HTML. V obou těchto případech byla využita technologie komponentového objektového modelu.

6.1. Komponentový objektový model

Jedná se o technologii pro konstrukci softwarových komponent, která byla představena firmou Microsoft v roce 1993. Využívá se k meziprocesové komunikaci a vytváření dynamických objektů v jakémkoliv programovacím jazyce podporujícím tuto technologii. Komponentový objektový model (*Component Object Model* – COM) se často ve světě používá jako pojem zastřešující termíny OLE, ActiveX, COM+ a DCOM. Přestože COM byl implementován na různých platformách, jeho primární využití je cíleno pro operační systém Microsoft Windows. [3]

Zjednodušeně řečeno jde o vytváření objektových komponent, malých programových fragmentů, které poskytují služby ostatním aplikacím, operačnímu systému a jiným komponentám (souhrnně klientům) přes komponentami definované rozhraní. V podstatě si tyto komponenty můžeme představit jako objekty obsahující určité funkce, které jsou k dispozici ostatním aplikacím (přesněji programátorům těchto aplikací). Tyto komponenty jsou pak umístěny v paměti a čekají na své využití, tedy až si je klient zavolá. Je zde jistá podobnost s komponentami ve vývojovém prostředí *Delphi*, kde používáme komponenty ke stavbě programu.

Podstatný rozdíl je však v jazykové nezávislosti. To znamená, že komponentu modelu COM vytvořenou ve *Visual C++* může použít například programátor v Delphi. Výhoda tohoto přístupu spočívá ve využití již hotových komponent, které lze jednoduše začlenit do nové aplikace nezávisle na tom, v jakém prostředí byla komponenta navržena. Tím se zjednodušuje práce programátorů. [11]

6.1.1. Historie

Od prvních verze Windows, která umožňovala běh více aplikací zároveň, byl kladen požadavek na jejich vzájemnou komunikaci. Výsledkem splnění tohoto požadavku byl velmi jednoduchý nástroj, který má však široké uplatnění i dnes – *clipboard* (schránka).

Evolučním rozšířením schránky byla technologie *Dynamic Data Exchange* (DDE), která měla umožnit výměnu informací mezi spuštěnými aplikacemi. Principem DDE bylo zasílání zpráv mezi DDE klientem (jednou spuštěnou aplikací) a DDE serverem (druhou spuštěnou aplikací). DDE komunikace se členila do témat (*topics*) a položek (*items*); celý mechanismus je založen na zasílání zpráv systému Windows. Mechanismus DDE již není v moderních aplikacích využíván. [10]

Následovaly technologie *Object Linking and Embedding* (OLE) a OLE2, v překladu „Vkládání a propojování objektů“, které z DDE vycházely. Zjednodušeně lze říci, že OLE je mechanismus přinášející schopnost vložit do aplikace objekt definovaný v jiné aplikaci. Vše funguje tak, že se uvnitř jedné aplikace (OLE klienta) otevře okno druhé aplikace (OLE serveru), obecně nazývané OLE kontejner, které poskytuje funkčnost OLE serveru uvnitř OLE klienta. Později byly některé části OLE přejmenovány na *ActiveX* a později se takto přejmenovala v podstatě celá technologie OLE – stalo se tak spíše z marketingových důvodů než z inovačních.

Ačkoliv technologie OLE2 byla postavena na komponentní technologii, její využití bylo pouze jednoúčelové (propojení dokumentů). Teprve v roce 1995 byla představena technologie, která umožnila vývoj aplikací založených na komponentním základu – COM.

6.1.2. Propojení s aplikací Microsoft Word

Jedno z praktických využití modelu COM, konkrétně technologie *OLE automation*, je právě propojení programu v *Delphi* a aplikace *Microsoft Word*, které bylo využito ke konverzi dokumentu této aplikace do formátu HTML. Aby bylo samotné propojení možné, je nutné použít modul `ComObj` v sekci `Uses` programu a zároveň deklarovat proměnou typu `Variant`, pomocí níž je pak realizováno vytvoření OLE objektu `Word.Application`. Posléze již pak standardní tečkovou konvencí voláme metody vzniklého OLE objektu. Můžeme tak například zakázat zobrazování varovných zpráv programu, nebo skrýt okno samotné aplikace. Právě pomocí takového propojení byl umožněn přístup k dokumentům vytvořených v aplikaci *Microsoft Word*.

```
1   Uses ComObj;

2   Var WordApp : Variant;

3   Procedure MainF.ImportIt;
4     try wordapp := CreateOleObject('Word.Application');
5     except on exception do
6       begin
          MessageDlg('Can''t import file. Application MS Word not
          found!', mtWarning,[mbOk],0);
          exit;
7       end;
8     end;

9     wordapp.DisplayAlerts := false;
10    wordapp.Visible := false;
11    wordapp.documents.open(pathToFile);
12    …
13    wordapp.activedocument.close;
14    if (not (VarIsEmpty(WordApp))) and
15    (IsObjectActive('Word.Application')) then
16    wordapp.quit;
17  end;
```

Zdrojový kód 6.1: Spuštění a ukončení aplikace *MS Word*

6.1.3. Komponenta Webbrowser

Klíčovým prvkem celého programu je komponenta vývojového prostředí *Delphi* nazývaná *Webbrowser*, která se nachází v paletě *Internet*. Jedná se o vizuální komponentu, která umožňuje využití aplikace webového prohlížeče *Microsoft Internet Explorer*, tj. jeho funkcí, v programátorem nově navrhované aplikaci. V podstatě se jedná o prvek *ActiveX*, tedy opět využití technologie COM v praxi, importovaný do vývojového prostředí *Delphi* v podobě komponenty. Tato komponenta zprostředkovala vizuální přístup k editaci dokumentů v jazyce HTML, který byl nezbytný k realizaci konceptu programu. K této komponentě se ještě později vrátíme. Nejdříve se však podíváme pod pokličku HTML.

6.2. HTML

HTML je zkratkou anglického termínu *Hypertext Markup Language*, což lze přeložit jako značkovací jazyk pro hypertext. Abychom mohli odtajnit samotnou podstatu HTML, je nutné se alespoň v krátkosti zmínit o jeho základu, jímž je hypertext.

6.2.1. Hypertext

Jedná se o způsob prezentování informací, kdy jedna informace odkazuje na jinou. V praxi hypertext umožňuje uživateli získat doplňkové informace o předešle zvolené informaci, jimiž mohou být definice nebo materiály vztahující se k vybranému slovu. Hypertextové odkazy, vedoucí k takovým informacím, tvoří rozvětvenou nebo síťovou strukturu, která povoluje přímý, nezprostředkovaný skok k související informaci. Hypertext je neodmyslitelnou součástí World Wide Webu, což je nejvyužívanější služba v prostředí internetu. Je to způsob, jakým jsou jednotlivé webové stránky (dokumenty) provázány mezi sebou, poskytující tak obsažnou strukturu informací, která zabraňuje zahlcení uživatele nepřiměřeným množstvím informací. [8]

6.2.2. Historie

Za jeden ze zlomových roků vývoje internetu je považován rok 1990, kdy Tim Berners-Lee a Robert Caillau, vědci v Evropském centru pro jaderný výzkum (CERN), navrhli jednoduchý jazyk, který umožňuje vytvářet a publikovat multimediální a elektronické dokumenty v prostředí internetu. Zrodil se tak jazyk HTML. Do té doby se k těmto účelům používal *Tex*, *PostScript* nebo SGML, avšak tyto jazyky byly pro publikaci dokumentů na internetu složité. Navíc s příchodem HTML došlo ke sjednocení jednotlivých elementů v jednom dokumentu, takže autoři již nemuseli publikovat dokumenty jako sestavu samostatných obrázků, textu a popřípadě zvuků. Od té doby prošlo HTML určitým vývojem, jehož výsledkem je v současné době prosazovaný jazyk XHTML založený na HTML 4. 01. To však neznamená, že se jazyk HTML již nepoužívá. [12, s. 3] [7]

6.2.3. World Wide Web Consortium

Se záměrem zaručit standardizace definice jazyka HTML bylo vytvořeno *World Wide Web Consortium* (W3C). Toto konsorcium zodpovídá za návrh specifikací standardu. Navíc se snaží reflektovat potřebu většiny dle ohlasů z internetu a dle těchto komentářů a připomínek standard dále upravovat.

Konsorcium W3C není zodpovědné pouze za HTML, ale i za další technologie související s World Wide Webem – tedy související s publikací a úpravou hypertextových dokumentů. Z těch nejdůležitějších jmenujme alespoň kaskádové šablony stylů a technologii XML (*Extensible Markup Language* – rozšiřitelný jazyk pro značkování). Zájemci o vývoj těchto standardů se mohou informovat o aktuálním vývoji přímo na stránkách W3C (http://www.w3c.org). [12, s. 7]

6.2.4. HTML: Co to je

HTML je nástrojem pro určení rozvržení dokumentu a hypertextových odkazů (též hyperlinků). Poskytuje definici příkazů, které se v prohlížeči těchto dokumentů přímo nezobrazují, ale které řídí způsob výsledného zobrazení struktury dokumentu,

a to včetně textu, obrázků a ostatních podpůrných médií. [12, s. 7] Jinými slovy HTML je jazyk, který stanovuje, jak vytvořit HTML dokument skládáním HTML elementů (prvků) a jak tyto elementy vypadají.

6.2.5. Syntaxe jazyka HTML

Jazyk HTML se skládá z párových a nepárových značek a atributů těchto značek, které se vkládají do dokumentu za účelem definování jeho struktury. Tyto značky se také někdy nazývají tagy (z anglického *tag* – visačka, cedulka). Částem textu uzavřených do párových značek se říká elementy dokumentu. Elementy je možné do sebe zanořovat. Není však možné elementy vzájemně zkřížit.

V případě párových značek rozlišujeme značky koncové a počáteční. Mezi tyto značky je pak uzavřena část dokumentu – element. Každou značku obklopují úhlové závorky, aby se jednoznačně oddělily od dokumentu. Navíc u koncových párových značek následuje po první úhlové závorce ještě lomítko. V následujícím příkladu je vidět použití párových značek pro vytvoření odstavce.

```
<p> Toto je nový odstavec. </p>
```

Zdrojový kód HTML 6.1: Příklad označení odstavce

Podobně se implementují do dokumentu i nepárové značky. S tím rozdílem, že se použije pouze počáteční značka. Jednou z takových značek je i horizontální dělící čára. V příkladu následujícího HTML kódu je názorně vidět i použití atributů u značek, konkrétně atributu *size* (velikost). Tento atribut ovlivňuje výslednou tloušťku této čáry.

```
… a tak končí náš první odstavec. </p>
<hr size=4>
<p> Alespoň může začít další …
```

Zdrojový kód HTML 6.2: Příklad nepárové značky

Kromě dělení na párové a nepárové lze značky rozdělit dle funkce na strukturální, sémantické a stylistické. Strukturální značky se používají k rozvržení dokumentu. Tedy například určují, co je odstavec, co je nadpis atp.. Sémantické blíže popisují význam obsahu části dokumentu uvnitř těchto značek. Například značka `<title>` označuje titulek stránky. Tento druh značek urychluje vyhledávaní v záplavě složitých dokumentů. Posledním typem jsou stylistické značky, které definují vzhled při zobrazení. Příkladem těchto značek je párová značka `<b>`, která se používá pro tučné písmo. V současné době je však trend tyto značky nepoužívat. Namísto toho se výsledný vzhled dokumentu doporučuje upravovat pomocí více flexibilních a vzhledově pestřejších kaskádových šablon stylů.

Dokument může mimo značkování obsahovat i další prvky:

- Direktivy – začínají znaky `"<!"`. Ty jsou určeny pro zpracovatele dokumentu, kterým je prohlížeč. (*Mozilla Firefox, Internet Explorer* atd.)
- Komentáře – Poznámky programátora. Ty jsou prohlížečem ignorovány.
- Kód skriptovacích jazyků
- Definice událostí a kód pro jejich obsluhu

6.2.6. Struktura HTML dokumentu

Od verze HTML 4.01 musí každý dokument začínat odkazem na deklaraci dokumentového typu (*Document Type Definition* – DTD), který říká v jaké verzi HTML je dokument napsán. V dokumentu je uveden pomocí klíčového slova `DOCTYPE`. Dle daného typu DTD prohlížeč pozná, jaké atributy jsou pro elementy v dokument povoleny. Následuje pak párový kořenový element `<html>` (`<html>` dokument `</html>`), který prohlížeči neříká nic jiného, než že následující obsah je v jazyce HTML. Dále se dokument dělí na hlavičku a tělo. Hlavička obsahuje meta-data (data popisující data) vztahující se k celému dokumentu. Definují například název dokumentu, jazyk, kódování, klíčová slova, popis, nebo použitý styl zobrazení. Hlavička je uzavřena mezi značky `<head>` a `</head>`. A konečně tělo dokumentu, které je uzavřeno mezi značky `<body>` a `</body>`, obsahuje samotný dokument. [7]

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
      <title>Můj název dokumetu</title>
   </head>
   <body>
      <h1>Nadpis stránky</h1>
      <p>Toto je tělo dokumentu</p>
   </body>
</html>
```

Zdrojový kód HTML 6.3: Ukázka HTML dokumentu

6.3. Kaskádové šablony stylů

Kaskádové šablony stylů (*Cascading Style Sheets* – CSS) slouží k popisu způsobu prezentace (zobrazení, tisku) HTML dokumentu. Jedná se o poměrně obsáhlé téma, proto se v této kapitole omezím pouze na technické pozadí nutné k pochopení, jakým způsobem byly kaskádové šablony stylů využity při programování aplikace. Zájemcům, kteří chtějí hlouběji proniknout do tajů kaskádových šablon stylů, vřele doporučuji webové stránky

http://www.w3schools.com/css/

Jazyk HTML se zaměřuje převážně na strukturu, obdobně se CSS soustředí právě na vzhledovou a prezentační stránku dokumentu. CSS je tedy vlastně jakýsi souhrn pravidel přiřazený určitému dokumentu, který určuje, jak se má dokument, konkrétně jeho elementy, prezentovat.

Vše funguje na velice jednoduché, leč geniální myšlence. Struktura dokumentu je rozčleněna pomocí HTML značek. CSS těmto značkám přiřadí určitý způsob prezentace. Například značce `<p>`, tedy každému odstavci, je přiřazeno písmo modré barvy o velikosti 12. V podstatě se tedy pro tuto značku vytvořilo pravidlo, které prohlížeči říká, jak ji má prezentovat. Tudíž odpadá zdlouhavé manuální nastavování patřičných atributů pro každou takovou značku, abychom docílili téhož. Navíc lze jeden styl přiřadit více dokumentům.

6.3.1. Přiřazení CSS v HTML dokumentu

Styl lze značce přiřadit třemi způsoby:

- Styl přiřadíme rovnou značce (tzv. řádkové styly)
- Styl přiřadíme na úrovni dokumentu
- Použijeme odkaz na externí styl

Styly přímo přiřazené značce vkládáme přes atribut `style` dané značky. Stačí, když za tento atribut vypíšeme vlastnosti značky. Jedná se o nejjednodušší způsob spojení stylu se značkou. Následující příklad ukazuje přiřazení stylu značce, kdy styl přebarví text na červeno.

```
<h1 style="color: red"> Nadpis1 <h1>
```

Zdrojový kód HTML 6.4: Přiřazení stylu značce

Styl přiřazený na úrovni dokumentu je prosté začlenění sekce s pravidly prezentace do hlavičky dokumentu. Tato sekce je ohraničena párovými značkami `<style>` a `</style>`. Tento způsob, který byl zvolen pro aplikaci ECA, již ukazuje skutečnou sílu CSS v praxi – jedna definice vzhledu pro více stejných značek.

```
<style>
  p {font-family: Arial;
     color: blue;
     line-height: 1.25;}

  h1 {font-size: 16px;
      color: red;}
</style>
```

Zdrojový kód HTML 6.5: Styl na úrovni dokumentu

V případě externího stylu se do hlavičky dokumentu zapíše odkaz ukazující, kde se soubor s definicemi stylu nachází. Způsoby připojení takového stylu jsou celkem dva: propojení pomocí příkazu `<link>` nebo pomocí příkazu `@import`. Pro potřeby této práce však není nutné blíže popisovat rozdílnosti těchto dvou příkazů, ani se blíže jimi zabývat vzhledem k tomu, že tento způsob nebyl při programování aplikace použit.

6.3.2. Třídy a pseudotřídy

U většiny elementů HTML je možno zadat i atribut, který jej zařadí do určité skupiny zvané třída (z angličtiny `class`). Tento atribut je jen jakýmsi pojmenováním, které nám umožňuje u jednoho typu elementu definovat více rozdílných stylů. Následující příklad demonstruje použití tříd u elementu `<p>`.

```
…
<style>
   p.text {color: black;}
   p.poznamka {color: blue;}
</style>

</head>
<body>
<p class=text>
   Toto je odstavec s textem.
</p>
<p class=poznamka>
   A tohle je odstaveček s nějakou nejapnou poznámkou.
<p>
</body>
</html>
```

Zdrojový kód HTML 6.6: Třídy elementů a CSS

U hyperlinků je možné navíc použít tzv. pseudotřídy, které upravují prezentaci odkazu dle jeho aktuálního stavu. Na rozdíl od normálních tříd se neoddělují tečkou, ale dvojtečkou. Navíc, názvy pseudotříd jsou předem nadefinované, a to takto:

- **link** – nastavuje zobrazení okazů, které uživatel ještě nevybral
- **visited** – určuje vzhled těch odkazů , které uživatel již navštívil (vybral)
- **active** – ovlivňuje zobrazení okazů, jenž uživatel vybral a právě je prohlížeč zpracovává
- **hover** – definuje vzhled odkazů, nad kterými se právě nachází (vznáší) kurzor

V praxi pak lze nastavit způsob zobrazení odkazu tak, že nenavštívené odkazy budou modré, navštívené černé a ty, nad kterými se právě nachází kurzor zeleně. Výsledná definice takového chování odkazů by vypadala následovně.

```
a:link {color : blue;}
a:visited {color : black;}
a:hover {color : green;}
```

Zdrojový kód HTML 6.7: Pseudotřídy v CSS

6.3.3. CSS v aplikaci „Essay Correction Assistant“

Jednoduchost a elegantnost CSS byla využita právě při definici vzhledu a prezentace značení, komentářů a statistiky. Odpadla tak nutnost definovat vzhled každé vložené značky samostatně a zároveň bylo umožněno snadné upravení vzhledu značení pro celý dokument. Navíc jsou pomocí CSS zhotoveny i vyskakující vysvětlivky u korekčních značek, které se studentům objeví, pokud se nad danou značku vyskytuje kurzor myši.

Kaskádová šablona stylu je do každého dokumentu vložena v jeho hlavičce a v případě změny je aktualizována. Samotný proces aktualizace stylu se postará i o odstranění těch značek, které uživatel z definic značení v nastavení programu odstraní. Výsledný styl je k nahlédnutí v libovolném opraveném dokumentu. Stačí tento dokument exportovat jako webovou stránku, otevřít jej v libovolném webovém prohlížeči a nechat si zobrazit zdrojový kód.

6.4. MSHTML

Jak již bylo zmíněno, klíčovou roli při sestavování programu hrála vizuální komponenta *Webbrowser,* která je součástí prostředí *Delphi*. Tato komponenta umožnila přístup k vizuální editaci HTML dokumentu. To však není zcela přesné. Komponenta *Webbrowser* totiž v podstatě zprostředkovává přístup k jiné objektové komponentě společnosti *Microsoft*, a tou je MSHTML (též známá jako *Trident*). MSHTML je například součástí aplikace *Microsoft Internet Explorer*, kde určuje výsledné rozvržení dokumentu na obrazovce uživatele. Také umožňuje upravování webových stránek (tedy HTML dokumentů) v jakémkoli prostředí podporujícím COM, a tím Delphi jsou. Hlavní předností MSHTML je, že v kombinaci

s komponentou *Webbrowser* umožňuje editaci HTML dokumentů jako WYSIWYG[i] editor. [16] Aby programátor mohl MSHTML využít, je nutné v sekci `uses` programu připsat následující deklarace: `MSHTML`, `SHDocVw`.

Následující příklad demonstruje práci s komponentou *Webbrowser* a MSHTML v prostředí *Delphi*. Nejdříve je přepnuto prohlížení dokumentu na jeho editaci (viz řádek 1). Posléze je pak text nacházející se ve výběru ztučněn.

```
1   (WebBrw.Document as IHTMLDocument2).designMode := 'On';
2   (WebBrw.Document as IHTMLDocument2).execCommand('Bold',False,0);
```

Zdrojový kód 6.2: Práce s komponentou Webbrowser a MSHTML

6.4.1. Praxe

Mohlo by se zdát, že práce s MSHTML je pak velmi snadná, bohužel opak je pravdou. Asi největším úskalím je, že zdrojový kód webové stránky je při použití MSHTML přeložen a opraven. Příkladem může být získání části zdrojového kódu z právě označené části dokumentu. Výsledkem volání příslušné funkce není ta část HTLM, ale validní část HTML. To znamená, že když ve výběru máme jenom jednu z párových značek elementu, druhá je automaticky přidána na konce či začátek vráceného řetězce – MSHTML se pokusilo získanou část HTML opravit. To samo o sobě není nic strašného, ale při pokusu vrátit řetězec zpět do dokumentu se programátor nestačí divit. Vlivem překladu totiž vznikne další odstavec navzdory původnímu očekávání, že tato akce nebude mít na strukturu dokumentu žádný vliv. Podobné chování nastává téměř při jakémkoli zásahu do zdrojového kódu dokumentu – tj. dokument projde fází překladu a MSHTML se pokouší odstranit syntaktické chyby.

Kromě doplňování a opravování HTML tagů se při fázi překladu projeví ještě další nemilá vlastnost MSHTML. Zdrojový kód je naformátován. Takže například veškeré identifikátory tagů dokumentu jsou z malých písmen automaticky převedena na velká. Což v podstatě znemožňuje použití XHTML, jelikož tento jazyk požaduje

[i] Z anglického „What you see is what you get", neboli co v editačním okně vidíte, to bude výsledek.

malé písmena u těchto identifikátorů. Formátování se bohužel netýká jen tagů, ale celého rozvržení HTML kódu, tj. formátovacích znaků zpřehledňujících výsledný kód. Následující příklad demonstruje zmíněné problémy.

Původní kód s označeným výběrem

```
<body>
   <p> Ahoj lidi jak se máte? </p>
</body>
```

Výběr vrácený MSHTML

```
<P> Ahoj lidi jak</P>
```

Výsledný kód po vložení výběru zpět

```
<BODY>
<P> Ahoj lidi jak</P><P> se máte? </P>
</BODY>
```

6.5. Další využité prostředky

6.5.1. Speller

Speller je balíček komponent umožňující přidání kontroly pravopisu do téměř libovolné aplikace vyvíjené v *Delphi 7*. Umí využít některé slovníky aplikace *Microsoft Word*, ale hlavně používá slovníky aplikace *ISpell*, což je program pro kontrolu pravopisu původně pro platformu *Unix*. Právě pomocí balíčku *Speller* byla do aplikace *Essay Correction Assistant* implementována kontrola pravopisu. Zájemci si mohou tento balíček stáhnout z www.luziusschneider.com/SpellerHome.htm.

6.5.2. CutePDF Writter

CutePDF Writer je virtuální tiskárna, která umožňuje ukládání dokumentů ve formátu PDF v libovolné aplikaci podporující tisk. Při spuštění tisku přes tuto tiskárnu dojde k otevření standardního dialogu, který vyzve uživatele k uložení daného dokumentu jako soubor formátu PDF. Právě spolupráce s tímto programem umožňuje aplikaci *Essay Correction Assistant* export opravených dokumentů jako soubory PDF. Aplikace *CutePDF Writer* však ke své činnosti potřebuje program

Ghostscript. Oba tyto programy lze získat na adrese www.cutePDF.com a jsou samozřejmě i na instalačním CD.

6.5.3. Inno Setup Compiler

Ve snaze usnadnit uživatelům instalaci aplikace, byl zvolen skriptovací program pro tvorbu instalačních balíčků – *Inno Setup Compiler*. Jedná se o pokročilý program s komplexním nastavením instalace libovolné aplikace, který je možno zdarma získat na stránkách www.jrsoftware.org. Svou výbavou se může měřit i s mnohými komerčními produkty.

7. Závěrem

Aplikace ECA je nástrojem, který má usnadnit opravování písemných prací v anglickém jazyce. Ke splnění tohoto cíle využívá vizuální aplikaci korekčních značek a komentářů. Součástí aplikace je i kontrola pravopisu, která umožňuje automatické označení slov nepatřících do anglického jazyka. Aplikace také poskytuje uživateli souhrnný přehled o počtu nedostatků a jejich závažnosti v písemné práci studenta. Vstupem programu může být v podstatě libovolný text, který je na konci celého procesu opravování uložen buď jako dokument PDF, nebo jako webová stránka.

7.1. Využití v praxi

Vzhledem k variabilitě korekčních značek a komentářů je možné použít aplikaci na všech stupních škol. Autor sám vyzkoušel program na Střední škole gastronomie a služeb v Liberci, kde se setkal s pozitivním ohlasem ze strany studentů. Jediným problémem ze strany studentů může být přístup k počítači a internetu, který není zatím zcela běžně dostupný pro všechny studenty. V některých případech se také studenti při odevzdávání písemných prací vymlouvají na nefungující internet, či jiné technické komplikace s počítačem. Ukázkové opravené práce studentů je možno nalézt na přiloženém CD ve složce *Sample Written Assignments*.

Aplikace byla také testována PeadDr. Zuzanou Šaffkovou, M.A., CSc, která působí jako lektorka na katedře anglického jazyka fakulty pedagogické v Liberci. Program byl použit pro korekci písemných prací studentů 2. ročníku anglického jazyka během kurzu psaní v letním semestru roku 2006. Výsledkem celého procesu testování není pouze zlepšení výsledného programu po funkční stránce, ale i reflexe poskytnutá touto lektorkou na výsledný program ve formě dotazníku, který je k nahlédnutí v příloze č. 1.

Literatura

[1] CANTÙ, M.: *Myslíme v jazyku Delphi 6 (2. díl)*. 1. vydání, Praha, Grada Publishing, 2002. ISBN 80–247–0335–1

[2] CHASTIAN, K.: *Developing Second-Language Skills: Theory and Practice*. 3. vydání, Orlando, Harcount Brace Jovanovich, 1988. ISBN 0–15–517619–6.

[3] *Component Object Model - Wikipedia, the free encyclopedia* [online]. c2006, poslední revize 10. 2. 2006 [cit. 15. 2. 2006]. <http://en.wikipedia.org/wiki/Component_object_model>.

[4] GOWER, R., PHILLIPS, D., WALTERS, S.: *Teaching Practice Handbook***.** 2. vydání, Oxford, Heinemann, 1995. ISBN 0–435–24059–5.

[5] HARMER, J.: *How to Teach English*. 1. vydání, Harlow, Longman, 1998. ISBN 0-582-29796-6.

[6] HARMER, J.: *The Practice of English Language Teaching*. 20. vydání, Harlow, Longman, 1990. ISBN 0–582–74612–4.

[7] *HTML - Wikipedie, otevřená encyklopedie* [online]. ©2006, poslední revize 5. 4. 2006 [cit. 18. 4. 2006]. <http://cs.wikipedia.org/wiki/Html>.

[8] *hypertext - Encyclopædia Britannica* [online]. ©2006, [cit. 20. 4. 2006]. <http://www.britannica.com/eb/article?tocId=9367686>.

[9] KADLEC, V.: *Učíme se programovat v Delphi a Object Pascal*. 1. vydání, Praha, Computer Press, 2001. ISBN 80–7226–245–9.

[10] KADLEC, V.: Umíme to s Delphi. *166. díl – Komponentový model Microsoft COM* [online]. ©1999, č. 166. [cit 2. 4. 2006]. Dostupné z <http:// www.zive.cz/h/Programovani/>.

[11] KADLEC, V.: Umíme to s Delphi. *38. díl – Technologie OLE jasně a srozumitelně, 1. část* [online]. ©1999, č. 166. [cit 2. 4. 2006]. Dostupné z <http:// www.zive.cz/h/Programovani/>.

[12] MUSCIANO, C., KENNEDY, B.: *HTML a XHMTL: Kompletní průvodce*. 1. vydání, Praha, Computer Press, 2000. ISBN 80–7226-407–9.

[13] SATRAPA, P.: *Pascal pro zelenáče*. 3. vydání, Praha, Neokortex, 2000. ISBN 80–86330–03–6.

[14] SMALLEY, R., REUTTEN, M.: Refining Composition Skills: Rhetoric and Grammar for ESL Students. 3. vydání, Boston, Heinle & Heinle Publishers, 1990. ISBN 0-8384-3385-5.

[15] SVOBODA, L. et al.: *1001 tipů a triků pro Delphi*. 2. vydání, Praha, Computer Press, 2003. ISBN 80–7226–488–5.

[16] *Trident (layout engine) - Wikipedia, the free encyclopedia* [online]. ©2006, poslední revize 17. 3. 2006 [cit. 22. 3. 2006]. <http://en.wikipedia.org/wiki/MSHTML>.

[17] WHITE, R., ARNDT, V.: *Process Writing*. 4. vydání, Harlow, Longman, 1994. ISBN 0–582–02444–7.

Příloha č. 1: Dotazník

Tento dotazník byl vyplněn lektorkou anglického jazyk na katedře anglického jazyka fakulty pedagogické v Liberci PeadDr. Zuzanou Šaffkovou, M.A., CSc, která testovala aplikaci ECA v letním semestru roku 2006 během kurzu psaní studentů 2. ročníku anglického jazyka.

1 V čem podle Vás spočívá přínos této aplikace?

Přínos aplikace zaměřené na opravování písemných projevů studentů je možné charakterizovat z několika rovin. Jednak je aplikace součástí on-line výuky, která kombinuje jak práci ve třídě, tak samostatné studium na počítači. Studenti pracují s řadou materiálů s cílem zvládnout některé požadavky samostatně a dále prokázat formou vypracování úkolů, že nastudovanou látku pochopili a jsou schopni ji aplikovat v nových situacích. Psaní statí je součástí této kombinované výuky a je tedy logické, že i kontrola a hodnocení písemného projevu studentů je realizována ve stejném on-line systému. Další důležitou skutečností a je i zpětná vazba, kterou studenti obdrží na svou e-mailovou adresu. Vzájemná interakce mezi učitelem a studentem probíhá plynule tak, jak studenti odevzdávají své práce a on-line komunikace umožňuje i několikanásobnou zpětnou vazbu. Tím, že studenti pracují na svých úkolech v čase, který není omezen pouze výukou ve třídě a s materiálem, který přesahuje pouze tištěnou formu, má učitel možnost poskytnout studentům daleko bohatší podněty pro jejich práci.

Samotná aplikace ECA je efektivním pomocníkem při opravování esejí, především z hlediska jednoduchého označování jazykových chyb, kalkulace pozitivních i slabších stránek vyjadřování v anglickém jazyce i podpůrné vizuální úpravy formátu opraveného textu. Pouhý pohled na opravený text naznačuje, do jaké míry student práci zvládl po stránce formální. Také možnost doplnění komentářů přímo do textu nebo konečné zhodnocení na závěr opravy naznačuje jak učiteli, tak především samotnému studentovi, jak práci zlepšit nebo naopak které úseky eseje jsou velmi dobře zpracované.

Další výhodou aplikace je jednoduchá statistika chyb, která patří k jednomu z kritérií hodnocení písemných prací studentů. Kromě toho, že vyučující okamžitě po ukončení opravy eseje vidí jasný přehled počtu gramatických, stylistických a některých obsahových chyb, může také z tohoto přehledu vyčíst, zda se jedná o chyby zásadní nebo menšího významu.

2 Jak hodnotíte ovládání a práci s aplikací?

Práce učitele s aplikací se týká dvou základních úkonů. Jednak je to její použití na opravování konkrétních prací studentů a jednak je to práce na soustavném doplňovaní a tedy i zlepšování možností aplikace. Po krátkém zapracování do jednotlivých úkonů aplikace je postup výběru označení chyby a vložení do textu záležitostí několika vteřin. S rostoucím počtem esejí se jednotlivé úkony stanou rutinní záležitostí, která spočívá v zapamatování umístění označení jednotlivých typů chyb a komentářů. Kromě vložení nových nebo závěrečných komentářů, které není možné zařadit do již vytvořeného seznamu komentářů vzhledem k jejich specifickému (někdy i osobnímu) obsahu týkající se dané eseje, spočívá veškerá práce s aplikací v ovládání myši.

Také doplnění nebo úpravy značek a komentářů aplikace je jednoduché. Postup jednotlivých úkonů je vizuálně podpořen náhledem, ve kterém jsou jednotlivé kroky jasně znázorněny, takže je možné je okamžitě zkontrolovat a popřípadě opravit.

Výhodou aplikace je také automatická kontrola správnosti manipulace s programem, která okamžitě upozorní uživatele o chybném kroku.

3 Jsou nabízené nástroje pro korekci a jejich možnosti nastavení dostačující?

Vzhledem ke zkušebnímu využití aplikace v rámci jednoho semestru je možné shrnout, že nabízené nástroje pro korekci a jejich možnosti nastavení jsou dostačující. Dodatečné úpravy a doplnění bude záležitostí další práce s aplikací, která ukáže, v jakých směrech bude nutné možnosti nastavení rozšířit nebo upravit.

Opravování esejí je do určité míry rutinní záležitost, pro kterou je možné nastavit konstantní parametry, na druhou stranu se jedná o individuální a osobní přístup k písemným projevům studentů a je třeba, aby hodnocení tyto jednotlivosti a specifiku odráželo. Toto bude vždy vyžadovat flexibilitu ze strany učitele tak, aby aplikace byla efektivním pomocníkem, který usnadňuje a urychluje opravování, ale zároveň neomezovala, umožňovala individuální přístup.

4 Kde vidíte prostor pro zlepšení? Co Vám v aplikaci chybí?

V současné době je jedinou nedořešenou záležitostí jednoduché odeslání opravených esejí do systému Moodle[i], ve kterém kurz psaní probíhá.

[i] Moodle je systém pro zprávu elektronických kurzů a výuky v prostředí internetu, který katedra angličtiny používá pro správu on-line kurzů.

Příloha č. 2: Obsah přiloženého CD

Nedílnou součástí této diplomové práce je i instalační CD, jehož obsah je rozdělen do následujících složek:

- **Installation** – Obsahuje instalační soubor aplikace ECA a aplikací potřebných pro export písemné práce do dokumentu PDF.
- **Source** – V tomto adresáři se nachází soubory se zdrojovými kódy aplikace ECA
- **Sample Written Assignments** – V této složce jsou k nahlédnutí opravené písemné práce.
- **Thesis** – Zde se nachází elektronická podoba tohoto dokumentu.