VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# REVIZNÍ SYSTÉM PRO LATEX

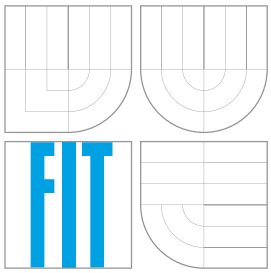
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ANTONÍN KYSELÁK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# REVIZNÍ SYSTÉM PRO LATEX

LATEX REVISION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ANTONÍN KYSELÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZBYNĚK KŘIVKA, Ph.D.

BRNO 2012

# Abstrakt

Práce se zabývá návrhem integrovaného nástroje textového editoru pro dokumenty systému LaTeX, který umožňuje sledování, zaznamenávání a schvalování změn v dokumentech jako podporu pro týmovou spolupráci při psaní dokumentů. Aplikace je určena především pro operační systém Windows, pracuje s LaTeXovou distribucí MikTeX a textovým editorem Texmaker.



# Abstract

This study describes the design of an integrated tool in text editor designated for documents written in LaTeX format. It offers the ability of tracking, storing and approving changes in document as a support for team cooperation, when writing a document. Application is designed mainly for operating system Windows. It works with LaTeX distribution MikTeX and the text editor Texmaker.



# Klíčová slova

LaTeX, TeX, sledování změn, MikTeX, Texmaker, revize, Windows

# Keywords

LaTeX, TeX, tracking changes, MikTeX, Texmaker, revision, Windows


# Citace

Antonín Kyselák: Revizní systém pro LaTeX, diplomová práce, Brno, FIT VUT v Brně, 2012

# Revizní systém pro LaTeX

## Prohlášení

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

.....
Antonín Kyselák
25. května 2012

## Poděkování

Tímto děkuji vedoucímu mé práce panu Ing. Zbyňku Křivkovi, Ph.D. za vedení a především ochotu při dokončování práce

# Obsah

# Seznam obrázků

# Kapitola 1

# Úvod

V dnešní době je týmová spolupráce, jak při vývoji softwaru, tak při jiných náročnějších pracovních úkolech, nezbytná. Proto je již léta zaváděna mezi pracovní nástroje vývoje celá řada systémů pro týmovou spolupráci, jako jsou různé typy správy verzí, plánovacího softwaru a nástrojů pro organizaci těchto týmů.

Tato podpora se týká také vytváření dokumentů v týmech, což je případ této diplomové práce. Většina nástrojů pro vytváření dokumentů, textových procesorů, je vybavena alespoň jednoduchou podporou pro sledování změn v dokumentu, což umožňuje přehled nad změnami dokumentů jednotlivých osob v týmu a jejich následné prohlížení, popř. schvalování, stornování atd.

Tato diplomová práce se zabývá vytvořením takového systému sledování změn pro sázecí systém LaTeXa jeho integraci do vybraného editoru dokumentů v jazyce TeX.

## 1.1 Cíl práce

Cílem této práce je tedy vytvořit takový nástroj, který dokáže pracovat s dokumenty v DTP systému LaTeX, bude pomocí něj možné určitým způsobem ukládat změny provedené jednotlivými uživateli pracujícími s daným dokumentem, podávat o nich souhrnné informace a umožnit tyto změny publikovat (nejlépe správci dokumentu, např. vedoucímu týmu).

Tento nástroj by měl být integrován do vybraného textového editoru pro dokumenty v LaTeXu tak, aby bylo jeho použití jednoduché, maximálně automatizované a schopné podat informace o změnách ve vhodné vizuální formě.

Předpokládá se offline použití, tzn. na rozdíl od nástrojů pro správu verzí apod., které jsou vyvinuty především pro vzdálené používání, verzování zdrojů a sledování jejich změn, tento nástroj bude pracovat s jedinou kopií dokumentu.

## 1.2 Popis jednotlivých kapitol

V kapitole 2 nalezneme stručný přehled nástrojů, zabývajících se správou revizí a jejich srovnání. Detailně je popsán způsob sledování změn v Microsoft Word. V kapitole 3 je stručně a vysvětlen systém LaTeX, principy práce s ním a dále popsána distribuce MikTeX. Kapitola 4 pojednává o existujících nástrojích pro tvorbu dokumentů v LaTeXu, nalezneme zde srovnání a doporučené použití. Nakonec je zde detailněji představen editor Texmaker. V kapitole 5 jsou pak zhodnoceny informace z předchozích kapitol a sestaven seznam důležitých bodů pro návrh nástroje, navrhovaného v této práci. Kapitola 6 detailně popisuje navržené řešení a nakonec jsou v kapitole 7 zmíněný vybrané implementační postupy.

# Kapitola 2

# Existující nástroje

Samostatných i integrovaných nástrojů pro týmovou spolupráci a správu verzí existuje celá řada. Každý systém má své výhody a nevýhody a je podle toho určen pro různá použití. Zde uvedeme několik základních typů těchto nástrojů a jejich zástupce (viz [14], [10]):

- **Systémy s centrálním úložištěm** - Většinou jde o systémy pro správu většího množství textových souborů libovolného typu, jsou vhodné např. pro rozsáhlé zdrojové kódy. Používají komunikaci typu klient - server, kde informace uchovává server.
  Zástupci: CVS (Concurrent versions system), SVN (Subversion)

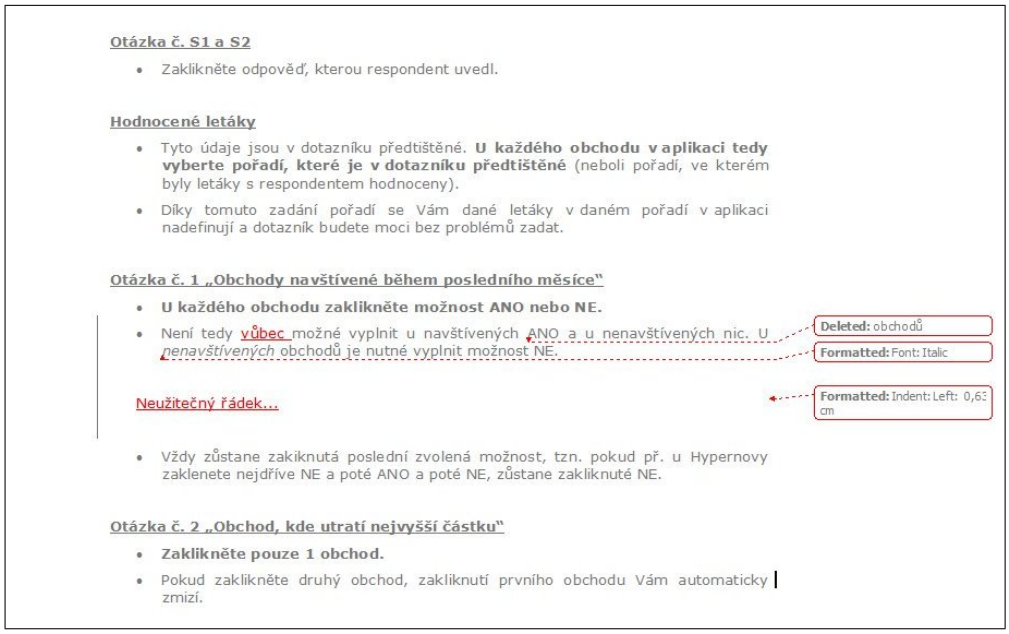
- **Distribuované systémy** - Tyto systémy jsou vhodné pro stejnou oblast jako výše uvedené centrální systémy, ale používají rozložení zátěže a paměťových nároku mezi jednotlivé participanty projektu. Používají tedy komunikaci typu peer-to-peer.
  Zástupci: Git, Bazaar

- **Lokální systémy pro správu** - Tyto systémy jsou jednodušší, protože zpravidla pracují s jednotlivými soubory, ne z celými adresářovými strukturami a navíc pracují spíše lokálně. Jsou tedy vhodnější např. pro jednoduché skripty, konfigurační soubory, textové dokumenty apod. Níže uvedený zástupce dokáže pracovat i s binárními soubory, ale se sníženou efektivitou.
  Zástupci: RCS (Revision control system)

- **Integrované systémy pro správu dokumentů** - V tomto případě se nejedná o samostatné programy, ale o integrované součásti v jiných programech, v tomto případě v textových procesorech. Všechny přední textové procesory tuto funkci sledování a revidování změn obsahují. Tento typ je pro tuto práci nejvhodnější, protože je vhodný pro práci s dokumentem přímo v textovém editoru.
  Zástupci: Microsoft Word, OpenOffice Writer, KWord, Pages

## 2.1 Kontrola změn v Microsoft Word

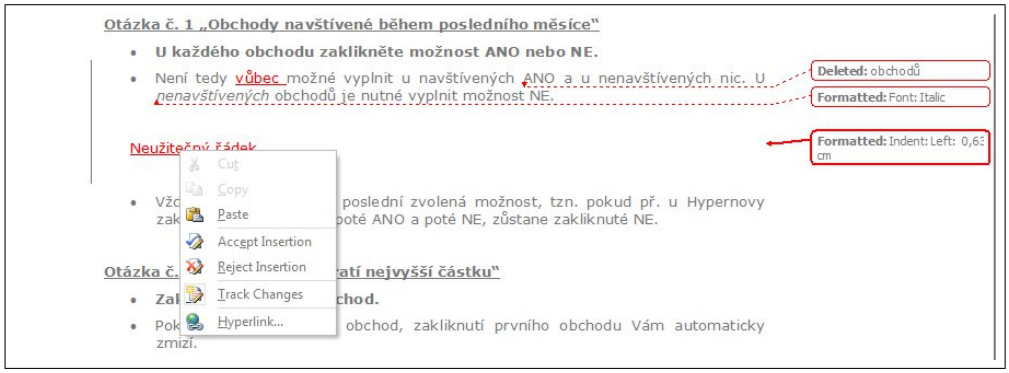
V textovém editoru Microsoft Word je již řadu let obsažen nástroj pro sledování změn v dokumentu. Po vytvoření úvodního dokumentu lze zapnout toto sledování a poté veškeré další změny jsou zobrazovány uživateli přímo při úpravě dokumentu grafickým odlišením (viz [5]). Řádky ovlivněné změnou jsou navíc na okraji označeny. Na obrázku 2.1 je tato funkce ilustrována.




Obrázek 2.1: Grafické zobrazení změn v dokumentu

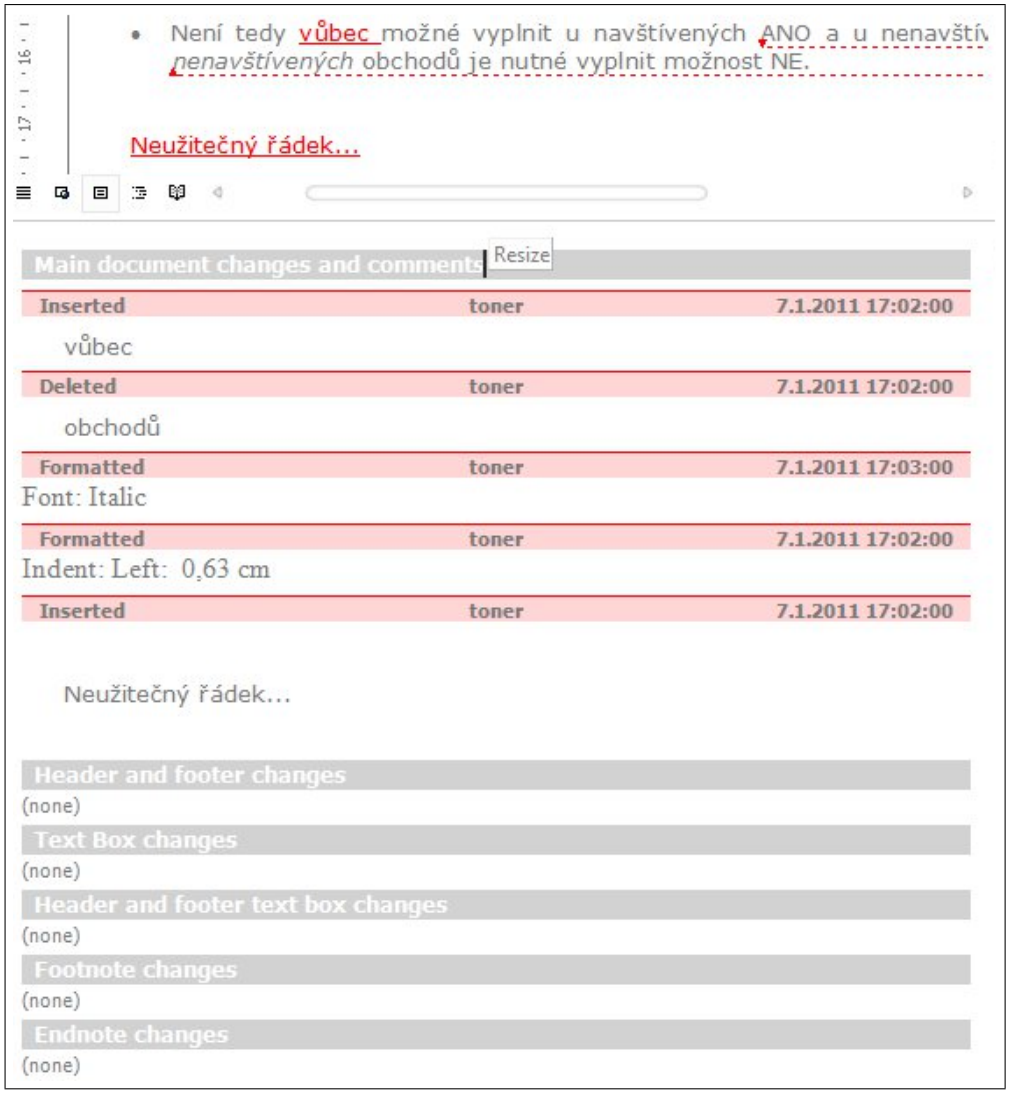
Od zapnutí sledování jsou veškeré změny v dokumentu zaznamenávány perzistentně, tzn. jsou ukládány do souboru dokumentu a nadále stále přístupné. Ke každé změně je uložena časová známka, dostupné informace o uživateli (v případě Microsoft Word informace nastavené přímo v textovém editoru, nikoli v systému) a je také možné k ní přidat komentář.

Pokud se některý z uživatelů pracující s dokumentem rozhodne pro revizi, lze změny jednotlivě či hromadně přijmout, resp. zamítnout. Pohled na seznam změn v Microsoft Word je zobrazen na obrázku 2.3. Možnost jednotlivé změny pomocí kontextového menu je vidět na obrázku 2.2.




Obrázek 2.2: Možnost jednotlivě schvalovat, resp. zamítat změny pomocí kontextového menu

Zobrazení změn v dokumentu je možné nastavit v možnostech textového procesoru. Lze přiřadit různé barvy, či způsoby zobrazení. Změny prováděné různými uživateli se automaticky zobrazují navzájem graficky odlišené (především barevně).




Obrázek 2.3: Zobrazení seznamu změn

# Kapitola 3

# Systém LaTeX

Systém LaTeX je balík maker pro DTP systém TeX. Není tedy sám o sobě sázecím programem, ale spíše nadstavbou sázecího programu. Usnadňuje používání TeXu prostřednictvím maker, které zprostředkovávají používané postupy tvorby dokumentu a jeho prvky tak, aby co nejvíce urychlil psaní zapouzdřením jinak složitých a zdlouhavých postupů nízkoúrovňovými způsoby, které TeX poskytuje.

LaTeX byl vytvořen v roce 1985 Lesliem Lamportem právě z důvody usnadnění psaní dokumentů kvůli velké programátorské náročnosti při používání samotného TeXu. V současné době je vyvíjena verze LaTeX3, která se snaží sjednotit různé verze LaTeXu, aktuální pracovní verze je nazývána LaTeX2$\varepsilon$.

Tvorba dokumentů v LaTeXu neprobíhá interaktivně s okamžitým výsledkem, jak je to běžné u textových procesorů. Zdroj dokumentu je zde uložen ve formě textového souboru ve formátu *.tex*, což je zdrojový kód značkovacího jazyku pro TeX. Pro vysázení dokumentu do výsledné podoby je potřeba tyto zdrojové kódy zkompilovat. Nepřítomnost okamžité vizuální informace je zde vynahrazena komplexnějším výpočtem celého dokumentu při kompilaci, což umožňuje precizní vysázení textu.

## 3.1 TeX

TeX je kvalitní sázecí program pro tvorbu dokumentů, ceněn je zejména pro jeho prvenství v sázení matematických vzorců, proto je oblíben v akademických kruzích, především v oblasti matematiky, fyziky a informatiky.

Obsahuje množství nízkoúrovňových funkcí (cca. 300) pro sázení dokumentu, patří mezi ně funkce pro ovlivňování načítání zdrojového textu, zadávání symbolických názvů pro objekty, nastavování parametrů objektů, specifikaci formátování výstupu (boxy, tabulky, atd.), zalamování a také pro vytváření maker, což jsou funkce, nad kterými vznikl LaTeX.

TeX je také velice populární a používaný díky množství rozšiřujících balíků, zlepšujícím jeho funkčnost a použití.

## 3.2 Jazyk a struktura formátu *.tex*

Jazyk dokumentů pro TeX je v základu velice jednoduchý značkovací jazyk. Značka je definována ve formátu `\příkaz[seznam parametrů, ...]{obsah příkazu}`. Tyto značky lze v textu téměř libovolně kombinovat a vnořovat. Blok { ... } lze použít i samostatně a potom v každém takto uvozeném bloku je uchováno vlastní nastavení, které se dále

neprojeví. Text dokumentu se v souboru může vyskytovat volně i uvnitř bloků, oddělení odstavců se provádí rozdělením dvou částí textu dvěma konci řádků.

Speciálním blokem je blok pro matematické vzorce uvozovaný mezi `$ ... $` (zůstane uvnitř textu) nebo mezi `$$ ... $$` (pro tento blok je vytvořen zvláštní odstavec). Pro tyto bloky platí jiná sémantická pravidla přizpůsobená jednodušší práci se vzorci. Další speciální bloky, pro které platí odlišná pravidla se nazývají *prostředí* a jsou uvozena příkazy `\begin{prostředí}` a `\end{prostředí}`.

Jazyk obsahuje také komentáře, zde jsou použity řádkové, uvozené znakem `%`.

### 3.2.1 Příklad souboru ve formátu *.tex*

Zde je jako ukázka uvedena část kódu souboru, obsahující několik možností jazyka:

```
Klasický text používající  {\it kurzívu},  {\bf tučné písmo}
a {\tt strojové písmo}. Dále matematické vzorce v~odstavcích
$ \frac{a^2}{b_i} - \epsilon $  i  v~samostatných  blocích:
$$  \frac{a^2}{b_i} - \epsilon  $$ % a také řádkový komentář
```

tento kód je vysázen v podobě:

Klasický text používající *kurzívu*, **tučné písmo** a `strojové písmo`. Dále matematické vzorce v odstavcích $\frac{a^2}{b_i} - \epsilon$ i v samostatných blocích:

$$\frac{a^2}{b_i} - \epsilon$$

## 3.3 Prostředí MiKTEX

Pro tuto práci je zvoleno prostředí MiKTEX([2]), které je nejpopulárnějším prostředím pro operační systém Microsoft Windows. Obsahuje:

- Vše potřebné pro kompilování dokumentů (TEX/ LATEX kompilátory)
- Přímé sázení do formátu .pdf, převodníky různých dalších formátů
- základní balíky LATEXu, sady různých stylů, fontů, ...
- Několik nástrojů a programů (editor *TeXWorks*, rychlý prohlížeč výstupů *Yap*, ...)
- efektivní systém správy balíků, který poskytuje instalaci chybějících balíků za běhu, tzn. při absenci balíku, který je vyžadován v dokumentu nabídne jeho stažení a instalaci. To dovoluje mít k dispozici vždy pouze ty balíky, které jsou potřebné.

Instalace této distribuce je uživatelsky přívětivá, probíhá přes jednoduchý grafický instalátor typický pro Microsoft Windows. Podrobnější popis instalace této distribuce a dalších nástrojů pro práci pod Microsoft Windows lze nalézt v [11].

Distribuce MiKTEX je vydávána pod open source licencí LPPL (The LaTeX Project Public License), což je speciální licence pro LATEX. Tato distribuce je pravidelně a často aktualizovaná a nově je dostupná pro Windows 7 i jako beta verze pro GNU/Linux.

# Kapitola 4

# Textové editory pro LaTeX

Existuje relativně velké množství editorů pro práci s dokumenty pro LaTeX. Lze je rozdělit do několika kategorií, a to podle typu editace (WYSIWYG, zdrojový kód), licence, popř. dostupnosti zdrojových kódů, typu operačního systému, pro který je určen a také podle rozsahu funkcí. Detailní srovnání dostupných editorů lze najít např. v [12].

## 4.1 Nejdůležitější používané funkce a nástroje

Mezi nejdůležitější funkce editorů pro soubory v LaTeXu patří především automatický proces kompilace a přehlednější zpětná vazba výstupu kompilátoru, zvýrazňování zdrojových textů a rychlejší orientace v nich (vyhledávání, regulární výrazy, zobrazení struktury objektů v textu, atd.), použití šablon, našeptávání příkazů při psaní, vrácení nechtěných změn, náhled výstupu a u WYSIWYG editorů dokonce přibližná podoba formátování přímo při psaní. Níže uvedené editory obsahují buď všechny, nebo alespoň většinu těchto funkcí.

## 4.2 Nejrozšířenější editory pro LaTeX

V této sekci nalezneme příklady a stručné popisy vybraných editorů. Nachází se zde zástupci různých kategorií editorů, jedná se o vždy o ty více rozšířené.

- **LyX** - jeden z nejrozšířenějších editorů podporující WYSIWYG editaci. Je dostupný pro operační systémy Windows, Linux i MacOS. Obsahuje většinu potřebných nástrojů pro komplexní práci s dokumenty. Grafické zobrazení formátování při editaci je pouze orientační, LyX nezvládne přesně napodobit složitý sázecí proces LaTeXu.
- **TeXnicCenter** - propracovaný a velmi rozšířený editor pro Windows. Obsahuje velké množství nástrojů pro jednodušší práci s dokumenty, starší verze nepodporovaly některé zásadní možnosti (např. Unicode), ale nejnovější alfa verze (2.0) již tyto nedostatky řeší. Největší předností tohoto editoru je jeho vysoká konfigurovatelnost.
- **AUCTEX** - existuje jako rozšíření pro textový editor Emacs, velmi populární editor především pro programátory, původně a dnes nejčastěji používaným v operačním systému Linux. Je velmi dobře konfigurovatelný, přizpůsobitelný a ovladatelný, ale vyžaduje vyšší znalosti a zkušenosti uživatele. Tak jako výše uvedené obsahuje velké množství funkcí pro práci s dokumenty. Stejně jako pro Emacs existuje i rozšíření pro podobně náročný a oblíbený editor Vim.

## 4.3 Editor Texmaker

Pro tuto práci je zvolen editor Texmaker, protože jde o jednoduchý, ale dostačující editor s množstvím funkcí, je nezávislý na distribuci, díky čemuž není problém integrace s prostředím MikTEX (viz 3.3), zvoleným pro tento projekt. Dalšími důležitými výhodami jsou jeho licence (GPL), díky které jsou také dostupné jeho zdrojové kódy a komunita jeho vývoje dává prostor pro další rozšiřování, a dále jeho přenositelnost (lze používat v operačních systémech Windows, Linux i MacOS). V současné době je dostupná nejnovější verze 2.2.1.

### 4.3.1 Funkce a přednosti editoru

Mezi hlavní nástroje a funkce editoru patří:

- **Uživatelské prostředí** - Dobře strukturované uživatelské prostředí, které se skládá z oken (textový editor, navigátor/tabulky příkazů, výstup kompilátoru) a lišt (menu, hlavní nástroje, formátování, uživatelské, atd.). Prostředí je velmi minimalistické, což klade nízké nároky na velikost zobrazovací plochy a poskytuje přehled nad velkou částí dokumentu.
- **Textový editor** - Poskytuje efektivní práci s textem, obsahuje funkce jako zvýrazňování syntaxe, našeptávání příkazů LaTeXu, kontrola souvisejících závorek, vyhledávání a nahrazování, kontrola pravopisu a další drobné nástroje. Editor podporuje kódování Unicode a lze v něm pracovat i se všemi dalšími známými kódováními. Tato část aplikace Texmaker bude stěžejním bodem cíle této práce.
- **Struktura dokumentu, tabulka příkazů** - Dovoluje mít přehled o struktuře dokumentu a jeho částí. Obsahuje pohled na strukturu obsahu dokumentu, seznam všech značek a bloků v dokumentu a seznam souborů vložených v projektu. Je to výborný prostředek pro orientaci a pohyb v celém projektu i jednotlivých dokumentech. Dále toto okno slouží pro zobrazení tabulek, či seznamů skupin příkazů a symbolů LaTeXu.
- **Konfigurovatelnost** - Editor obsahuje nastavení řady položek, jako jsou například příkazy pro různé typy kompilace, uživatelské klávesové zkratky, uživatelské značky a příkazy LaTeXu. Dále dovoluje konfigurovat barvy a písma editoru, parametry rychlého překladu, či kódování textu.
- **Prohlížeč *.pdf* souborů** - Texmaker obsahuje vestavěný a velice nenáročný prohlížeč *.pdf* souborů, který dovoluje prohlížet výstup okamžitě po překladu. V aplikaci lze ale nastavit i externí prohlížeče, jak *.pdf* soubory, tak také další typy, jako dokumenty ve formátu *.dvi* nebo *.ps*.

### 4.3.2 Vývojové prostředí editoru

Editor je vyvíjen nezávisle na operačním systému (lze zkompilovat pro operační systémy Windows, Linux i MacOS). Zdrojové kódy jsou psány v jazyce C++, jsou volně k dispozici a je možné se po domluvě s autorem podílet na vývoji. Grafické uživatelské prostředí (GUI) je v aplikaci vyvíjeno s použitím knihovny Qt, což je plně multiplatformní, obsáhlá a velmi dobře dokumentovaná knihovna. Výhodou je také přítomnost pomocných aplikací pro snadnější návrh uživatelského prostředí (Qt Creator, Qt Designer).

### 4.3.3 Prerekvizity a závislosti

K překladu, resp. běhu aplikace je nutné mít k dispozici několik knihoven a nástrojů:

- **Qt toolkit** - Soubor knihoven pro tvorbu uživatelského rozhraní aplikace. Kromě toho poskytuje další nástroje jako lokalizace, práce s SQL, XML, vlákny, soubory a multimédii.
- **Knihovna Poppler** - Knihovna pro zobrazování dokumentů ve formátu *.pdf*.
- **Distribuce LaTeXu** - Nutnost pro překlad dokumentů. Pro operační systém Windows je nejrozšířenější distribucí zmiňovaný MikTeX (viz 3.3), pro Linux pak TeXlive a pro MacOS MacTeX.

Editor také používá několik knihoven, nutných pro překlad, ty jsou ale poskytovány spolu se zdrojovými kódy, není tedy nutné je zvlášť instalovat. Jsou jimi:

- **Hunspell** - Knihovna, poskytující třídu pro kontrolu pravopisu v dokumentu. Využívá volně dostupných slovníků pro kontrolu pravopisu ve formátu *.dic* a *.aff*.
- **Synctex parser** - Syntaktický analyzátor pro SyncTeX, umožňující snadnější synchronizaci mezi vytvářeným dokumentem a výstupem ve formátu *.pdf*.

### 4.3.4 Struktura zdrojového kódu

Zdrojové kódy aplikace jsou velice jednoduše čitelné, struktura tříd není příliš složitá. Čitelnost zdrojových kódů je také zjednodušena tím, že velká část tříd (především týkajících se grafického uživatelského rozhraní) je odvozena od existujících tříd knihovny Qt. Tzn., že velká část funkcionality je skrytá, ale dohledatelná v dokumentaci knihovny Qt.

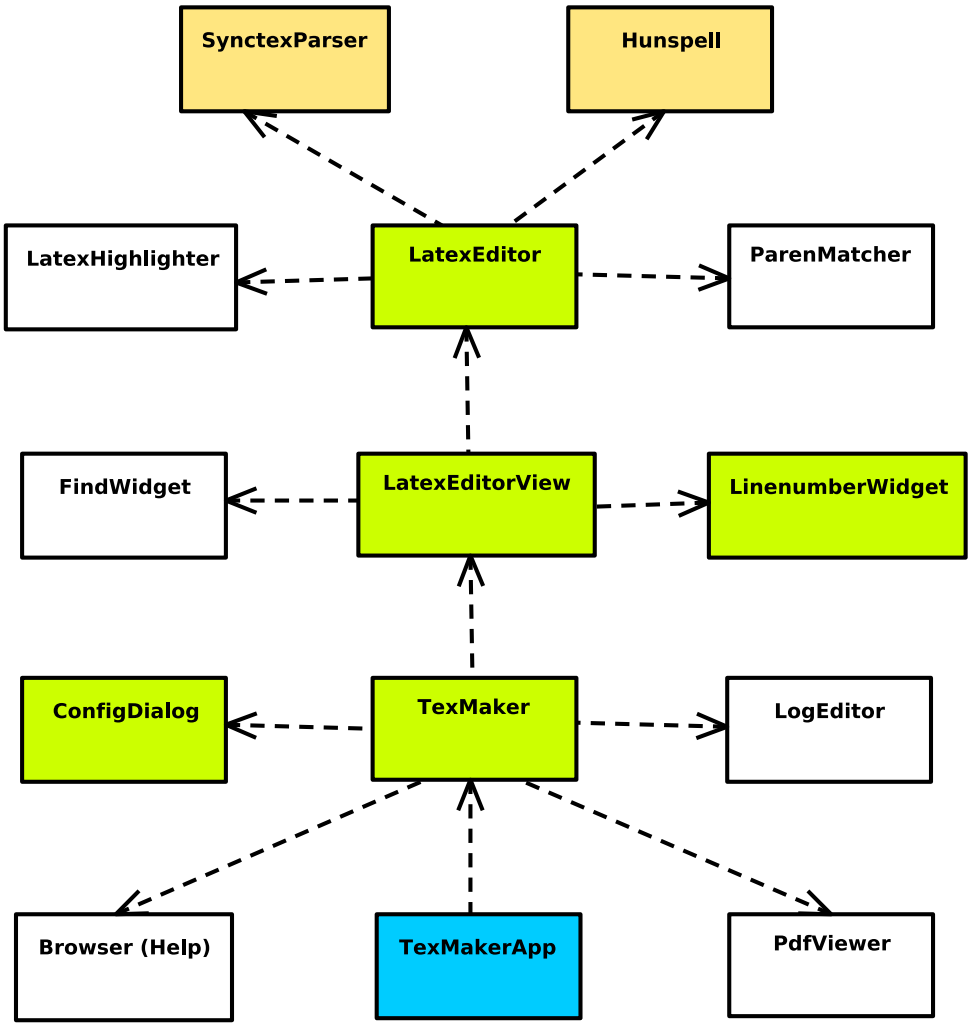
Hlavní třídou aplikace je třída `TexmakerApp`, která provádí inicializaci aplikace, načtení důležitých souborů (konfigurace, lokalizace, atd.) a vytvoření hlavního okna. To je představováno třídou `Texmaker`, nejobsáhlejší třídou aplikace, zajišťující skladbu prvků hlavního okna, práci s nastavením aplikace a většinu jejích zpětných volání. Provádí také veškerá načítání a ukládání souborů, což je jedno z důležitých míst, které je nutné uvažovat v návrhu.

Tato třída pak využívá několik důležitých tříd pro zobrazování různých informací o dokumentu, jako nastavení možností (`ConfigDialog`), zobrazení nápovědy (`Browser`), prohlížení *.pdf* výstupu (`PdfViewer`), zobrazení přehledného výstupu překladu (`LogEditor`). Nejdůležitější součástí je okno textového editoru, představováno třídou `LatexEditorView`, obsahující widgety pro zobrazení čísel řádků s informacemi, vyhledávání v textu a samozřejmě textový editor, popsán třídou `LatexEditor`.

Třída `LatexEditor` je nejdůležitějším místem pro navrhovaný nástroj, protože se stará o zobrazování textu a manipulaci s ním a tím provádí i změny textu, důležité pro tuto práci. Sám využívá již několik tříd pro různé kontroly a zvýrazňování textu, jmenovitě `LatexHighlighter` pro barevné zvýrazňování textu, `Hunspell` pro kontrolu pravopisu a `ParentMatcher` pro zvýrazňování souvisejících závorek.

Dále aplikace obsahuje další třídy pro dodatečnou funkcionalitu a řadu dialogů, které se ale nijak nedotýkají navrhovaného nástroje.

Stručný graf závislostí zobrazující propojení nejdůležitějších tříd aplikace lze nalézt na obrázku 4.1.




Obrázek 4.1: Jednoduchý graf závislosti důležitých tříd v aplikaci Texmaker, žlutě jsou označeny knihovny třetích stran, zeleně třídy ovlivněné rozšířením a modře hlavní třída aplikace

# Kapitola 5

# Zhodnocení požadavků a stručný návrh

V této kapitole jsou shrnuty dosavadní požadavky a informace sebrané z předchozích kapitol a vyvozeny z nich možnosti a omezení vztahující se na tuto práci. V druhé části kapitoly je pak s pomocí těchto závěrů vytvořen stručný návrh aplikace, připravený jako podklad pro další práci.

## 5.1 Shrnutí předchozích kapitol

Zde jsou shrnuty dosavadní poznatky do třech podkapitol, zabývajících se postupně možností revizí změn, ukládání změn v dokumentu, uživatelským prostředím editoru Texmaker a možností jeho využití pro navrhovaný nástroj.

### 5.1.1 Možnosti revizí dokumentů

V kapitole 2 lze vidět srovnání různých typů systému pro správu změn nebo verzí. Jednoznačně vyplývá, že nejvhodnější pro tuto práci je možnost poslední, a to jednoduchý systém pro zaznamenávání pouze neschválených změn (bez historie) v dokumentu a jejich uchování do schválení (resp. zamítnutí).

Ukládání změn by mělo být jednoduše ve formátu řádkového záznamu v souboru dokumentu, či pro tyto potřeby vytvořeném metasouboru. Schválené a zamítnuté změny by měly být z tohoto seznamu vyřazeny.

### 5.1.2 Ukládání záznamů o změnách

Pro ukládání záznamu o změnách se nabízejí dvě základní možnosti, obě mají své výhody a nevýhody, které v této podkapitole shrneme.

První možností je ukládání záznamů přímo do souborů dokumentu, kde lze pro tato data využít komentáře. To s sebou nese výhodu uložení všech dat v jednom souboru, což je výhoda např. při přenosu samostatného souboru dokumentu. Nevýhoda tkví v tom, že je nutné při každém načtení souboru filtrovat záznamy v komentářích a odstranit je ze zobrazeného kódu, aby uživatel nijak nepocítil přítomnost těchto metadat v souboru. Při ukládání dokumentu musí aplikace naopak ke každé změně takovýto komentář vložit na vhodné místo do souboru.

Druhou možností je vytváření ke každému sledovanému souboru další soubor s daty o změnách. To má výhodu v jednodušší správě těchto změn, ale naopak nevýhodu v rozdělení dat do více souborů, což může přinést komplikace v podobě ztráty integrity při přenosu souboru.

### 5.1.3 Prostředí editoru Texmaker

V sekci 4.3 můžeme vidět, že editor nabízí hned několik míst, kde je možné zobrazit informace o zaznamenaných změnách v dokumentu. V první řadě je třeba je vizuálně odlišit v textu editoru tam, kde je možnost zvýrazňování syntaxe. Dále je možné zobrazovat změny strukturovaně podle příslušnosti kapitol, což umožní uživateli rychlejší přehled a orientaci. Další možnost, kterou editor nabízí je okno výstupu, které lze v tomto projektu využít jako vhodný prostor pro zobrazení seznamu změn s podrobnějšími informacemi o nich. Vhodné bude použití záložek s možností přepnutí mezi okny s výstupem a seznamem změn. Zde by bylo také vhodné poskytnout uživateli hledání a filtrování těchto změn, podle jejich parametrů. Podrobněji bude toto řešení představeno v sekci 5.2.3.

## 5.2 Stručný návrh nástroje

Tato podkapitola je vůči budoucí finální podobě diplomové práce pouze rozpracováním, podrobnější pohled na návrh bude popsán až v druhé části práce. Nachází se zde podrobnější nástin vlastností programu, jeho integrace do editoru, typu ukládaných dat, kompozice uživatelského rozhraní a integrací modulů a tříd do zdrojových kódů editoru.

### 5.2.1 Uložení provedených změn

Jako typ úložiště pro změny (diskutované v sekci 5.1.2) je po zvážení vhodnější první typ, a to uložení záznamů přímo do souboru s dokumentem. Výhoda lepší udržitelnosti dat pohromadě převyšuje nad nevýhodou složitější implementace.

Prakticky lze pak ukládání řešit způsobem, kdy text dokumentu bude v souboru zachycen ve změněné podobě (kvůli možnosti vidět změněný text i v jiných editorech) a tyto změny pak budou v souboru uloženy za textem dokumentu v komentářovém bloku v daném formátu (viz níže v kapitole 6.5.4). Uložení změn až za textem dokumentu s sebou nese výhodu zachování pozic změn (čísla řádků a pozic v řádcích).

Nezávisle na typu úložiště lze stanovit formát ukládaných dat. Ten by mohl vypadat jednoduše jako řádkový záznam ve vybraném souboru obsahující čas a datum změny, uživatele provádějícího změnu, řádek a pozici změny, typ změny (vložení textu, smazání textu) a také změněná data. Formát by mohl vypadat např. takto:

```
UŽIVATEL, ČAS, DATUM, TYP ZMĚNY, ŘÁDEK, POZICE, KOMENTÁŘ, "DATA"
```

Většinu dat lze získat z prostředí přímo při provádění dané změny. Výjimku tvoří pole *uživatel*, které lze získat více způsoby. Textový editor žádné možnosti profilů neposkytuje, tím se jako vhodný způsob nabízí získání jména uživatele z aktivního účtu operačního systému. Další výjimkou je pole *komentář*, které je nepovinné. Komentář bude moci uživatel volitelně přidat pomocí dialogu, vyvolatelného z kontextového menu dané změny.

Data budou při každém načtení odfiltrována ze souboru, restrukturována do reprezentujících objektů a při uložení opět z těchto strukturovaných dat vložena na konec souboru.

### 5.2.2 Řešení pádu aplikace

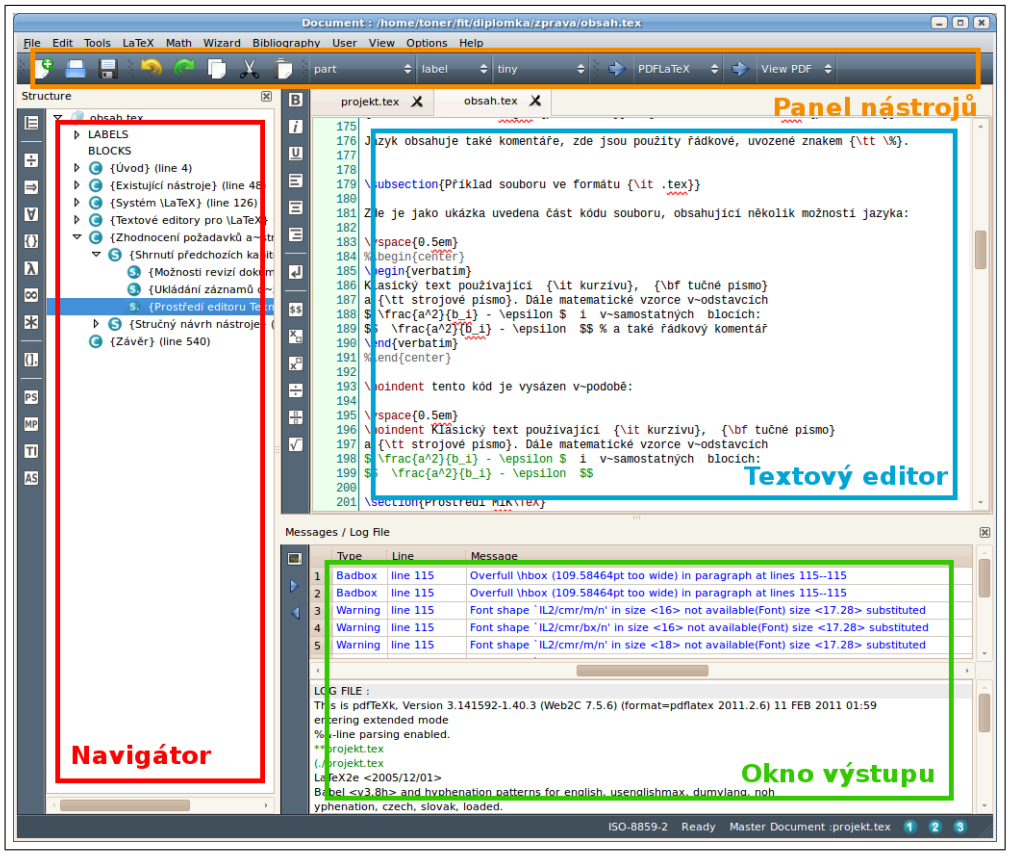
Editor Texmaker nenabízí nijak sofistikovaný nástroj pro zotavení po pádu aplikace. Udržuje si pouze seznam otevřených souborů, které lze pak po pádu aplikace opět najednou otevřít. Neřeší ale případnou ztrátu dat. To lze řešit tak, že ke všem otevřeným souborům se při jejich načtení vytvoří pracovní kopie, která bude průběžně automaticky ukládána. Při správném zavření daného souboru pak kromě odstranění ze seznamu otevřených souborů aplikace odstraní její pracovní kopii. Naopak při pádu aplikace a jejím opětovném spuštění lze detekovat pád přítomností pracovních kopií otevřených souborů. Název pracovní kopie může být např. ve formátu `%jméno_souboru%.tex~`.

### 5.2.3 Kompozice uživatelského rozhraní

V sekci 5.1.3 je nastíněna možná integrace do uživatelského prostředí, všechna zmíněná místa by měla být v návrhu využita, jmenovitě:

- **Textový editor** - V textovém editoru budou graficky odlišeny veškeré změny pro dobrý přehled uživatele. Změněný text bude odlišen od ostatního jiným formátem, barvou textu, barvou pozadí, podtržením, či jiným zvýrazněním, např. kurzívou. Způsob odlišení bude možné nastavit v *nastavení editoru*, viz níže. Kontextové menu vyvolané nad změněným textem bude obsahovat volby pro manipulaci s danou změnou, konkrétně přijetí a zamítnutí změny.
- **Navigátor** - V okně navigátoru, které standardně dovoluje orientaci a pohyb v souborech a dokumentech přibude záložka pro strukturované zobrazení změn s příslušností k daným kapitolám. Zobrazení struktury bude vycházet z existujícího, ale navíc zde bude pod každým názvem sekce (kapitoly, podkapitoly atd.) zobrazen seznam změn vyskytujících se v této sekci. Dále bude v tomto okně zobrazen widget pro vyhledávání v těchto změnách. Bude možné vyhledávat podle autora, nebo podle času a data změny.
- **Okno výstupu** - Je ideálním prostorem na přidání záložky pro zobrazení podrobného seznamu změn taktéž s možností hledání a navíc i filtrace. Seznam bude zobrazen pomocí tabulky, která dovolí i řazení změn podle vybraných klíčů prostřednictvím tlačítek v hlavičce této tabulky. V této tabulce bude možnost vícenásobného označení vybraných změn. U každého záznamu bude zobrazeno číslo řádku, po kliknutí na něj přesune aplikace kurzor na související změnu. Okno bude obsahovat nástrojovou lištu s tlačítky pro hromadné označení a odznačení všech záznamů tabulky, vyvolání dialogu pro vyhledávání a filtrování změn a také tlačítka pro přijetí, resp. zamítnutí označených změn.
- **Panel nástrojů** - Na panel nástrojů přibude možnost vložení skupiny ovládacích prvků pro manipulaci se změnami, konkrétně hromadné přijetí a zamítnutí změn, skok na další a předchozí změnu a tlačítko zapnutí, resp. vypnutí funkce záznamu změn.
- **Nastavení editoru** - V dialogu pro nastavení aplikace bude přidána záložka s možnostmi konfigurace nástroje pro sledování změn. Bude zde možné nastavit grafické zobrazení změn a chování nástroje.

Na obrázku 5.1 lze vidět současnou kompozici uživatelského rozhraní editoru s vyznačením klíčových oblastí pro rozšíření.




Obrázek 5.1: Ovlivněné prvky současného uživatelského rozhraní

### 5.2.4 Integrace do zdrojových kódů

Při budoucím detailním návrhu modulů a tříd nástroje pro sledování změn bude vhodné co nejvíce oddělit tyto objekty od existujících, tzn. co nejméně zasáhnout do struktury a kódu již vytvořeného. Modul nástroje by měl být centrálně řízen a poskytovat spíše zapouzdřené filtry při zpracování textu dokumentu. Jako příklad můžeme zvolit editaci dokumentu. Místo stávajícího přístupu:

$$\text{Třída iniciující změnu} \rightarrow \text{Třída realizující změnu}$$

by měl být proud změnen na:

$$\text{Třída iniciující změnu} \rightarrow \text{Třída zachycující změnu} \rightarrow \text{Třída realizující změnu}$$

Jako návrhový vzor se jeví vhodný pozorovatel (observer), který bude upozorňován na změny v textu. Tato pozorující třída bude uchovávat kopii textu dokumentu, neboť použité knihovny nenabízejí jednoduchou signalizaci před změnou, ale až po ní. Je tedy nutné znát původní text především v případě mazání částí textu. Po každé změně bude tato kopie aktualizována, aby zůstala zachována integrita.

Takto by měla pracovat většina tříd, tj. odděleně od ostatních. Všechny třídy tohoto nástroje by měly vzájemně komunikovat pomocí třídy zapouzdřujícím zasílání zpráv mezi těmito třídami (návrhový vzor prostředník).

Ideální datový typ pro uložení změn je strom (nebo obecně asociativní pole) obsahující uzly reprezentující jednotlivé změny. To je vhodné jak pro rychlé vyhledání, tak pro úpravu zaznamenaných změn. Každý uzel bude obsahovat všechny informace o změně (viz 5.2.1). Každá provedená změna bude zařazena podle následujících pravidel:

- Pokud vložení textu zleva nebo zprava zasáhne do již existující změny vložení textu, tak při totožném autorovi se změny sloučí do jedné, pokud se autor liší, text nezasažen novou změnou bude ponechán v původní změně a pro nový text bude vytvořena změna nová.
- Pokud vložení textu zasáhne doprostřed jiné změny, tak při totožném autorovi se změny sloučí do jedné, pokud se autor liší, text nezasažen novou změnou bude rozdělen na dvě změny, obklopující nový text a pro tento nový text bude vytvořena změna nová.
- Pokud změna nezasáhne do jiné, bude jednoduše vytvořena nová
- Pro mazání textu bude systém pracovat obdobně

Tyto položky budou ve stromě (asociativním poli) vždy vzestupně seřazeny proto, aby bylo možné efektivně vyhledávat. Každé další vložení či odebrání položky změny musí toto řazení respektovat.

Třídy pro filtraci načítání a ukládání souborů budou pracovat obdobně, jako třída zachycující změny dokumentu, ale navíc nebude pouze zachycovat informace, ale filtrovat data (tzn. spíše vzor prostředník). Při načtení bude tedy nejdříve předán text souboru této třídě a až ve změněné podobě bude předán editoru, při ukládání naopak.

Pro každou instanci textového editoru (tj. pro každý otevřený dokument) bude vytvořena jedna instance třídy sledující změny s vlastní kopií textu dokumentu a stromem změn.

V této kapitole jsou uvedeny pouze základní principy jádra navrhovaného nástroje. Podrobnější návrh jednotlivých tříd a jejich kompozice v existující aplikaci, návrh složitějších tříd (třídy pro uživatelské rozhraní, zobrazování změn, slučování dokumentů, apod.) a datových typů bude představen v budoucích kapitolách.

### 5.2.5 Možná rozšíření

V projektu jsou uvažována některá rozšíření, u kterých se zatím neuvažuje o jejich implementaci, ale je možné, že do budoucna se mohou v implementaci, nebo alespoň návrhu objevit. Patří mezi ně např.:

- Uložení historie změn (evidence, vhodné pro zpětnou informaci o dřívějších změnách)
- Integrace do TeXonWeb, webový editor LaTeXových souborů, vhodné pro vzdálenou práci více účastníků (viz [3])

# Kapitola 6

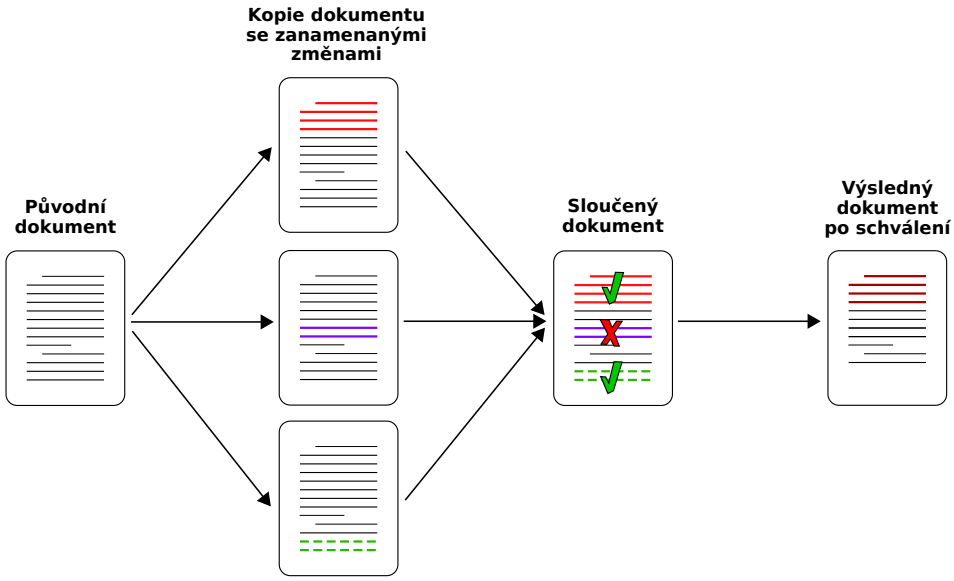
# Návrh systému

Tato kapitola se zabývá podrobnějším zkoumáním vývojového prostředí editoru Texmaker, knihovnou Qt, kterou editor vyžívá a sestavením návrhu systému pro sledování změn, který má být do toho prostředí integrován. Budou zde podrobně shrnuty všechny požadavky na tento systém, přehledně znázorněny a poté podle nich bude vytvořen datový a funkční návrh tohoto nástroje a jasně stanoveno rozhraní vůči zmíněnému prostředí.

## 6.1 Kompletní shrnutí požadavků

V kapitole 5 byly již zhruba probrány požadavky na vyvíjený nástroj a nastíněny postupy k jejich naplnění. V této kapitole bude sestaven kompletní výčet požadavků, který bude sloužit jako jasný a pevný základ pro zkoumání prostředí editoru a vytvoření návrhu odpovídajícího těmto požadavkům a co nejvhodnější integrace do editoru Texmaker.

### 6.1.1 Ilustrace fungování systému sledování změn




Obrázek 6.1: Princip využití nástroje pro sledování změn

### 6.1.2 Záznam změn při psaní dokumentu v reálném čase

V zadání práce je doporučená inspirace z již existujících nástrojů pro tvorbu dokumentů, konkrétně byl doporučen editor *Microsoft Word*, dále je inspirace čerpána z textového editoru *OpenOffice Writer*. Tyto editory poskytují již delší dobu nástroj pro sledování změn dokumentu a jejich revize a poskytují jednoduché rozhraní k jejich ovládání. Podrobněji jsou představeny v kapitole 2.1. Následující seznam pokrývá požadavky na záznam změn v reálném čase:

- Záznam změn musí jít vypnout a zapnout i uprostřed práce s dokumentem.
- Při otevření souboru se zaznamenanými změnami by mělo být sledování změn automaticky zapnuto.
- Pokud jsou v dokumentu přítomny změny, měly by být zobrazeny i v případě, že je zaznamenávání. vypnuto
- Při práci s dokumentem a zapnutým sledováním jsou veškeré změny uživatelem v textu. zaznamenávány.
- Při práci s dokumentem a vypnutým sledováním nejsou změny v textu zaznamenávány, ale můžou být ovlivněny již zaznamenané změny, to se týká především mazání textu (tímto je možné zaznamenané změny odstranit), ale i vkládání textu (vložením textu doprostřed změny je tato změna rozdělena).
- Změny vkládající text do dokumentu musí být graficky odlišeny podbarvením, změny od různých uživatelů od sebe musí být barevně rozlišeny.
- Změny odstraňující text z dokumentu musí být zobrazeny ponecháním smazaného textu v souboru, ale poté musí být také graficky odlišen, a to podbarvením stejnou barvou, jako v případě vkládání, text ale musí být intuitivně označen jako smazaný, tj. přeškrtnutím.
- Záznam změny musí obsahovat mimo typ, pozici a rozsah této změny také informace o autorovi, datu a času změny a musí být možné tuto změnu jejím autorem okomentovat.
- V praxi se počítá spíše s tím, že v jednom změněném dokumentu se budou vyskytovat pouze změny jednoho autora, je ale nutné zahrnout do řešení i opačný případ (ten nalezne uplatnění především při slučování dokumentů, viz. 6.1.3).
- Změny stejných typů a autorů musí být při jejich bezprostředním napojení či vložení do sebe sloučeny do jedné.
- Smazáním již smazaného textu (text je stále zobrazen, jen graficky odlišen) zůstává smazaná část neměnná, stále se jedná o původní změnu, to se týká kombinování změn jednoho i více autorů.
- Smazáním vloženého textu předchozí změna vkládající text zaniká (pokud je smazána jen část, změna je pouze ořezána), to se týká kombinování změn jednoho i více autorů.
- Zaznamenané změny lze přijmout nebo odmítnout. Přijetím změny je text podle ní upraven a záznam této změny je smazán, při odmítnutí je situace stejná, text je však vrácen do podoby původní.

### 6.1.3 Formát uložení, slučování a porovnávání

Zde jsou shrnuty požadavky na výstupní formát a ukládání souboru, dále na stav souborů při práci s dokumentem a také na způsob slučování a porovnávání paralelně zpracovaných kopií dokumentu. Podobně jako v předchozí podkapitole 6.1.2 je velká část inspirovaná ze zkoumaných textových editorů *Microsoft Word* a *OpenOffice writer*.

- Uložený dokument musí obsahovat aktuální text a informace o provedených změnách, tzn. ke každé změně musí být uloženy informace o typu, pozici, autorovi, datu a času změny. U změn vkládajících text bude uložena pouze délka vloženého úseku, u změn mazajících text bude uložen smazaný text.
- Změny musí být v souboru uloženy tak, aby neovlivňovaly obsah nebo formu dokumentu.
- Při práci s dokumentem musí být možné aktuální text zkompilovat LaTeXem, proto je nutné ukládat jej ve změněné podobě se záznamy změn ve speciálně formátovaných komentářích tak, aby výstupní dokument obsahoval vždy to, co vidí a očekává uživatel.
- Pokud dva dokumenty budou ctít původní obsah a veškeré změny budou zaznamenávány (tj. původní text zůstane zachován), bude možné je sloučit (tzn. shrnout všechny změny z více dokumentů do jednoho). Toto je vhodné především při revizi změn z více dokumentů od různých autorů.
- Pokud se vyskytne kopie dokumentu, která bude změněna běžným způsobem (tzn. původní text bude změněn) a změny nebudou zaznamenány, editor bude moci porovnat takto změněný dokument s původním s co nejoptimálnějším vyhledáním změn, které budou poté zaznamenány.
- Jako drobné vizualizační rozšíření bude integrována funkce generování dokumentu s označenými změnami. Editor bude schopen vygenerovat zvláštní dokument, ve kterém budou všechny zaznamenané změny z původního dokumentu vhodným způsobem označeny LaTeXovými značkami tak, aby bylo ve výstupním dokumentu vizuálně patrné které části textu byly změněny a kým byly změněny.

### 6.1.4 Zobrazení a ovládání v grafickém uživatelském rozhraní

Jako poslední je představen seznam požadavků, týkajících se grafického uživatelského rozhraní. Oproti existujícím řešením se toto bude lišit v omezení některých prvků, ale naopak rozšířením jiných. Požadavky jsou dány takové, aby byla především zachována intuitivnost, jednoduchost a funkčnost ovládání. Stručná představa je popsána v kapitole 5.2.3.

- Zaznamenané změny musí být jasně zobrazovány a aktualizovány v reálném čase při práci na dokumentu, musí být odlišeny změny vkládající text a změny mazající text.
- V panelu s čísly řádků budou řádky ovlivněné jakoukoliv změnou barevně zvýrazněny, to je vhodné především při změnách textu, obsahujícího netisknutelné znaky (většinou jde o znaky konce řádku), které nelze v textu editoru podbarvit.
- Přímo v editoru s použitím kurzoru myši bude možné provádět operace nad jednotlivými změnami, konkrétně přijetí, zamítnutí a editace komentáře změny; tyto operace budou přístupné v kontextovém menu, vyvolaném stiskem tlačítka myši nad příslušnou změnou.
- Po přesunu kurzoru myši nad změnu v textu bude zobrazena informace o autorovi a času této změny.

- Globální ovládací prvky pro práci se změnami, jako jsou zapnutí/vypnutí záznamu změn, zobrazení seznamu změn, přijetí a odmítnutí všech změn, sloučení a porovnání dokumentů, budou přístupné v hlavním menu aplikace, vybrané prvky pak budou umístěny i v nástrojové liště editoru.
- Seznam všech změn s příslušnými informacemi (autor, datum, komentář) bude umístěn ve spodním části okna aplikace, jak bylo znázorněno na obrázku 5.1. Jednotlivé změny bude moci uživatel označovat a hromadně nad nimi provádět operace přijetí i zamítnutí, tyto operace budou dostupné v nástrojové liště podokna se seznamem změn.
- Ve výše uvedeném seznamu změn budou zobrazeny všechny záznamy změn v přehledné tabulce, jejíž položky bude možné filtrovat podle zadaných kritérií. Tato kritéria bude moci uživatel zadat v panelu, který lze zobrazit tlačítkem na nástrojové liště podokna se seznamem změn.
- Každá položka tabulky reprezentující změnu bude mít přístupné kontextové menu s operacemi přijetí, odmítnutí a editace komentáře, podobně jako v textovém editoru.
- V podokně navigátoru, který zobrazuje hierarchii kapitol v dokumentu (a umožňuje přesuny kurzoru na jejich pozice) budou zvýrazněny (barevně či jinou ikonou) ty kapitoly a podkapitoly, ve kterých se nachází zaznamenaná změna. Zde se mírně zjednodušuje požadavek oproti kapitole 5.2.3, a to pouze na zvýraznění změněných kapitol, protože funkce vyhledávání a filtrování bude obsahovat výše zmíněná tabulka se seznamem změn.

## 6.2 Způsob práce v knihovně Qt

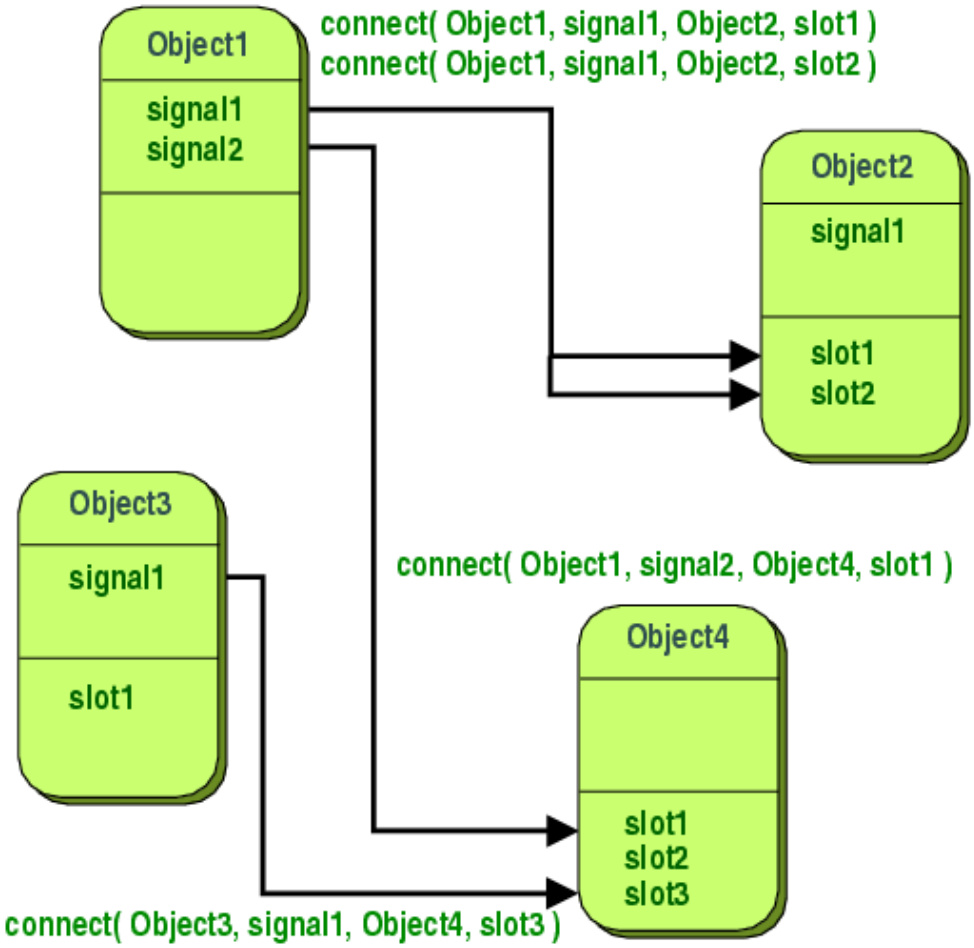
Základním rysem knihovny Qt (a také vlastností, díky níž je tolik populární a využívaná) je způsob komunikace objektů ve vytvořeném systému. Probíhá napojováním *signálů* na *sloty* mezi všemi objekty, které mají jako předka základní třídu knihovny Qt, *QObject*. Tato třída zajišťuje veškerou nízkoúrovňovou funkcionalitu děděných objektů, jako je spojování objektů do hierarchie předků a potomků (výhodné především pro objekty grafického uživatelského rozhraní), zmíněná komunikace mezi nimi, zasílání a zpracovávání událostí.

**Signály** v prostředí Qt reprezentují oznámení o provedení nějaké akce, oznámení o určitém stavu apod. Jsou tedy prostředkem pro signalizaci z nějakého objektu do jiných. Jeden signál vždy patří jedné třídě, je v ní přítomen jako speciální beztypová metoda, která může obsahovat parametry signálu. Signály se, jako jiné metody, mezi třídami dědí.

**Sloty** naopak reprezentují reakce na tyto signály. Jsou to také speciální metody tříd, které mají přidanou vlastnost takovou, aby mohly být pro příjem signálů použity. Lze je ale stejně tak používat jako běžné metody.

Oproti běžnému způsobu napojení tzv. zpětných volání, neboli *callback* mají signály tu výhodu, že poskytují typovou kontrolu při předávání signálů slotům a také nejsou silně vázané. Signály jsou spojovány tak, že jakýkoliv signál může mířit do jakéhokoliv slotu s kardinalitou *n:n*, viz obrázek 6.2. Tyto vztahy lze kdykoliv za běhu programu vytvářet, rušit, nebo blokovat signály určitého objektu, více v [8].

Jak bylo řečeno výše, nutnou podmínkou pro používání tohoto druhu komunikace v určité třídě je ta, že tato třída musí být potomkem třídy *QObject*. V kódu jsou signály, resp. sloty v deklaraci tříd umístěny ve speciálních sekcích rozhraní třídy, v sekci `signals:`, resp. `slots:` (v případě jazyku C++), což jsou nestandardní sekce, zdrojové kódy je tedy nutné překládat speciálním příkazem `qmake`, více v sekci 7.




Obrázek 6.2: Způsob komunikace objektů v prostředí knihovny Qt, zdroj [8]

## 6.3 Struktura aplikace Texmaker

V kapitole 4.3.4 byla již zběžně představená hrubá struktura tříd editoru Texmaker. Nyní se budu zabývat podrobnější strukturou vybraných částí systému, a to těch, které jsou důležité pro začlenění do návrhu vyvíjeného nástroje.

### 6.3.1 Okno textového editoru, jeho rozhraní a komunikace

V okně textového editoru jsou zobrazeny textové dokumenty, přístupné přes záložky, kde každá záložka reprezentuje jeden otevřený soubor. Dále obsahuje widget pro zobrazení čísel řádků (případně záložek) a skrytý widget pro vyhledávání v textu, který je zobrazován pouze v případě žádosti uživatele. Tyto prvky tvoří základní ovládací a zobrazovací rozhraní pro editaci textu, většina komunikace prostřednictvím signálů při editaci tedy probíhá mezi těmito objekty.

Dalším důležitým aplikačním objektem, figurujícím v lokálních komunikačních vztazích při editaci textu je objekt textového zvýrazňovače, který provádí syntaktické formátování textu. Dalšími objekty v tomto vztahu jsou i objekt kontroly pravopisu a vyhledávač souvisejících závorek. Tyto objekty stejně tak okamžitě reagují na změny textu a jsou propojeny

se syntaktickým zvýrazňovačem, ale jejich funkce se nijak nedotýká návrhu vyvíjeného nástroje pro sledování změn.

Hlavním aktérem je zde samozřejmě editační okno, kde je zobrazen text dokumentu, umožňuje navigaci v textu (včetně označování) a primární editaci textu psaním na klávesnici, případně manipulací se systémovou schránkou (operace kopírování, vyjímání a vkládání textu). Toto okno vysílá důležité signály, oznamující změny textového kurzoru a především textové změny v dokumentu.

Všechny tyto objekty (mimo vyhledávací dialog) vždy okamžitě reagují na textové změny dokumentu a provádějí adekvátní operace reagující na tyto změny. Důležitými objekty v těchto vztazích, které je podle požadavků nutné zahrnout do návrhu jsou tedy:

- Textový editor
- Zvýrazňovač textu
- Panel s čísly řádků

### 6.3.2 Změny v textu a možnosti jejich zachycování

Změny textu v dokumentu může primárně iniciovat uživatel při psaní na klávesnici (případně manipulací se schránkou), pokud je okno editoru aktivní. Dále je ale možné měnit dokument pomocí uživatelských akcí z hlavního menu aplikace, nástrojových lišt, bočního okna s rychlým přístupem a také pomocí klávesových zkratek, které lze navíc uživatelsky definovat. Není tedy možné jednoduše vytvořit prostředníka zachycujícího žádosti o změny, protože by bylo nutné měnit fungování většiny objektů aplikace.

Objekt textového editoru ale poskytuje pouze rozhraní pro úpravu a zobrazování textu dokumentu, samotný dokument reprezentuje speciální objekt dokumentu, který je s textovým editorem asociován. Pro manipulaci s textem dokumentu jsou používány tzv. textové kurzory, instancí těchto kurzorů může být vytvořeno teoreticky neomezené množství. Objekt dokumentu i editoru uchovává vždy jeden textový kurzor jako hlavní (ten, který vidí uživatel a na němž jsou prováděny primární změny textu iniciované uživatelem), ale změny v dokumentu může provádět libovolný kurzor.

Nabízí se tedy možnost vytvořit novou třídu jako potomka třídy dokumentu či kurzoru a nahradit jejími objekty ty existující. Dědění třídy dokumentu je sice teoreticky možné, ale požadovaných vlastností bychom nemohli dosáhnout jednoduchým způsobem, také protože neposkytuje téměř žádné chráněné metody k implementování.

Další možností by bylo vytvoření třídy jako potomka třídy kurzoru, ale v tomto případě není možné zanést nový typ do existujících tříd, protože se pro jejich předávání využívá výhradně kopírování, nikoli předávání referencí, což degraduje typ objektu zpět na předka. Nová funkcionalita by tedy nebyla použitelná.

Jedinou možností zachycování změn tedy zůstává reakce na signály objektů dokumentu a textového editoru, jejichž instance je vždy pro každý textový dokument jediná. To s sebou nese nevýhodu nemožnosti zachycení akce před změnou. Tu lze zachytit až jako signál o jejím provedení, což je problém především při mazání textu, protože informace o obsahu smazaného textu se tím ztrácí. Kvůli tomuto omezení tedy bude muset být zachována kopie textu dokumentu, která bude stále obsahovat text před změnou, z něhož bude možné chybějící informace vyčíst a poté kopii aktualizovat.

### 6.3.3 Možnosti zobrazování informací o změnách

V kapitole 5.2.3 byla stanovena úprava grafického uživatelského rozhraní, která bude vyhovovat požadavkům na přehledné a efektivní zobrazování změn a změněných částí souboru. Jak bylo již řečeno, zobrazování změn a ovládacích prvků nástroje pro sledování má být v aplikaci přidáno na několik různých míst:

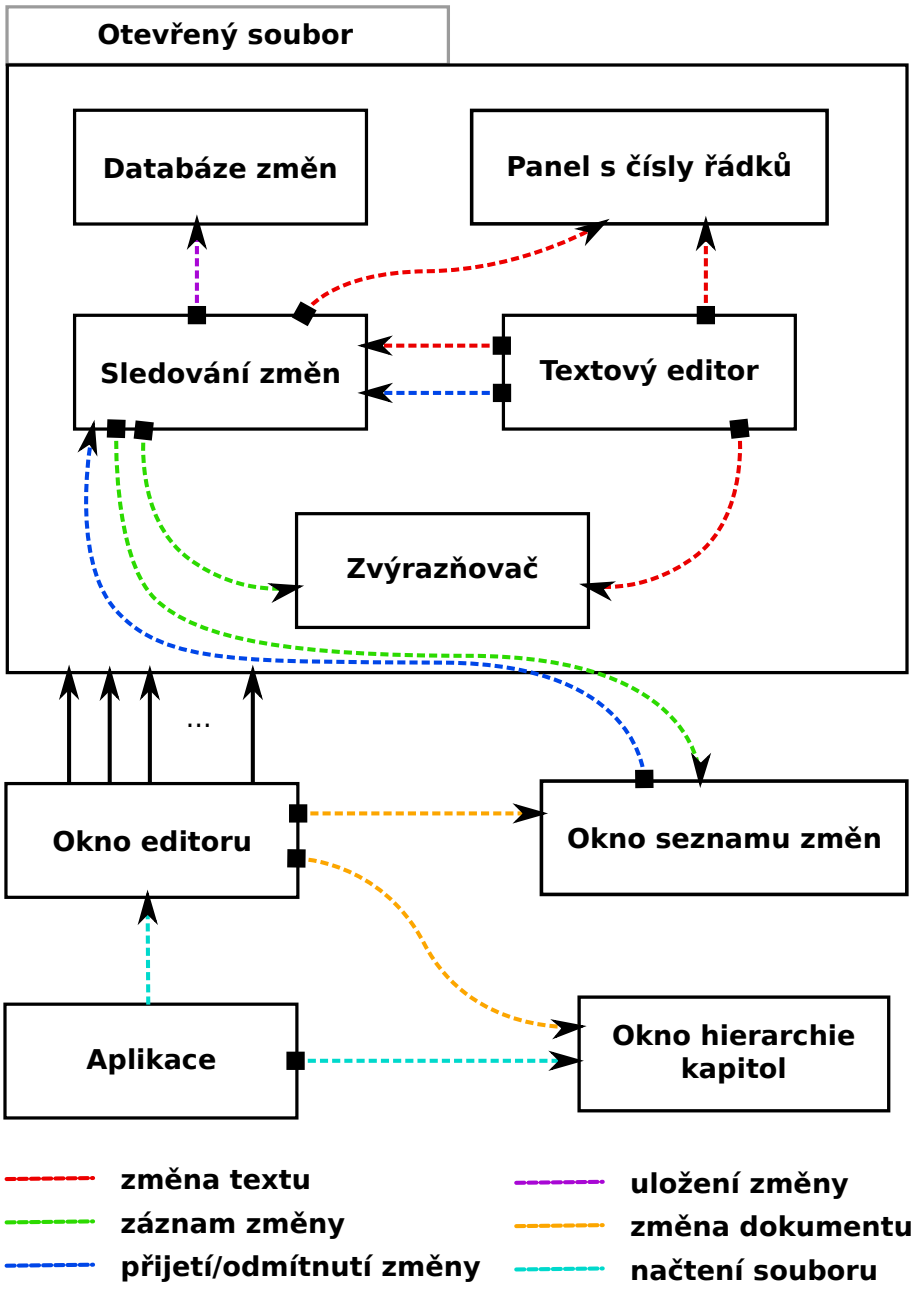
- V kapitole 6.3.1 byl zmíněn objekt syntaktického zvýrazňovače, který má za úkol syntaktické formátování textu. Ten ale není omezen pouze na textové informace, má také přístup k uživatelským datům k právě zvýrazňovanému textovému bloku. Tohoto je v aplikaci již využito pro zobrazování souvisejících závorek. Text je zvýrazňován po blocích a v jedné smyčce zvýrazňovací operace jsou předány vždy jen ty bloky, které byly změněny. Tohoto poznatku lze využít i při zvýrazňování změněných částí textu. V každém textovém bloku mohou být uloženy informace o změnách, vyskytujících se v daném bloku a odpadá tak náročnější zjišťování těchto informací (např. vyhledáváním v databázi změn). Textové bloky odpovídají textovým řádkům dokumentu (tzn. vždy text mezi dvěma konci řádku).

- Další využitelnou částí aplikace pro zobrazování změněných částí dokumentu je postranní okno pro zobrazení hierarchie kapitol. Ta je v aplikaci přepočítávána po načtení souboru a při každém jeho uložení. Hierarchie je zkoumána postupným průchodem všech textových bloků a pozice těch bloků, na kterých je nalezen text odpovídající počátku LaTeXové kapitoly (podkapitoly, apod.), jsou zaznamenány do zobrazovaného stromu hierarchie na příslušnou úroveň. Zde můžeme využít poznatku z předchozího bodu a to přítomnost informací o změnách v bloku pomocí uložených uživatelských dat příslušného bloku. Takto budou moci být detekovány změněné bloky a aktuálně zpracovávaný oddíl (včetně všech jeho nadřazených oddílů) v zobrazené struktuře vhodným způsobem označit jako změněné.

- Posledním a důležitým místem z hlediska funkcionality je okno pro zobrazení celého seznamu změn včetně informací o nich v přehledné tabulce. Toto okno bude reprezentováno vlastní třídou, v tuto chvíli je tedy pouze nutné znát informace o možnostech jeho rozhraní a komunikace se zbytkem aplikace. Důležité je vědět, že toto okno bude v aplikaci existovat pouze v jediné instanci, ale musí zobrazovat informace o aktuálně viditelném dokumentu. Okno textového editoru nabízí signalizaci jak změny záložky a tedy viditelného dokumentu, tak i zavření záložky (dokumentu). Tyto signály musí proto toto okno registrovat a adekvátně na ně reagovat změnou seznamu změn pro aktuálně viditelný dokument. Signály informující toto okno textovým editorem a naopak signály informující textový editor tímto oknem při uživatelské manipulaci se změnami (přijetí, odmítnutí) musí být předávány vždy mezi tímto oknem a aktuálně viditelným editorem. Proto je nutné tyto komunikační vazby dynamicky přepojovat při změně viditelného dokumentu.

## 6.4 Definování rozhraní nástroje v systému

Z předchozí kapitoly vychází směr návrhu rozhraní, které bude muset respektovat zavedené způsoby manipulace s textem v editoru Texmaker i knihovně Qt. Z předchozích poznatků je také patrné, že vyvíjený nástroj nebude možné navrhnout jako čistě oddělenou jednotku, ale bude nutné zapojit jeho části i do stávajících tříd aplikace. Obrázek 6.3 ilustruje nastavení komunikačních vazeb mezi stávajícími třídami aplikace a prozatím abstraktními třídami vyvíjeného nástroje.




Obrázek 6.3: Komunikační vztahy v systému objektů, čárkovanými čárami jsou vyznačeny signály

## 6.5 Návrh revizního systému

V této části bude představen již podrobný a propracovaný návrh tříd, které budou reprezentovat systém pro záznam změn, zobrazování změn, porovnávání a slučování změn. V sekci 6.4 byl již popsán návrh rozhraní a komunikace mezi stávajícími třídami aplikace a vyvíjeným systémem. Z tohoto návrhu bude návrh interní struktury tříd částečně vycházet, musí být uzpůsoben této navržené komunikaci.

### 6.5.1 Sledování a záznam změn

Sledování a záznam změn bude probíhat výhradně na pokyn signálů z objektu textového editoru (resp. dokumentu). Tento signál poskytne informaci o poloze změny a počtu přidaných a odebraných znaků. Tento signál ve třídě pro sledování změn vyvolá akci, která předá data změny databázi k vložení a adekvátním způsobem upraví kopii dokumentu, asociovanou s objektem této třídy. V sekci 6.5.3 je podrobně navrženo přidávání a odebírání sledovaných změn.

### 6.5.2 Databáze změn

Při návrhu datové struktury pro databázi změn bylo nutné zohlednit několik faktorů. Jednotlivé změny v dokumentu budou reprezentovány heterogenními záznamy s metadaty změny, konkrétně pozice, délka, autor, datum a čas a komentář. Tyto záznamy je ale nutné uchovávat v takové formě, aby veškeré operace čtení, mazání a editace byly co nejefektivnější. Proto bylo zvoleno uložení v asociativním poli (implementováno jako strom) s indexem ekvivalentním pozici změny v dokumentu. Jednak je zde výhoda jednoduchého vyhledání, protože nejčastěji je třeba hledat změnu přímo na zadané pozici, nebo v jejím okolí. Uvedené požadavky asociativní pole efektivně plní. Dále je častým požadavkem vyhledání určité souvislé řady změn (seznam změn v textovém bloku, seznam změn v mazaném úseku textu, apod.), což udržování seřazené posloupnosti též umožňuje.

Důležitým faktorem, který je třeba zohlednit je stálá aktualizace pozic uložených změn. Pokud je totiž vložen nebo smazán text na libovolném místě dokumentu, pozice změn vyskytujících se za pozicí této úpravy jsou posunuty o délku této úpravy. Pokud by byly záznamy indexovány pouze celými čísly, reprezentujícími pozice v dokumentu, musely by všechny ovlivněné záznamy být uloženy pod novým klíčem (v tomto případě pod novou pozicí v dokumentu).

Řešení tohoto problému nabízí již zmíněné textové kurzory vázané na instanci objektu textového dokumentu. Všechny vytvořené instance kurzoru vázané na určitý dokument (tzn. ukazují na nějaké místo v dokumentu) udržují ne číselnou, ale logickou pozici, u všech těchto instancí je po každé změně aktualizována číselná pozice automaticky. Textové kurzory navíc obsahují metody implementující porovnávací operátory, takže jsou vhodné k použití na místo asociativních klíčů instantně bez jakýchkoliv úprav. Podmínku zachování pořadí kurzory respektují z podstaty, takže změnou jejich pozic není narušena integrita asociativního pole.

### 6.5.3 Operace nad databází změn

Databáze změn je důležitým prvkem systému a při práci je neustále vytížena, používají ji zprostředkovaně všechny části tohoto systému různými způsoby, v této sekci proto definujeme nutné operace a zdůvodnění jejich využití:

- **Vložení nové změny** - tato operace je nejčastěji prováděnou operací, neboť je spouštěna po každé změně textu dokumentu. Na první pohled se jedná o jednoduchou operaci přidání záznamu do databáze, ale naopak skrývá celou řadu úskalí. Vkládaná změna totiž může ovlivňovat již přítomné změny různými způsoby, konkrétně:
  - Změna vkládající text se může vyskytnout uprostřed jiné změny, která s ní buď je kompatibilní[1], potom je existující změna pouze prodloužena, nebo není kompatibilní a potom je existující změna rozdělena na dvě a mezi ně vložena nová
  - Změna mazající text může mazat část textu, ve které jsou přítomny jiné změny. Mazací změny zůstávají zachovány, ale vkládací změny jsou touto operací mazány, ty, které jsou obsaženy celé jsou odstraněny a ty, které zasahují jen částečně jsou ořezány. Pokud se při této operaci dostanou dvě kompatibilní změny bezprostředně vedle sebe, jsou spojeny v jednu.
  - Dále je třeba připomenout, že záznam změn může být vypnutý, ale již vložené změny zůstávají zachovány. To znamená, že je stále třeba hlídat konflikty měněného textu a existujících změn, podobně jako výše.
- **Přijetí, odmítnutí změny** - tyto operace jsou prováděny především při revizi dokumentu. V porovnání s předchozím typem operace je tato výrazně jednodušší, změna je pouze odebrána z databáze. Jedinou akcí navíc, kterou je třeba provést je promítnutí této přijaté změny do textu dokumentu.
- **Vyhledávání změn** - Operace vyhledání změny může být různého druhu a náročnosti.
  - Nejjednodušší variantou je pouze vyhledání změny začínající přímo na zadané pozici. Toho se využije většinou tehdy, kdy už je pozice známa a je třeba pouze získat informace o změně (např. zobrazení informací v bublinové nápovědě).
  - Mírně složitější variantou je vyhledání změny pod zadanou pozicí, tzn. hledaná pozice se nachází mezi počátkem a koncem změny. Toho bude využito především při vyvolání kontextového menu proto, aby měla aplikace informaci, zda je na tomto místě možné provést akci nad danou změnou.
  - Nejnáročnější variantou je hledání první zasahující změny do daného vymezení. Tato operace již kombinuje vyhledávací a průchodovou funkci (po tomto hledání je možné procházet další změny v pořadí). Tuto metodu využije např. zvýrazňovač textu nebo kontrola při načítání změn ze souboru.
- **Průchody sledem změn** - Jedná se o iterativní operace, kde volající objekt nejprve iniciuje průchod výše uvedeným hledáním a poté opakovaně žádá o další prvek ve sledu. Průchody lze rozdělit na dva typy, jednodušší průchod od začátku seznamu až do konce a složitější, kdy je sled omezen zadaným vymezením.

[1] Kompatibilní změnou se zde rozumí změna stejného typu a stejného autora

### 6.5.4 Ukládání změn do souboru

V sekci 5.2.1 byly diskutovány možnosti ukládání dat o změnách do souboru. Jednoznačné výhody poskytuje řešení, kdy budou změny uloženy spolu s původním textem dokumentu v jednom souboru. S přihlédnutím k navrženým datovým strukturám a typu uložení dat byl stanoven následující formát souboru

```
% CHTRACK [TEXT_LEN] %
[DOC_TEXT]
% CHTRACK CHANGES START %
 ...
%% CH [TYPE], [POS], [LEN], [TIME], '[AUTOR]', '[CMNT]', '[REMOVED]' %%
 ...
% CHTRACK CHANGES END %
```

`TEXT_LEN` odpovídá délce textu původního dokumentu. Proměnná `DOC_TEXT` obsahuje přesný text změněného dokumentu a jeho délka musí odpovídat délce uvedené v hlavičce souboru. Následuje výčet změn, uvozený startovacím řádkem a ukončený ukončovacím řádkem (`CHTRACK CHANGE START` a `CHTRACK CHANGE END`). Jednotlivé změny reprezentují řádky typu `CH ...`, kde:

- `TYPE` je typ změny s hodnotou + (vložení), nebo - (smazání)
- `POS` je pozice změny v dokumentu, `LEN` je délka dané změny ve znacích
- `TIME` je časová známka ve formátu ISO 8601 [13]
- `AUTOR` je jméno autora změny
- `CMNT` je text komentáře k dané změně
- `REMOVED` je smazaný text (pro vkládací změnu prázdné)

### 6.5.5 Slučování a porovnávání souborů

V této sekci budou navrženy postupy, jakým způsobem slučovat a porovnávat dokumenty určené k revizi. Slučování dvou dokumentů je jednodušší proces, má ale omezení (pozitivní omezení) a to totožný původní text dokumentu u obou porovnávaných souborů, což se ale při procesu editace dokumentu více autory očekává.

#### 6.5.5.1 Slučování souborů

Slučování dokumentů bude probíhat tak, že uživatel k již otevřené jedné kopii vybere v dialogu soubor s jinou kopií ke sloučení. V již otevřené kopii se změny vyextrahují z databáze a dokument bude vrácen do původní podoby. Změny z porovnávaného souboru se dekódují do datových struktur, sloučí se do jednoho seznamu společně se změnami z první kopie souboru a déle budou zpracovány společně. Změny se tedy zpracují podle postupu:

1. Změny z aktuálně otevřeného dokumentu budou vzestupně podle pozice vkládány do seznamu (asociativního pole) a zároveň bude vždy změna z textu odebrána tak, aby na konci procesu jako výstup vznikl původní text a seznam zaznamenaných změn.

2. Změny z porovnávaného dokumentu se zpracují analogicky pouze s rozdílným zdrojem dat - v tomto případě budou změny postupně načítány ze seznamu ve zdrojovém souboru.
3. Poté se provede validace původního textu, tzn. zda jsou oba texty identické
4. Nakonec bude z těchto vstupů vytvořen dokument s původním textem následovaným seznamem změn z obou slučovaných dokumentů ve formátu uvedeném v 6.5.4. Takto zpracovaný text bude dekódován stejným způsobem, jako při načtení textu ze souboru.

#### 6.5.5.2 Porovnávání souborů

Porovnávání dokumentů bude probíhat podobným způsobem, porovnávat se však budou dokumenty odlišné od původních, tj. ty které byly editovány bez zapnutí záznamu změn. V těch bude nutné nalézt rozdíly sofistikovaným způsobem, tzn. neporovnávat hrubou silou, ale vyhledat logické části textu, které byly změněny, nebo naopak zachovány. Pro tuto funkci bude využito knihovny třetí strany, konkrétně knihovny *Google diff match patch* [6]. Ta dokáže porovnat dva textové řetězce, přičemž výstupem je seznam bloků textu s informacemi, zda byl blok zachován, vložen, nebo smazán. Průchodem tímto seznamem získáme postupně změny, jejich pozice a délky. Pro průběžné ukládání změn je opět použito asociativní pole. Důležitým prvkem je v tomto procesu stále aktualizovaná pozice, která dovoluje určovat, kde se změny v originálním dokumentu vyskytují.

Bloky jsou zpracovávány následovně:

- Na začátku je nastavena aktuální pozice dokumentu na nulu a jako referenční text je zvolen text originální
- Potom postupně pro všechny bloky:
  - Pokud je blok stejný, jako v originálním dokumentu, aktuální pozice je pouze zvětšena o délku bloku
  - Pokud byl blok smazán, změna je vložena do pole a aktuální pozice je pouze zvětšena o délku bloku
  - Pokud byl blok vložen, změna je vložena do pole ale aktuální pozice zůstává stejná, neboť v originálním dokumentu se text nevyskytuje, takže následující blok bude přímo navazovat na danou pozici.

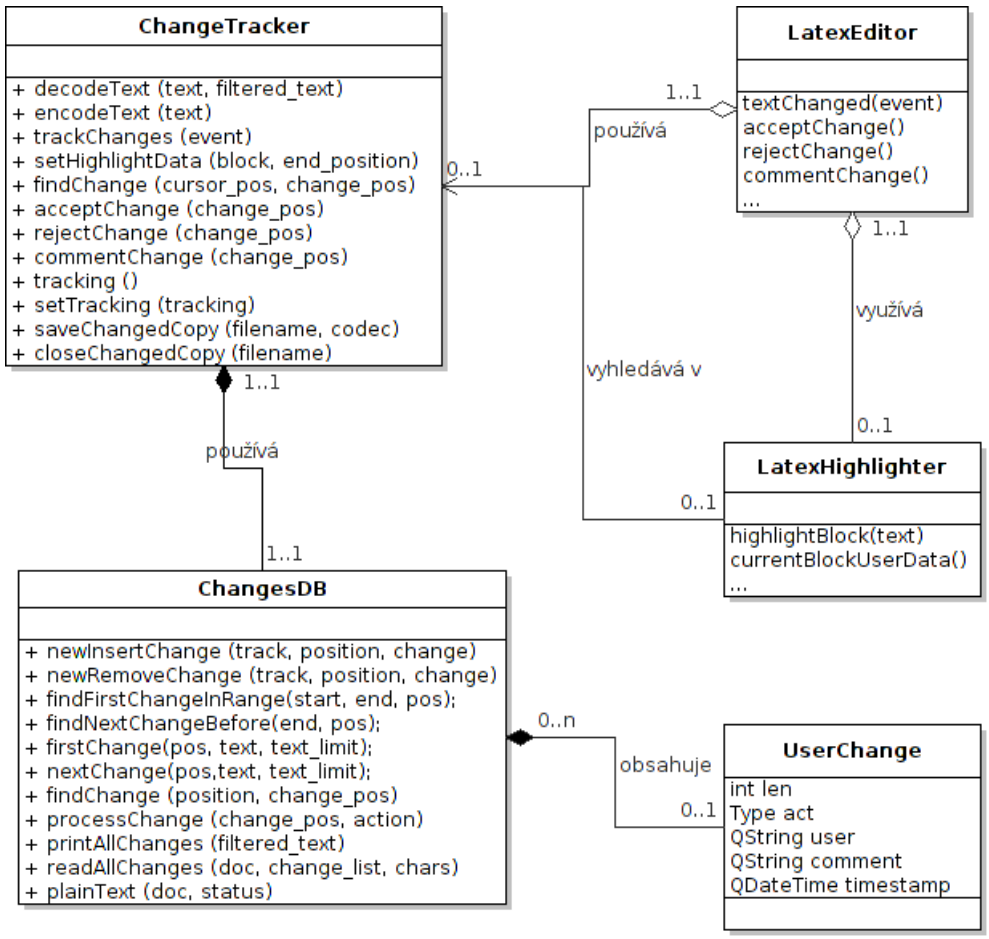
Poté jsou data, stejně jako v případě slučování, transformována do podoby vstupního textu, tzn. text dokumentu následovaný seznamem změn ve formátu uvedeném v 6.5.4, a následně jsou předána stejným způsobem jako při načtení souboru.

## 6.6 Návrhové diagramy

V této kapitole budou vyobrazeny diagramy třídních vztahů v různých částech aplikace. Kompletní diagram tříd celé aplikace (resp. významných tříd) není uveden z prostorových a logických důvodů. Aplikaci je z pohledu vývoje revizního systému vhodnější rozdělit na několik funkčních a logických celků a ty popisovat nejprve nezávisle.

### 6.6.1 Diagram tříd figurujících v zaznamenávání změn

Na obrázku 6.4 je graficky zobrazen diagram tříd části systému, týkající se zaznamenávání změn a jejich zobrazování v editoru.



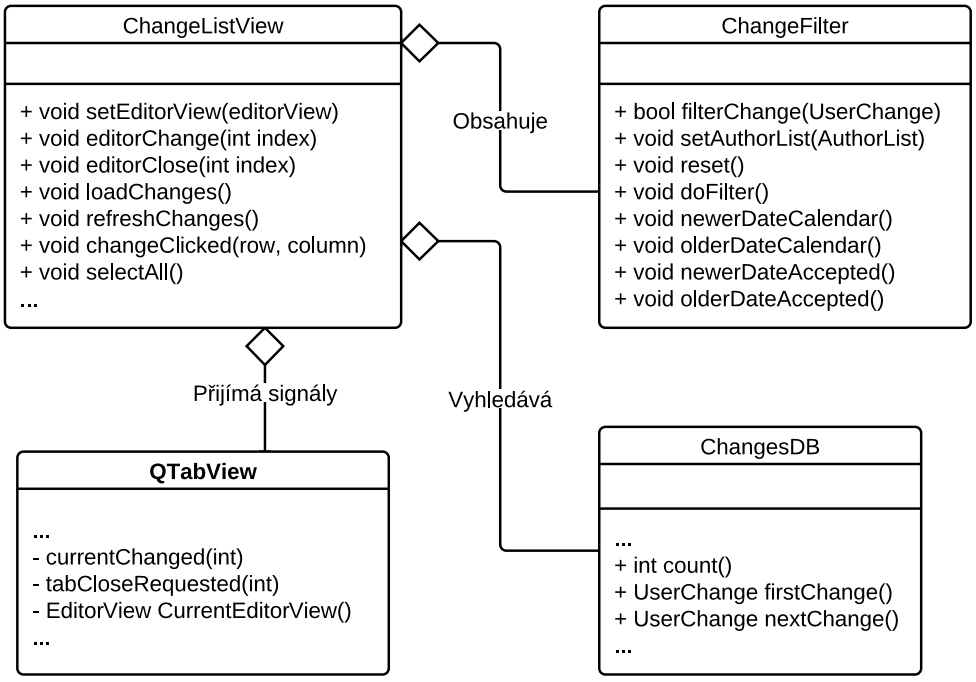

Obrázek 6.4: Diagram tříd zaznamenávajících textové změny

V diagramu lze vidět složení objektů tříd obsluhujících textové změny, které adekvátně zpracovává a ukládá. Každá instance třídy `LatexEditor` obsahuje instanci třídy `ChangeTracker`, který, jak lze vidět z výčtu jeho metod, je hlavním komunikačním bodem při práci se změnami v dokumentu.

Pro ukládání změn a řadu složitějších operací nad nimi uchovává objekt třídy `ChangesDB`, zapouzdřující databázi změn, která se stará o správné vkládání, mazání a revizi změn. Voláním těchto operací dostává objekt třídy `ChangeTracker` zpětné informace o nutných změnách v textu dokumentu kvůli zachování integrity kopií dokumentu navzájem i vůči pozicím a rozsahům zaznamenaných změn. Strukturou, reprezentující záznam o změně je třída `UserChange`, jejíž objekty jsou řazeny do asociativní mapy zapouzdřené uvnitř objektu databáze.

### 6.6.2 Diagram tříd okna seznamu změn

V této sekci je zobrazen diagram tříd, které jsou součástí okna seznamu změn, nebo jsou s ním spjaty asociativními vazbami. Obrázek 6.5 ukazuje rozhraní těchto tříd a jejich vazby.




Obrázek 6.5: Diagram tříd zprostředkovávající funkcionalitu seznamu změn

Z diagramu lze vyčíst, že problematika okna seznamu změn je relativně jednoduchá. Objekt třídy `ChangeListView` je asociován s jediným objektem třídy `ChangeFilter`, která se stará o filtrován zobrazeného seznamu změn. Objekt třídy `ChangeListView` má vždy přístup (přes `QTabWidget`) k aktuální databázi změn, ve které může libovolně číst.

Problémem je stále se měnící vazba mezi seznamem změn a právě aktuální databází změn. Proto mezi objektem seznamu a objektem `QTabWidget`) probíhá komunikace sdělující aktuálně zvolenou záložku s dokumentem a tedy i aktuální asociovanou databázi změn.

# Kapitola 7

# Implementace

V této kapitole je podrobně představeno vývojové prostředí a prerekvizity potřebné k vývoji. Dále jsou zde popsány některé detailní postupy v implementaci, které je vhodné zmínit (především zásahy do již existujících tříd aplikace). Na konci kapitoly je předvedena funkcionalita a grafické uživatelské rozhraní editoru.

## 7.1 Vývojové prostředí a systémové požadavky

Aplikace je vyvíjena, jak bylo zmíněno v kapitole 6.2 s pomocí knihovny (resp. toolkitu) **Qt-toolkit**. Ta poskytuje komplexní zapouzdření velkého množství funkcí, pracujících se složitějšími datovými typy, grafickým uživatelským rozhraním a poskytuje také sofistikovanou komunikaci mezi objekty.

Knihovna **Qt-toolkit** je multiplatformní knihovnou, díky je tomu je multiplatformní i aplikace Texmaker, což je její nespornou výhodou. V průběhu implementace se ale objevily určité nesrovnalosti v chování mezi platformami Windows a Linux. Aplikace byla vyvíjena původně na platformě Linux, pro Windows byla později odladěna. V této podkapitole budou tyto problémy dále zmíněny.

### 7.1.1 Qt Creator IDE

Integrované vývojové prostředí (*Integrated Development Environment - IDE*) **Qt Creator** je přímo určené pro vývoj aplikací s grafickým uživatelským rozhraním jak pro osobní počítače, tak i mobilní telefony s použitím knihovny Qt toolkit. Stejně jako knihovna Qt toolkit je toto vývojové prostředí multiplatformní. Mimo správu a editaci zdrojových kódů se sofistikovanou navigací v nich umožňuje také vizuální tvorbu prvků grafického uživatelského rozhraní.

Dále poskytuje možnost automatizovaného překladu přímo nastaveného pro knihovnu Qt toolkit. Jak bylo zmíněno v kapitole 6.2, při práci s knihovnou se v deklaracích tříd používají speciální sekce pro deklaraci *signálů* a *slotů*. To vyžaduje mezikrok při kompilaci v podobě vytvoření zdrojových souborů přeložitelných ve standardní formě jazyka C++ pro všechny třídy, které dědí z bázové třídy knihovny, a to třídy `QObject`.

Součástí prostředí je také základní debugger, poskytující dostatečné možnosti pro ladění aplikace. Na platformě linux je výhoda automatického nastavení cest všech knihoven a závislostí při použití balíkovacího systému, díky tomu je debugger možné použít prakticky bez dalších nastavení. Naopak na platformě Windows je třeba pro zprovoznění provést nutná nastavení, neboť použité knihovny se nemusí zcela integrovat do systému.

### 7.1.2 Prerekvizity pro sestavení a spuštění aplikace

Pro sestavení aplikace jsou nutnou součástí vývojového prostředí tyto závislosti:

- **Qt toolkit** - Aplikace byla vyvíjena s verzí Qt 4.6 a nově přeložena s verzí 4.7. Aktuálně je dostupná knihovna ve verzi 4.8.1 (včetně dokumentace a dalších zdrojů) na webové adrese `http://qt.nokia.com/`.
- **Knihovna Poppler** - Knihovna používaná aplikací pro zobrazení výstupu LaTeXu ve formátu *.pdf*. Tato knihovna je standardně součástí balíků open source knihoven KDE. V případě použití balíkovacího systému na platformě Linux je situace s integrací knihovny snadná. Na platformě Windows je kompilace samostatné knihovny velice náročná (viz. [1]), je tedy vhodnější instalace balíku KDE doplňkově s hlavičkovými soubory pro vývoj (viz. [4]).
- **Kompilátor C++ včetně nutných knihoven** - V linuxových distribucích je standardně obsažen kompilátor *gcc*, včetně všech nutných knihoven. Pro platformu Windows lze použít např. překladače *GNU MinGW*, či *Visual studio compiler*, který je součástí vývojového prostředí *Microsoft Visual studio*.
- **Google diff-match-patch** - knihovna pro porovnávání textových souborů a hledání rozdílů mezi nimi. Obsahuje i rozhraní pro Qt toolkit (použití řetězců `QString`). Zdrojové kódy knihovny jsou připojené v archivu. Knihovna je dostupná včetně dokumentace na [6].

Dále aplikace vyžaduje přítomnost distribuce LaTeXu, jak bylo zmíněno v sekci 4.3.3. Je ale vyžadována pouze pro generování výstupu, což je v podstatě nutná podmínka pro provoz aplikace, pro její sestavení, či spuštění však nemusí být přítomna.

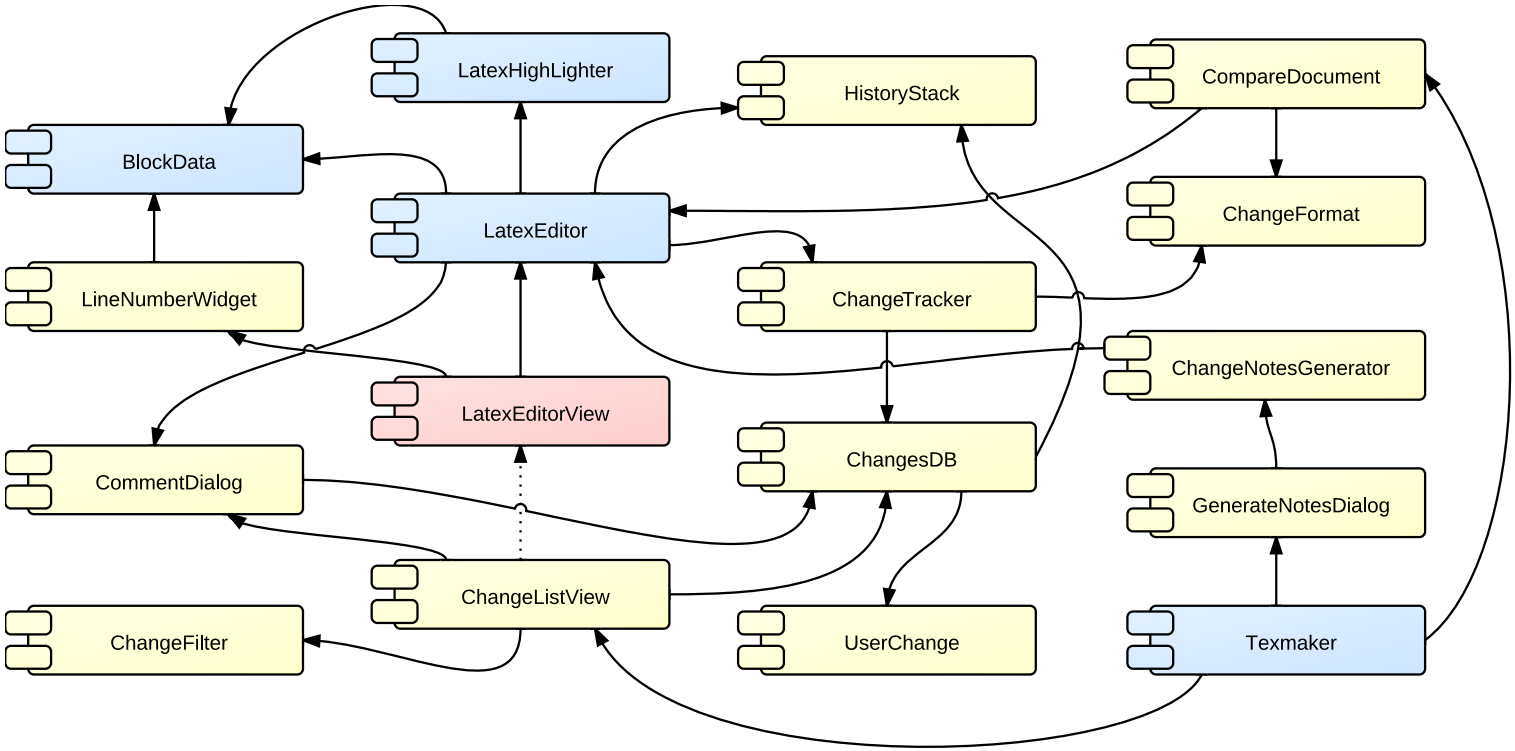
Pro plnohodnotné využití implementovaného rozšíření, včetně doplňkového nástroje *Generování přehledu změn* (viz. 6.1.3 a B.1.2.7) je třeba přítomnosti LaTeXového balíku `TrackChanges` [9]. Tento balík je dále závislý na balících (`color`, `ifthen`, `calc`, `soul`, `morefloats`), které jsou obsaženy buďto v základním souboru balíků dostupných distribucí, nebo v jejich doplňkových repositářích.

## 7.2 Implementace revizního systému

V této kapitole je popsán způsob implementace vybraných částí revizního systému v editoru Texmaker pro hlubší vysvětlení fungování částí systému i jeho celku. V textu jsou hojně citovány odkazy na třídy knihovny Qt toolkit, podrobnou dokumentaci lze nalézt v [7].

### 7.2.1 Souhrnná ilustrace systému komponent

Před samotným popisem implementace jednotlivých částí systému je vhodné jej ilustrovat jako celek. Komponenty systému, reálně analogické k jednotlivým třídám (resp. hlavičkovým souborům zdrojového kódu), lze rozdělit na několik logických skupin. I přesto ale existuje řada závislostí, které se vyskytují i mezi těmito skupinami. Patří mezi ně záznam a zobrazení změn (`LatexEditor`, `ChangeTracker`, `ChangesDB`, `LatexHighlighter`, `HistoryStack`), rozšíření grafického rozhraní aplikace (`ChangeListView`, `ChangeFilter`) nebo práce se soubory (`CompareDocument`, `ChangeNotesGenerator`).




Obrázek 7.1: Souhrnný pohled na systém komponent s vazbami jejich závislostí. Žlutou barvou jsou vyznačeny nově přidané třídy, modrou barvou již existující změněné a červenou nezměněné.

### 7.2.2 Datové struktury pro záznam změn

Záznam změny je v aplikaci reprezentován třídou `UserChange`, obsahující potřebné atributy, tedy *Typ*, *Délku ve znacích*, *Jméno autora*, *Datum a čas* a *Komentář*.

Tyto změny jsou uchovávány a manipulování prostřednictvím třídy `ChangesDB`, která ukládá změny v asociativním poli typu `QMap<QTextCursor, UserChange *>`. Navíc uchovává pracovní kopii textu dokumentu (kvůli synchronizaci textu při mazacích změnám). Výhody tohoto řešení, postup vkládání změn do databáze a další operace nad databází jsou podrobně rozepsány v kapitole 6.5.2. Implementace algoritmů tomuto návrhu zcela odpovídá.

Pro vložení změny třída poskytuje metody `newInsertChange()` a `newRemoveChange()`, jimž jsou předány atributy změny a také lze specifikovat, zda je zapnutý záznam změn.

Třída dále obsahuje metody pro vyhledávání změn a postupný průchod řadou změn v daném rozmezí.

#### 7.2.2.1 Vyhledávání změn podle pozice nebo rozmezí

Pro vyhledání změny pod danou pozicí slouží metoda `findChange()`, která je použita především při použití kontextového menu v textovém editoru. Dále lze vyhledávat změny v zadaném rozmezí pozic; pro to je určena metoda `findFirstChangeInRange()` - Tato metoda vrací první změnu zasahující do zadaného rozmezí, tudíž i takovou, jejíž pozice předchází tomuto vymezení. Tato metoda je v aplikaci široce využívána (např. dekódování souborů - viz. 7.2.6 nebo zvýrazňování syntaxe - viz. 7.2.5). První změnu v databázi lze najít metodou `firstChange()`.

Pro hledání v asociativním poli bez znalosti přesné pozice změny je použita metoda `lowerBound()` třídy `QMap`, která vrací položku s nejbližším nižším klíčem, než hledaným.

Všechny tyto metody zároveň vrací vyhledanou změnu a na tuto změnu nastavují interní iterátor objektu databáze, který lze poté použít při průchodu polem od hledané změny, viz. následující sekce.

#### 7.2.2.2 Průchod polem změn

V případě, že byla použita některá z vyhledávacích metod (viz. předchozí sekce) a ta nalezla změnu, je možné od této změny procházet následující změny v dokumentu ve vzestupném pořadí podle jejich pozice.

Pro tento účel existují ve třídě dvě metody. První a jednodušší variantou, je metoda `nextChange()`, která vrací následující změnu bez určeného rozmezí pozic a opakovaným voláním lze tedy projít až k poslední změně v dokumentu. Naopak metodou `findNextChangeBefore()` lze procházet sledem změn pouze do zadané konečné pozice. Každým voláním libovolné z těchto metod se posunuje interní iterátor objektu databáze změn na další položku pole.

#### 7.2.2.3 Další funkcionalita databáze změn

Třída `ChangesDB` kromě výše zmíněné funkcionality obsahuje také rozhraní pro revizi změny (přijetí - `acceptChange()` a odmítnutí - přijetí - `rejectChange()`), dále je přes tuto třídu přístupný zásobník změn (viz. 7.2.4) a také jsou zde metody pro kódování a dekódování seznamu změn, viz. 7.2.6.

### 7.2.3 Záznam změn

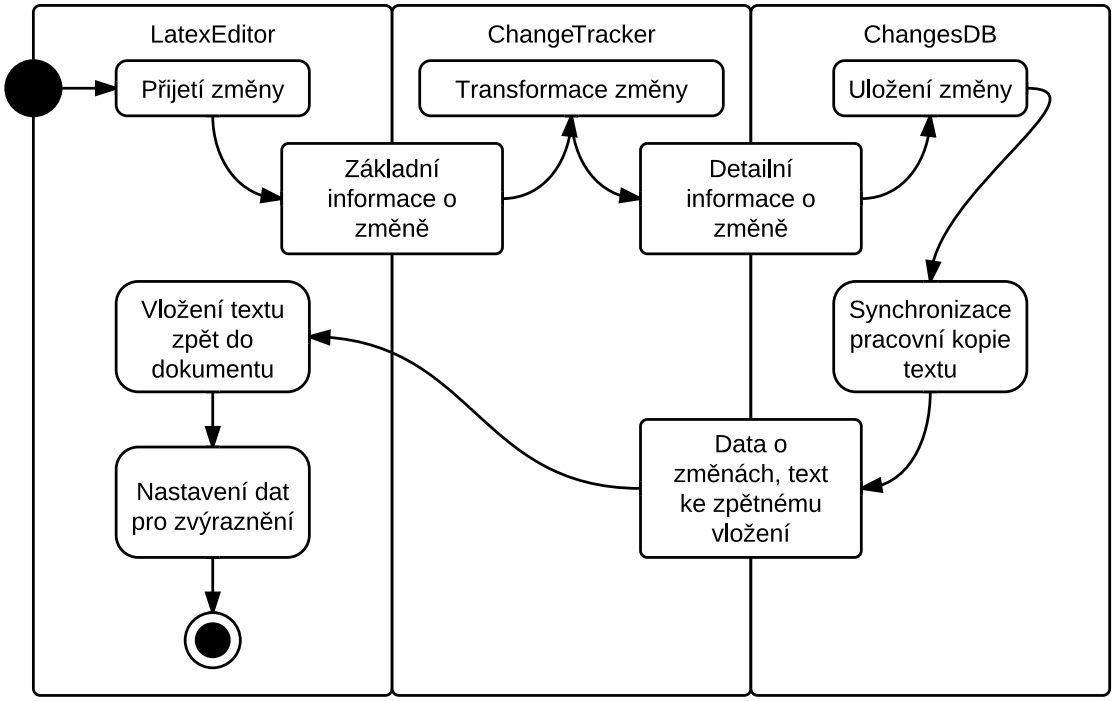
Záznam změn probíhá signalizací textovým editorem o události změny a její následná interpretace do formy záznamu v databázi změn. Pro správné fungování bylo nutné do značné míry upravit třídu aplikace Texmaker `LatexEditor` (potomek třídy `QTextEdit`), reprezentující okno textového editoru.

Zachycení změny probíhá přijetím signálu `contentsChange()` textového dokumentu `QTextDocument` - v tu chvíli je změna vložena do jednoduchého bufferu a následně zpracována po přijetí signálu `textChanged()` textového editoru `LatexEditor`. Toto zpracování bylo zvoleno proto, aby byla vyloučena možnost synchronizačních chyb - mezi zmíněnými signály mohlo dojít k časové prodlevě, ve které mohla přijít již další změna, která by přepsala změnu původní. Tímto jsou změny vždy zaznamenány ve správném pořadí.

Přijatá změna je poté předána metodou `trackChanges()` objektu třídy `ChangeTracker` zajišťující logiku zaznamenávání, tato třída také zprostředkovává řadu operací nad databází změn volajícím externím objektům. Po předání změny je tato objektem zpracována (je zjištěn typ a autor změny) a předána databázi ke vložení. Smazané části textu jsou při vložení do databáze extrahovány z pracovní kopie dokumentu a výstupními parametry předány zpět textovému editoru.

Tři výše zmíněné objekty si lze představit jako vzor *MVC* (*Model-View-Controller*), kde třída `ChangesDB` představuje *model* (má na starosti především správu dat), třída `LatexEditor` zde vystupuje jako *view* (zobrazuje pohled na aktuální text se změnami a poskytuje uživatelské rozhraní pro změny textu) a na konec třída `ChangeTracker` zastupuje *controller* (přijímá uživatelské vstupy a podle nich operuje s daty modelu).

Následující obrázek 7.2 popisuje zjednodušený algoritmus přijetí změny.




Obrázek 7.2: Algoritmus vložení nové změny při úpravě textu uživatelem

### 7.2.4 Historie uživatelských akcí

Kvůli možnosti standardně používat textový editor včetně historie úprav (akce *Zpět* a *Vpřed*) bylo nutné implementovat vlastní zásobník historie, umožňující uchovávat mimo jednoduché akce představující úpravy textu také akce spojené s přidáváním, odebíráním, nebo úpravou změn.

Zásobník je implementován třídou `HistoryStack`, jehož objekt je vždy asociován s objektem třídy `ChangeTracker`. Definuje soubor akcí, jež je možné do něj vkládat. Těmito akcemi jsou:

- Úprava délky změny
- Úprava pozice změny
- Úprava komentáře změny
- Odstranění změny
- Přidání změny
- Odstranění textu
- Přidání textu

Každá akce, která se vyskytne v aplikaci a koresponduje s některou z výše uvedených je do zásobníku uložena pomocí metody `insertOperation()`. Uživatelské akce jsou takto vrstveny do zásobníku jednotlivě - pro oddělení bloků akcí, kde každý blok je brán jako atomická jednotka posloupnosti historie (krok zpět je roven sledu reverzních akcí jednoho bloku), slouží speciální typ akce nazvaný *Separátor*. Ten je vložen podle volání metod `beginEditStep()` a `endEditStep()` (obě metody jsou delegovány i ve třídách `ChangesDB` a `ChangeTracker`).

### 7.2.5 Grafické zvýrazňování změn

Pro zvýrazňování změn je využita již existující třída `LatexHighlighter`, která využívá systému knihovny Qt toolkit, kde každý blok textu (odpovídá jednomu řádku) lze rozšířit libovolným objektem třídy, která je potomkem třídy `QTextBlockUserData`, v tomto případě třída `BlockData`. Do ní byl přidán atribut *Pozice první změny na řádku* typu `QTextCursor`, který obsahuje buď nulovou hodnotu, nebo právě pozici první změny na řádku.

Toho je následně využito ve třídě `LatexHighlighter`, která jakožto potomek třídy `QSyntaxHighlighter` může být asociována s objektem textového editoru a potom je pro každý změněný blok textu automaticky volána metoda `highlightBlock()`, která může podle nastavených uživatelských dat bloku upravit formát textu do žádané vizuální podoby.

Metoda byla upravena pro potřeby revizního systému tak, že pro každý změněný blok je načtena podle nastavené pozice první změna v bloku a použitím vyhledávací a průchodové metody objektu třídy `ChangesDB` (viz. 7.2.2.2) jsou vyhledány všechny změny v bloku a text bloku je poté naformátován do požadované podoby.

Pro potřeby zvýraznění řádků zasažených zaznamenanou změnou v panelu s čísly řádků byla upravena třída `LineNumberWidget`, která v aplikaci vykresluje do panelu čísla řádků, případně značky záložek. Podobně jako při zvýrazňování textu je po každé úpravě textu zaslána událost překreslení okna, kde je jednoduše detekován řádek se změnou pomocí uživatelských dat textového bloku (pozice první změny je nenulová) a číslo řádku je zvýrazněno odlišnou barvou pozadí.

### 7.2.6 Kódování a dekódování souborů

Kódování textu se změnami z datových struktur do textového formátu (viz. 6.5.4) a naopak dekódování textu a jeho změn z textového formátu do datových struktur je v aplikace využíváno na více místech.

Primární a nejdůležitější použití zastává při otevírání a ukládání souborů s dokumenty. Aplikace Texmaker původně pouze otevřela soubor, přečetla jej a text vložila do objektu textového editoru (`LatexEditor`) pomocí metody `setPlainText()`. Při ukládání analogicky přečetla text dokumentu z textového editoru pomocí metody `plainText()` a zapsala jej do souboru.

Jako vhodné se ukázalo řešení, kdy jsou ve třídě `LatexEditor` zmíněné dvě metody znovu implementovány takovým způsobem, aby zprostředkovávaly kódování a dekódování dat automaticky. Takto lze původní kód zachovat ve stejné podobě. Pokud totiž bude právě otevřený soubor obsahovat naformátované změny, bude toto automaticky detekováno v metodě `setPlainText()` a data budou adekvátním způsobem transformována (analogicky pro ukládání souboru).

Při načítání textu jsou v metodě `decodeText()` třídy `ChangeTracker` ze souboru změny přečteny a uloženy do seznamu. Poté je volána metoda `readAllChanges()` třídy `ChangesDB`, která uloží zatím nezměněný text do pracovní kopie a změny ze seznamu postupně ukládá a upravuje podle nich text kopie dokumentu (smazaný text musí být vložen zpět do textu dokumentu, aby mohl být zobrazen). Na konci tato metoda vrací upravený text, který má být zobrazen v textu editoru. Při ukládání textu dokumentu do souboru provede zrcadlově stejný postup.

Těchto výhod je využíváno i v případě slučování a porovnávání dokumentů, viz. 7.2.7.

### 7.2.7 Slučování, porovnávání a generování přehledu

Funkce *Slučování* a *porovnávání* jsou zapouzdřeny ve třídě `CompareDocument`. Funkce *Generování přehledu* je implementována v samostatné třídě `ChangeNotesGenerator`. Všechny tyto funkce používají pro práci vždy aktuálně otevřenou záložku (mj. dostupnou přes metodu `currentEditorView()` třídy `Texmaker`, což je hlavní třída aplikace).

#### 7.2.7.1 Slučování dokumentů

Slučování dokumentů, spustitelné metodou `execMerge()` využívá postup popsaný v 6.5.5.1. Z aktuálního dokumentu je vyextrahován původní text a seznam změn (prostředky popsanými v 7.2.6), ze slučovaného dokumentu stejně tak, ale změny jsou přidávány do stejného seznamu. Poté jsou porovnány otisky obou textů a pokud jsou shodné, je průchodem seznamem vytvořen text ve formátu, který se používá pro ukládání do souboru. Pro jeho načtení lze tedy jednoduše použít metodu `setPlainText()`, jak bylo zmíněno v 7.2.6.

#### 7.2.7.2 Porovnávání dokumentů

Porovnávání dokumentů funguje obdobně, pracuje ale s jinými daty. Funkce porovnává soubory, které mohou mít totožný původní obsah, jejich změny ale nebyly zaznamenány. Vstupním bodem funkce je metoda `execDiff()`, která s použitím knihovny *Google diff-match-patch* [6] vyhledá změny mezi nimi. Tyto změny jsou poté spolu s původním textem podobně jako v případě slučování převedeny do formátu pro uložení do souboru a předány metodě `setPlainText()`.

#### 7.2.7.3 Generování přehledu změn

Tuto funkci lze vyvolat voláním metody `execGenerator()` třídy `ChangeNotesGenerator`. Funkce vytvoří nový dokument, ve kterém jsou změny zvýrazněny LaTeXovými značkami balíku *TrackChanges*. Nejdříve je načten text dokumentu spolu se změnami, ty jsou poté vzestupně procházeny - pro každou změnu je její text nahrazen adekvátní značkou se jménem autora změny. Pro vkládací změny je použita značka \`add[autor]`{`text`}, pro mazací změny je použita značka \`remove[autor]`{`text`}.

V průběhu zpracování změn je průběžně aktualizován seznam autorů zaznamenaných změn. Po vygenerování značek pro změny je doplněna hlavička, kde kromě zahrnutí balíku *TrackChanges* je nutné přidat postupně všechny dále použité autory příkazem \`addeditor` {`autor`}. Hlavička je vložena za poslední výskyt příkazu \`usepackage`, příp. úvodního příkazu \`documentclass`. Pokud není žádný výskyt nalezen, je vložena na začátek textu.

Jak bylo řečeno v 7.1.2, toto rozšíření lze použít pouze v případě, kdy je přítomen balík *TrackChanges* spolu s jeho závislými balíčky.

### 7.2.8 Okno seznamu změn

Okno seznamu změn je v aplikaci zastoupeno třídou `ChangeListView`, které spolu s filtrem změn (třída `ChangeFilter`) tvoří funkční obsah tohoto okna.

Seznam změn je v grafickém rozhraní oproti textovému editoru spolu s databází změn reprezentován pouze jedinou instancí. Zatímco v případě textového editoru se při přepnutí záložky se souborem přepne kontext automaticky, kontext okna seznamu změn musí být upraven uživatelsky. Toho je v praxi docíleno tím, že okno je při konstruování objektu aplikace napojeno na signály `tabCloseRequested()` a `currentChanged()` objektu třídy `QTabWidget` (Zde objekt pro zobrazení více otevřených dokumentů v jednom okně).

Při každém obdržení jednoho z těchto signálů je v metodě `editorChange()`, resp. `editorClose()` zkontrolováno, zda se změna týká aktuálně otevřené záložky s textovým editorem. Pokud ano, musí být odpojeny již neaktuální vazby na zavřený, nebo skrytý editor a vytvořeny nové vazby na aktuálně viditelný editor.

Na aktuální editor je seznam změn připojen signálem editoru changeDone(), což je speciální signál oznamující dokončení jakékoliv operace změny, nebo revize dokumentu. Po přijetí tohoto signálu je vyvolána metoda `loadChanges()` znovu načte aktuální seznam změn v dokumentu. Aby se zamezilo příliš častému znovu-načítání seznamu, v objektu je přítomen časovač, který dovolí načíst změny až $500ms$ od poslední změny - při každé nově příchozí změně je časovač opět resetován na nulu - tímto je zaručeno, že například při rychlejším psaní nebude seznam načten po každém vloženém znaku, ale až uživatel udělá přestávku v psaní.

## 7.3 Ukázka aplikace

V této podkapitole, která uzavírá práci, je představena výsledná rozšířená aplikace Texmaker jak ve vizuální formě, tak také ve formě ukázkového výstupu, který je schopna vyprodukovat. Nachází se zde pouze ilustrační příklady, uživatelská dokumentace popisující plnou funkcionalitu rozšíření se nachází v příloze B.

### 7.3.1 Grafické uživatelské rozhraní

Na obrázku 7.3 je zobrazeno okno aplikace s popisem jednotlivých uživatelských prvků, týkajících se implementovaného rozšíření.

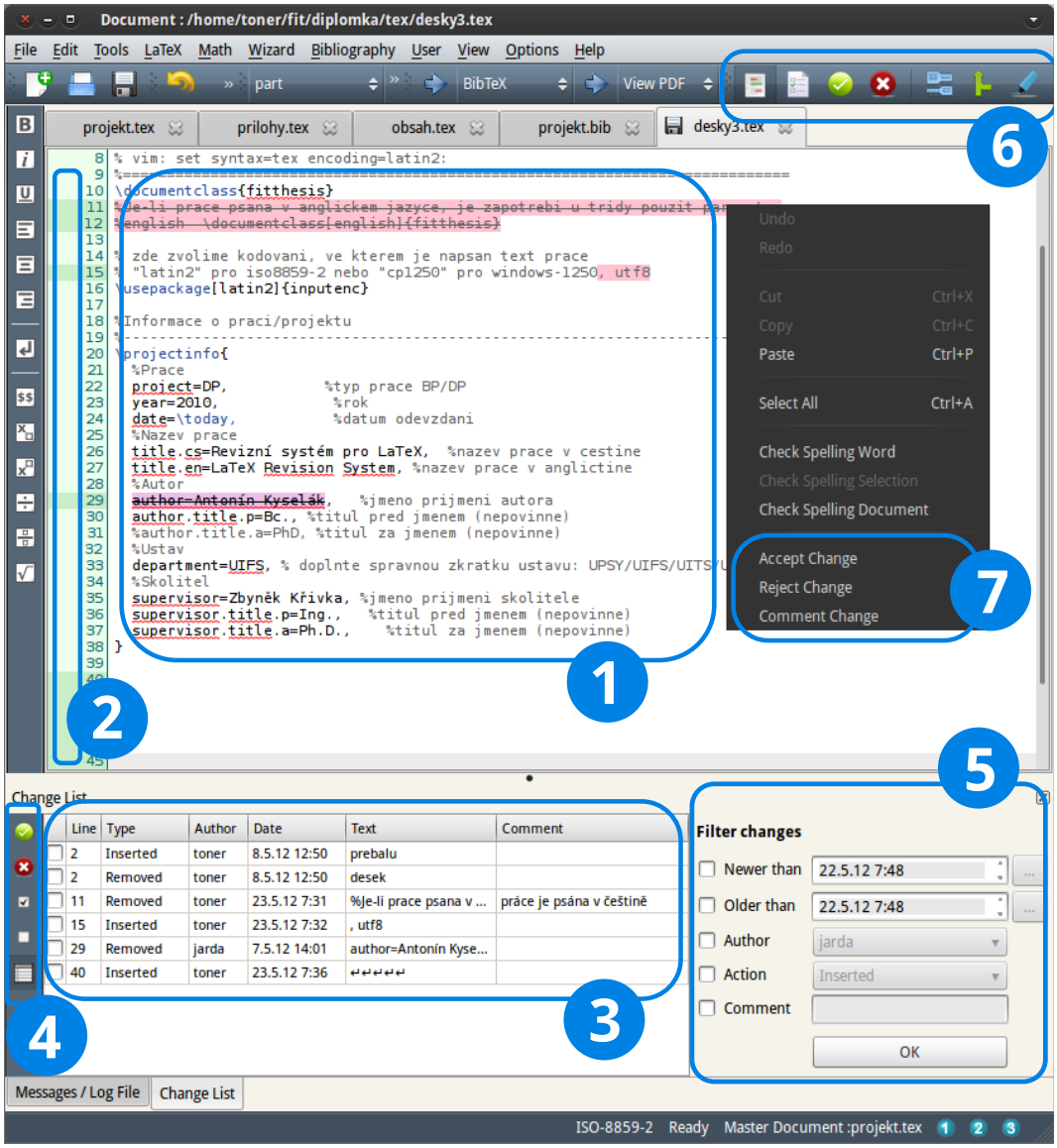
Jak je na první pohled patrné, rozšíření je nenásilně, ale dostatečně integrováno do editoru Texmaker tak, že pokud uživatel chce pracovat bez použití rozšíření, po vypnutí záznamu změn a zavření okna se seznamem změn ve spodní části uživatel jeho přítomnost téměř nepocítí.

Naopak z pohledu uživatele, který chce provádět revizi s využitím sloučení několika verzí dokumentu a následného průchodu seznamem změn editor poskytuje prostředky pro provedení takové operace během okamžiku.

### 7.3.2 Ukázkový soubor

Na obrázku 7.3 je vyobrazena aplikace s testovacím souborem. V této sekci je podobný ukázkový soubor prezentován ve formátu, jaký je použit pro ukládání dokumentu se zaznamenanými změnami. Vlastní text dokumentu je kvůli lepší přehlednosti zkrácen, důležitý je výpis seznamu změn v řádkovém formátu, definovaném v 6.5.4.

```
01 % CHTRACK 1565 %
02 %============================================================================
03 % tento soubor pouzijete pro vysazeni přebalu
04 % (c) 2008 Michal Bidlo
05
06 ...
07 ...
08 ...
09
10 \begin{document}
11   % Vysazeni desek
12   % ------------------------------------------------
13   \makecover
14 \end{document}
15
16 % CHTRACK CHANGES START %
17 %% CH   +,    116,      8, 2012-05-08T12:50:29, 'toner', '', '' %%
18 %% CH   -,    124,      5, 2012-05-08T12:50:27, 'toner', '', 'desek' %%
19 %% CH   -,    506,    121, 2012-05-23T09:11:07, 'toner', 'práce je psána v
   češtině', '%Je-li prace psana v anglickem jazyce, je zapotrebi u tridy pouzit
   parametr &n&%english  \documentclass[english]{fitthesis}' %%
20 %% CH   +,    618,      6, 2012-05-23T09:11:00, 'toner', '', '' %%
21 %% CH   -,   1044,     22, 2012-05-07T14:01:01, 'jarda', '', 'author=Antonín
   Kyselák' %%
22 % CHTRACK CHANGES END %
```




Obrázek 7.3: Výsledný pohled na okno aplikace s popisem jeho částí

1. **Textový editor** s možností editace textu a zvýrazněním zaznamenaných změn
2. **Panel s čísly řádků** se zvýrazněnými řádky, zasažené změnou
3. **Okno seznamu změn** s úplným výpisem informací o všech změnách v dokumentu
4. **Nástroje seznamu změn** umožňují provádět hromadné operace nad změnami
5. **Filtr seznamu změn** omezující výběr seznamu podle požadovaných kritérií
6. **Nástroje rozšíření** shrnující veškeré důležité funkce rozšíření
7. **Kontextové menu** pro rychlé zobrazení akcí nad vybranou změnou

# Kapitola 8

# Závěr

Tento text obsahuje informace a poznatky, popisující vývoj nástroje pro sledování změn při vytváření dokumentů v systému LaTeX. Byly zde popsány všechny nástroje, vztahující se, nebo dokonce nutné pro tento projekt.

Práce pokračuje shrnutím a stručným návrhem aplikace, která je praktickým výsledkem této diplomové práce. V další části práce je představen detailní návrh struktury tříd nástroje, jejich funkčnosti a integrace do struktury editoru.

Tato práce dala vzniknout jednoduchému nástroji, umožňujícímu týmově pracovat na dokumentech psaných v jazyce LaTeX. Z velké části vychází z již existujících řešení, pracujících především na poli robustních textových editorů pro tvorbu dokumentů se zobrazením výsledného formátování v reálném čase (takzvané WYSIWYG editory), které jsou součástí komplexních kancelářských balíků. Nástroje použitelné do této doby pro týmovou práci na textových dokumentech v jazyce LaTeX jsou příliš univerzální, tzn. použitelné genericky pro jakékoliv textové (případně binární) soubory. Tyto nástroje ale neposkytují vhodné rozhraní pro jednoduchou a přehlednou práci s větveným vývojem dokumentů a jejich revidování.

Vyvinutý nástroj poskytuje základ pro takovýto druh software, který na trhu zatím není vůbec rozšířen. Jeho přínosem je tedy integrace do vybraného textového editoru pro LaTeX, Texmaker, který je relativně značně rozšířen (patří mezi nejpoužívanější editory) a tím dává možnost velké části uživatelé k jeho používání.

## 8.1 Diskutovaná rozšíření nástroje

Vyvinutý nástroj je pouze základním kamenem; pro rozvinutí v opravdu silný a dobře použitelný nástroj je třeba jej do budoucna doplnit o funkcionalitu především v oblasti grafického uživatelského rozhraní.

Dále by bylo vhodné jej navrhnout více genericky, popř. přímo jako knihovnu poskytující rozhraní pro obecnou práci se změnami a revizemi dokumentů.

Diskutovatelné mezery vhodné k dalšímu studování bychom nalezli v oblasti výkonu a efektivnější práce s datovými strukturami pro lepší odezvy náročnějších operací v reálném čase, jejichž občasná těžkopádnost může působit uživateli problémy.

V neposlední řadě je nutné zmínit výhody vyšší úrovně konfigurovatelnosti, která umožňuje uživateli přizpůsobit si chování a ovládání nástroje svým návykům, což tento nástroj v této chvíli neumožňuje.

# Literatura


[1] Adin Riviera: Compiling Poppler on Windows [online]. http://laconsigna.wordpress.com/2011/07/14/compiling-poppler-on-windows/, 2011.

[2] Christian Schenk: MikTeX 2.8 Manual [online]. http://docs.miktex.org/2.8/manual/, 2009.

[3] Jan Přichystal: TexOnWeb dokumentace [online]. http://tex.mendelu.cz/navod.pl, 2011.

[4] KDE TechBase: Projects/KDE on Windows/Installation [online]. http://techbase.kde.org/Projects/KDE_on_Windows/Installation, 2011.

[5] Microsoft: Sledování změn během úprav [online]. http://office.microsoft.com/cs-cz/word-help/sledovani-zmen-behem-uprav-HA001218690.aspx, 2004.

[6] Neil Fraser: Diff, Match and Patch libraries for Plain Text. http://code.google.com/p/google-diff-match-patch/, 2011.

[7] Nokia Corporation: Qt 4.7 API Reference - All Classes [online]. http://qt-project.org/doc/qt-4.7/classes.html, 2011.

[8] Nokia corporation: Qt 4.7: Signals & slots [online]. http://doc.qt.nokia.com/latest/signalsandslots.html, 2011.

[9] Novimir Pablant, Felix Salfner: TrackChanges Help [online]. http://www.xm1math.net/texmaker/doc.html, 2009.

[10] Petr Baudiš: Výlet do říše verzí [online]. http://www.root.cz/serialy/vylet-do-rise-verzi/, 2004.

[11] Petr Matouch: Instalace TeXu pro Windows [pdf]. , 2003.

[12] Wikipedia: Comparison of TeX editors. http://en.wikipedia.org/wiki/Comparison_of_TeX_editors, 2011.

[13] Wikipedia: ISO 8601. http://cs.wikipedia.org/wiki/ISO_8601, 2011.

[14] Wikipedia: Revision control [online]. http://en.wikipedia.org/wiki/Revision_control, 2011.

# Příloha A

# Obsah CD

Následující výčet popisuje adresářovou strukturu umístěnou v kořenovém adresáři přiloženého CD s příslušným popisem obsahu.

- **src/** - Zdrojové kódy aplikace
- **doc/** - generovaná dokumentace nově vytvořených tříd
- **help/** - manuál k ovládání revizního nástroje
- **samples/** - ukázkové soubory k demonstraci nástroje
- **README** - soubor s pokyny k orientaci v obsahu CD
- **INSTALL** - soubor s popisem postupu překladu aplikace

# Příloha B

# Uživatelská příručka

Tato uživatelská příručka popisuje ovládání a popis funkcí nástroje pro revizi změn dokumentů pro LaTeXv textovém editoru Texmaker. Rozšíření dovoluje práci více uživatelů na jednom dokumentu formou záznamu změn u jednotlivých verzí a jejich následnému sloučení a revizi. K tomuto je editor vybaven řadou funkcí.

Pro běžné funkce editoru použijte uživatelskou příručku k editoru Texmaker dostupnou online na adrese `http://www.xm1math.net/texmaker/doc.html`.

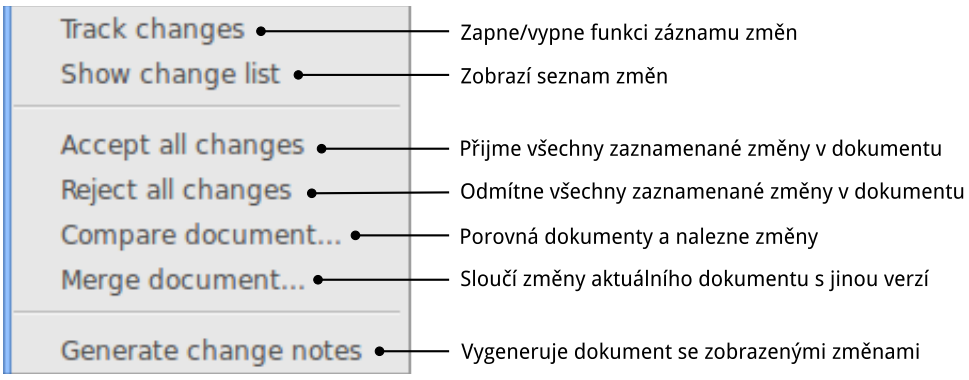
## B.1 Ovládací prvky

### B.1.1 Textový editor

Rozšíření je přímo integrováno v textovém editoru aplikace Texmaker. Podrobný popis funkce textového editoru se nachází v sekci B.2.

### B.1.2 Menu aplikace

Funkce rozšíření se nacházejí v sekci hlavního menu: *Edit* → *Change tracking*. Na obrázku B.1 je zobrazen popis jednotlivých položek menu.




Obrázek B.1: Popis jednotlivých funkcí rozšíření v hlavním menu aplikace Texmaker

#### B.1.2.1 Zapnutí záznamu změn

Tento přepínač nastavuje a indikuje, zda má editor provádět záznam změn při psaní. Pokud je nastaven, editor zaznamenává všechny změny pod právě přihlášeným uživatelem. Při otevření souboru, který obsahuje zaznamenané změny se tento přepínač automaticky zapíná. Stejně tak se automaticky zapne, pokud uživatel provede *sloučení dokumentů* (B.3.1), nebo *porovnání dokumentů* (B.3.2).

#### B.1.2.2 Zobrazení seznamu změn

Zobrazí seznam změn v dokumentu s dodatečnou funkcionalitou, viz. B.4.2.

#### B.1.2.3 Přijetí všech změn

Přijme všechny změny zaznamenané v dokumentu - změny budou odstraněny ze záznamu, vložené texty zůstanou v dokumentu a smazané texty budou odstraněny. Více viz. B.4.

#### B.1.2.4 Odmítnutí všech změn

Odmítne všechny změny zaznamenané v dokumentu - změny budou odstraněny ze záznamu, vložené texty budou odstraněny a smazané texty zůstanou v dokumentu. Více viz. B.4.

#### B.1.2.5 Porovnání dokumentu

Vyvolá dialog pro výběr souboru k porovnání. Podrobněji je toto vysvětleno v B.3.2.

#### B.1.2.6 Sloučení dokumentu

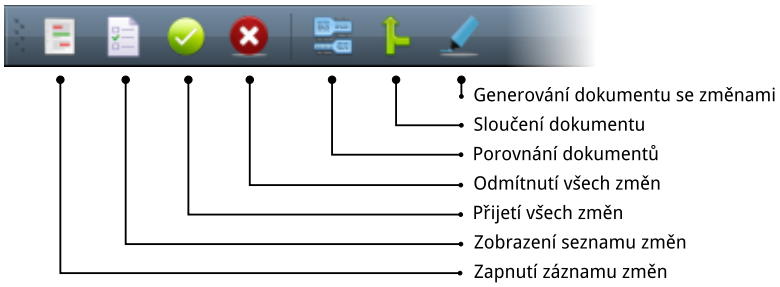
Vyvolá dialog pro výběr souboru ke sloučení. Podrobněji je toto vysvětleno v B.3.1.

#### B.1.2.7 Generování přehledu změn

Vygeneruje dokument, ve kterém jsou vyznačeny zaznamenané změny pomocí speciálních značek LaTeXu. Podrobné informace se nacházejí v sekci B.2.3.

### B.1.3 Lišta nástrojů

Pro rychlejší přístup k funkcím nacházejících se v hlavním menu aplikace slouží ovládací tlačítka na nástrojové liště hlavního okna. Popis tlačítek se nachází na obrázku B.2.




Obrázek B.2: Popis jednotlivých funkcí rozšíření v nástrojové liště aplikace Texmaker

Funkce nacházející se na nástrojové liště přesně odpovídají funkcím popsaným v sekci pojednávající o hlavním menu aplikace, viz. B.1.2.

### B.1.4 Strom kapitol

V levé části hlavního okna aplikace se nachází panel se stromově uspořádanou hierarchií kapitol. Pro snadnější orientaci jsou kapitoly, pod kterými se nachází zaznamenané změny, zvýrazněny červenou ikonou, namísto modré.

## B.2 Editování dokumentu

Záznam změn probíhá v reálném čase úpravou dokumentu v okně textového editoru, pokud je zapnutá funkce *záznam změn* (viz. B.1.2.1). Pokud je tato funkce vypnutá, během editace dokumentu nejsou nové změny zaznamenávány, mohou být ale smazány nebo zkráceny již zaznamenané, pokud se v dokumentu vyskytují.

Psaním textu je zaznamenávána *vkládací změna*, odstraňováním textu je naopak zaznamenávána *mazací změna*. Pokud je přepsána část textu - například pokud je označená část textu a přes ní je vložen jiný text standardní funkcí *Vložit* - jsou vytvořeny dvě změny, nejprve změna mazací a poté vkládací vedle sebe.

Bezprostředně navazující bloky změn stejného typu (vložení/smazání textu, autor změny) jsou vždy sloučeny do jednoho bloku a poté reprezentovány jediným záznamem.

Při editaci se záznamem změn lze použít historii příkazů stejně, jako při klasické úpravě dokumentu. Zaznamenanou změnu lze vzít zpět použitím příkazů *Zpět* a *Vpřed*.

### B.2.1 Zobrazení změn

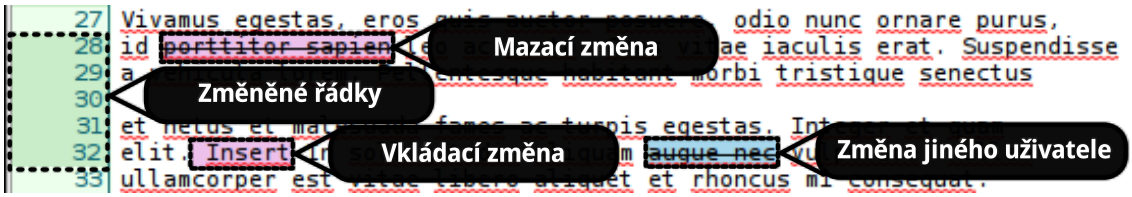
Zaznamenané změny jsou v textovém editoru zvýrazněny - jak barvou pozadí textu, tak také stylem. Původní zvýraznění syntaxe značek LaTeXu je zachováno v popředí.

Všechny změny jsou zvýrazněny barvou pozadí, jednotlivé barvy odlišují autory - každý autor má přiřazenu jednu barvu z přednastavené škály.

Mazací změny jsou pro odlišení od vkládacích zvýrazněny středovou horizontální čárou (přeškrtnutím), po celou dobu existence záznamu je text změny zachován v dokumentu.

Posledním zvýrazňovacím prvkem v okně textového editoru je panel s čísly řádků. Všechna čísla řádků, zasažených libovolnou zaznamenanou změnou jsou zvýrazněny podbarvením. Toto je vhodné především pro zvýraznění vložených či smazaných prázdných řádků, které jakožto neviditelné znaky nemohou být zvýrazněny barvou pozadí.

Následující obrázek B.3 ilustruje výčet možností zobrazení změn.




Obrázek B.3: Zobrazení změn v textovém editoru

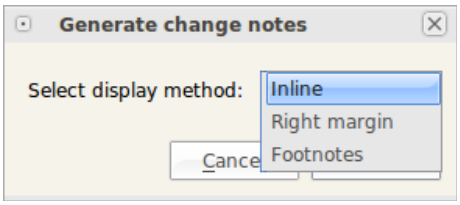
### B.2.2 Komentáře

Ke každé zaznamenané změně lze vložit uživatelský komentář, což může přispět ke snadnější orientaci při pozdější revizi dokumentu.

Komentář lze vložit výběrem možnosti *Comment change* v kontextovém menu, vyvolaném stiskem pravého tlačítka myši nad zvolenou změnou. Podobně lze komentář vložit pomocí kontextového menu nad řádkem v seznamu změn, kde jsou poté také zobrazeny, viz. B.4.2.

### B.2.3 Generování přehledu změn

Tuto funkci lze vyvolat z hlavního menu aplikace (sekce B.1.2.7) nebo pomocí tlačítka nástrojové lišty (sekce B.1.3). Po spuštění je zobrazen dialog pro výběr zobrazovací metody (obr. B.4), dostupné možnosti jsou *Zobrazení přímo v textu (Inline)*, *Zobrazení v pravém sloupci (Right margin)* a *Zobrazení v poznámkách pod čarou (Footnotes)*.




Obrázek B.4: Dialog pro výběr zobrazovací metody generování poznámek ke změnám

Po potvrzení dialogu je z aktuálně otevřeného dokumentu do nové záložky vygenerován upravený dokument obsahující speciální značky pro vizualizaci zaznamenaných změn. Tento dokument lze poté uložit a stejně jako jiný dokument přeložit příkazem LaTeXu (viz `http://www.xm1math.net/texmaker/doc.html`).

Na obrázku B.5 je ilustrován příklad s použitím zobrazovací metody *Zobrazení přímo v textu (Inline)*.

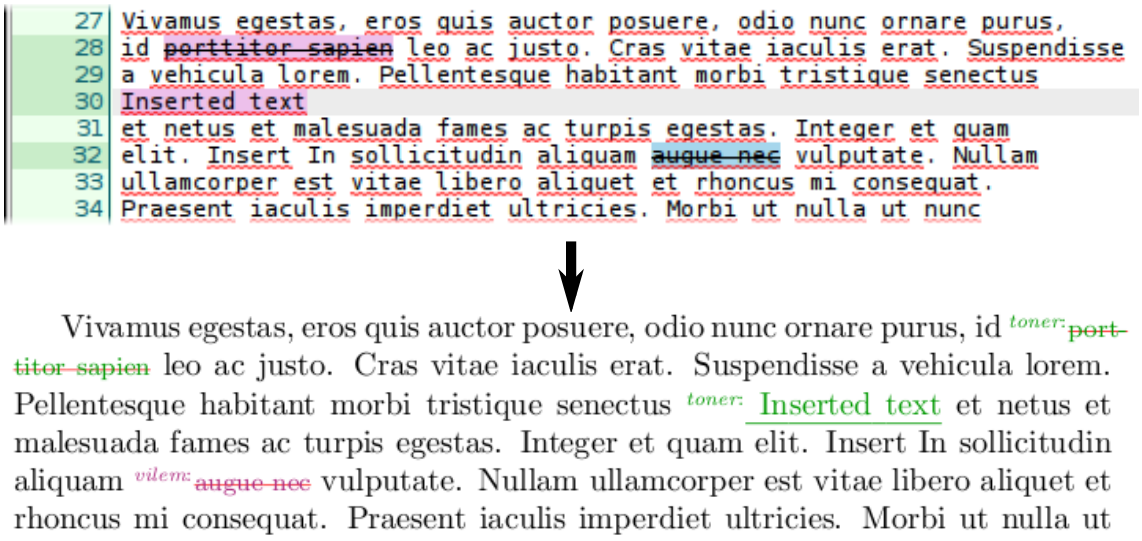
## B.3 Spojování dokumentů

V této sekci je popsána práce s různými verzemi stejného dokumentu pomocí *Slučování dokumentů* a *Porovnávání dokumentů*.

### B.3.1 Sloučení dokumentů

Tuto funkci lze vyvolat z hlavního menu aplikace (sekce B.1.2.6) nebo pomocí tlačítka nástrojové lišty (sekce B.1.3).

Funkce *Sloučení dokumentů* umožňuje sloučit zaznamenané změny ve více dokumentech, pokud jsou vytvořeny ze **stejného originálního dokumentu** (tzn. v obou dokumentech musí být zaznamenány všechny změny, které byly provedeny v původním dokumentu) - v jiném případě nelze sloučení provést, neboť již není možné jednoduše správně zaměřit pozice změněných bloků.




Obrázek B.5: Příklad generování poznámek ke změnám

#### B.3.1.1 Postup slučování dokumentů

- Nejprve je otevřena jedna z verzí dokumentu a je zobrazena v aktuální záložce editoru
- Poté je spuštěna funkce *Sloučení dokumentů*, která vyzve uživatele k výběru souboru s další verzí dokumentu.
- Po potvrzení výběru dojde ke sloučení změn v obou verzích dokumentu - změny jsou zobrazeny v textovém editoru (B.2.1) a v seznamu změn (B.4.2)

Tímto způsobem lze postupně sloučit neomezené množství jednotlivých verzí změněného dokumentu.

### B.3.2 Porovnání dokumentů

Tuto funkci lze vyvolat z hlavního menu aplikace (sekce B.1.2.5) nebo pomocí tlačítka nástrojové lišty (sekce B.1.3).

Funkce *Porovnání dokumentů* umožňuje porovnat originální verzi dokumentu a jeho změněnou verzi **bez zaznamenaných změn**. Porovnáním jsou nalezeny změny oproti původnímu dokumentu a ty jsou poté zaznamenány standardním způsobem v textovém editoru. Jako autor změny je zaznamenán aktuálně přihlášený uživatel.

#### B.3.2.1 Postup porovnání dokumentů

- Nejprve je otevřena původní verze dokumentu a je zobrazena v aktuální záložce editoru
- Poté je spuštěna funkce *Porovnání dokumentů*, která vyzve uživatele k výběru souboru se změněnou verzí dokumentu.

- Po potvrzení výběru dojde k porovnání změn v obou verzích dokumentu - změny jsou transformovány, zaznamenány a zobrazeny v textovém editoru (B.2.1) a v seznamu změn (B.4.2)

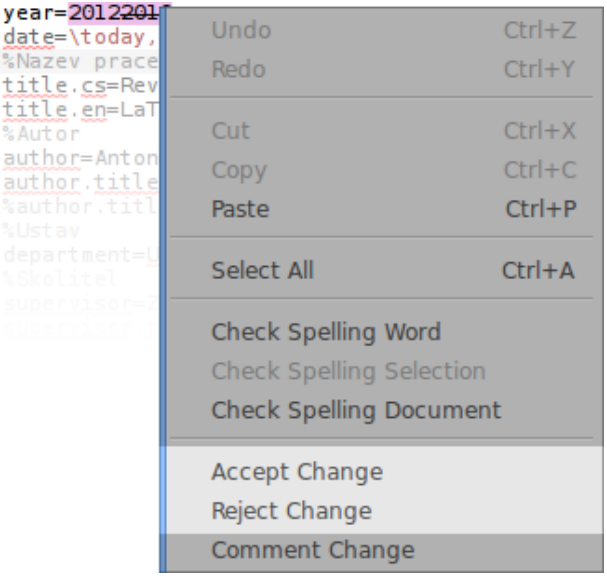
Porovnání nelze provádět v dokumentech obsahujících zaznamenané změny kvůli nejednoznačné interpretace změněných částí dokumentu a jejich pozic.

## B.4 Revize dokumentu

Pro revizi dokumentu (neboli postupný průchod zaznamenanými změnami v dokumentu a jejich následné potvrzení, či odmítnutí) existuje v tomto rozšíření především panel se seznamem změn (B.4.2). Mimo to lze změny revidovat interaktivně přímo v textovém editoru nebo provést souhrnné přijetí či odmítnutí všech změn pomocí funkcí v hlavním menu aplikace (B.1.2) nebo v nástrojové liště aplikace (B.1.3).

### B.4.1 Revize v textovém editoru

Přijetí či odmítnutí změny v textovém editoru lze provést stlačením pravého tlačítka myši nad vybranou změnou a výběrem možnosti *přijetí změny (Accept change)* nebo *Odmítnutí změny (Reject change)* z následně vyvolaného kontextového menu. Použití je vizuálně znázorněno na obrázku B.6.




Obrázek B.6: Kontextové menu změny dokumentu

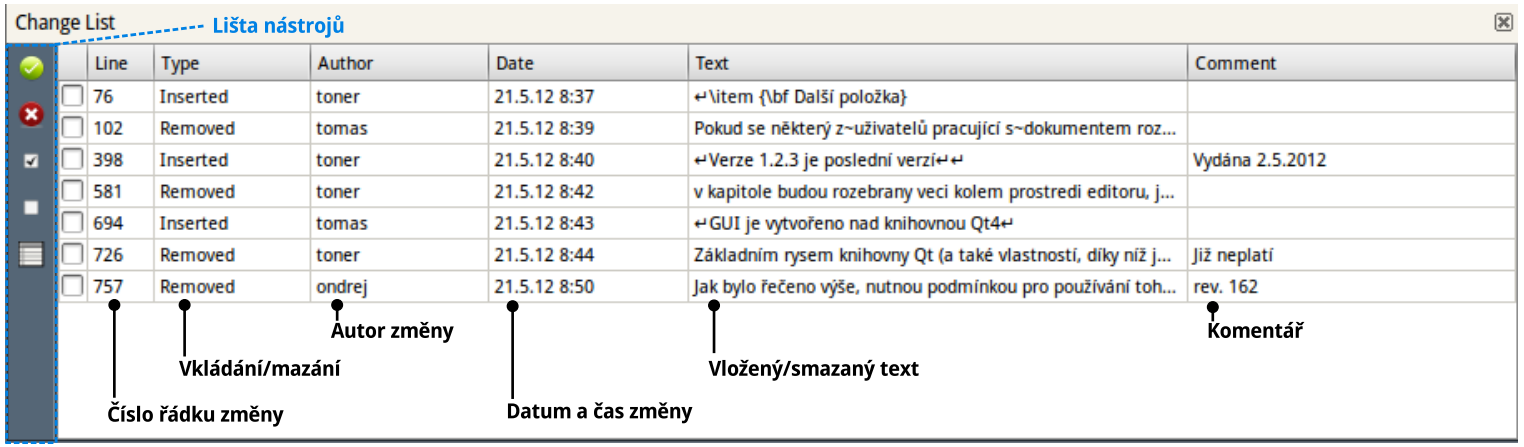
### B.4.2 Seznam změn

Seznam změn lze zobrazit pomocí hlavního menu aplikace (B.1.2.2) nebo nástrojové lišty (B.1.3). Panel se seznamem změn se standardně zobrazí ve spodní části hlavního okna aplikace.

Tento seznam zobrazuje detailní informace o zaznamenaných změnách ve formě tabulky, kde je každý záznam reprezentován jedním řádkem této tabulky. Řazení je prováděno vzestupně podle pozice změny. Tabulka zobrazuje přehledně tyto sloupce:

`[číslo řádku], [typ], [autor], [datum], [vložený/smazaný text], [komentář]`

Na obrázku B.7 je zachycen snímek obrazovky seznamu změn s ukázkovými daty včetně popisů jednotlivých sloupců.



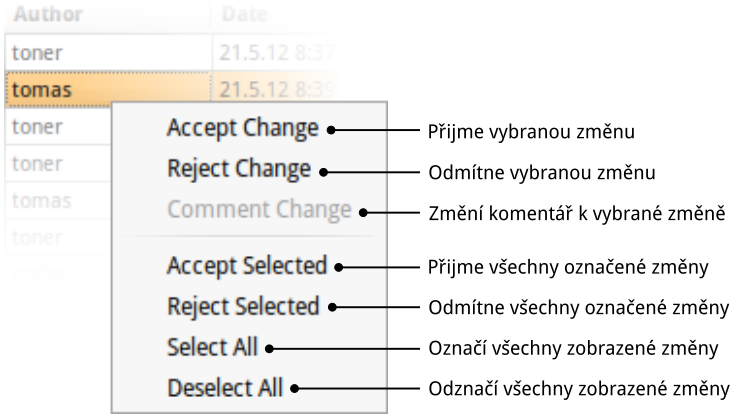

Obrázek B.7: Pohled na panel seznamu změn

### B.4.2.1 Revize v seznamu změn

Revidovat lze v seznamu změn jak po jednotlivých položkách, tak také hromadně.

Pro využití hromadných operací slouží označování/odznačování jednotlivých změn použitím zaškrtávacího tlačítka na začátku řádku, dále pak tlačítka na nástrojové liště *Vybrat vše* a *Vybrat nic*, podrobněji popsána v sekci B.4.2.2

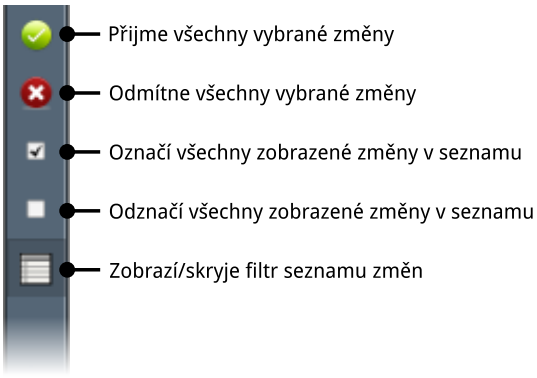
Pro práci s jednotlivými změnami je vhodné použití kontextového menu, vyvolaného stiskem pravého tlačítka nad příslušnou změnou. Nabídka menu pak umožňuje uživateli (podobně jako v sekci B.4.1) změnu přijmout, odmítnout nebo okomentovat (pokud je jejím autorem). Dále kontextové menu nabízí hromadné operace (*Přijmout vybrané změny*, *Odmítnout vybrané změny*, *Vybrat vše*, *Vybrat nic*), jež jsou popsány podrobněji v sekci B.4.2.2. Obsah kontextového menu je ilustrován na obrázku B.8.




Obrázek B.8: Kontextové menu změny dokumentu

### B.4.2.2 Lišta nástrojů seznamu změn

Hromadné funkce pro práci nad seznamem změn se nacházejí na nástrojové liště, umístěné na levé straně panelu seznamu změn. Obrázek B.9 popisuje význam jednotlivých nástrojů.

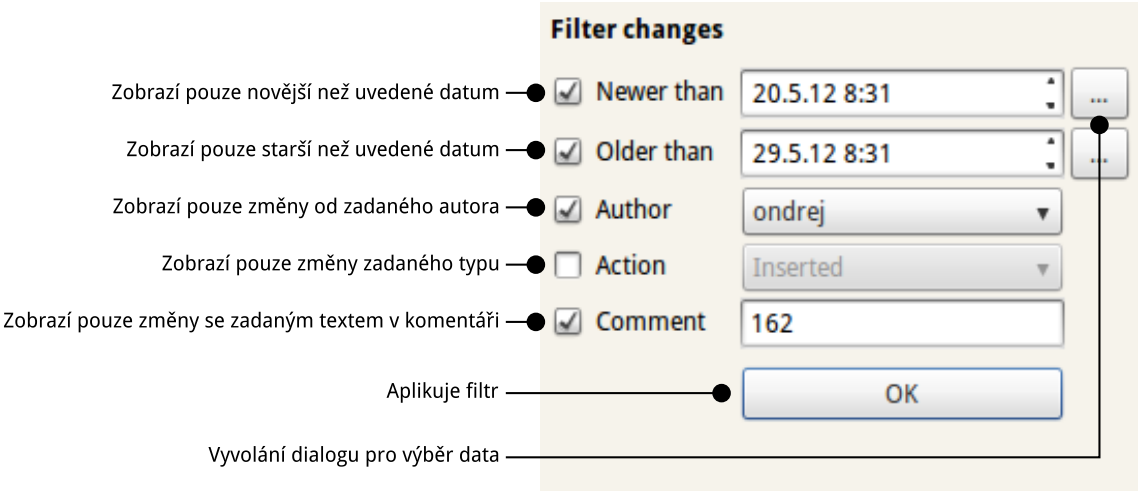



Obrázek B.9: Popis funkcí nástrojové lišty seznamu změn

### B.4.2.3 Filtr seznamu změn

Pro snadnější orientaci v seznamu změn a možnost výběru hledaných změn je přítomen *Filtr změn*. Lze jej zobrazit pomocí určeného tlačítka na nástrojové liště seznamu změn (B.4.2.2).

Filtr změn umožňuje vyhledání a zobrazení změn odpovídajících zadaným kritériím. Filtrovat seznam lze s použitím kritérií *Novější než*, *Starší než*, *Autor*, *Typ změny* a *Text komentáře*. Následující obrázek B.10 vysvětluje možnosti filtrování změn.




Obrázek B.10: Filtr změn s popisem jeho možností