VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Sestavovací systém pro LaTeX

Build system for LaTeX

2014 Michal Tománek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Tománek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: LaTeX build systém
LaTeX Build System

Zásady pro vypracování:

Cílem této práce je návrh a implementace sestavovacícho systému (build system) pro kompilaci rozsáhlých typografických projektů sázených pomocí systému LaTeX. Jde například o knihy, skripta, sadu prezentací k celému předmětu atd. V těchto případech není obvykle dostačující překlad ze specializovaného editoru, či prosté spuštění překladače z příkazové řádky. Zde by bylo vhodnejší využít nástroj typu Ant či make. Cílem práce je vytvořit nástroj tohoto typu určený speciálně pro LaTeX.

Zásady pro vypracování:
1. Průzkum a kritické zhodnocení aktuálně dostupných sestavovacích systémů.
2. Objektový návrh a implementace vlastního systému. Tento systém bude umožňovat překlad dokumentů v LaTeXu, volání další nástrojů (typicky makeindex, bibtex), úklid souborů vzniklých při kompilaci, komunikace s SVN repository.
3. Programátorská a uživatelské dokumentace.

Předpokládá se popis sestavení pomocí XML, implementace sestavovacího systému v C# a provoz na platformě .NET v OS Windows 7 a vyšší.

Seznam doporučené odborné literatury:

[1] Helmut Kopka: Patrick W. Daly: LaTeX kompletní průvodce, Computer Press, Brno 2004, ISBN 80-722-6973-9
[2] Archiv CTAN, www.ctan.org

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla. Velké díky patří hlavně doc. Mgr. Jiřímu Dvorskému, Ph.D, který jakožto vedoucí bakalářské práce mi ochotně pomohl se vším co jsem potřeboval.

Abstrakt

Cílem této práce je analyzovat současné sestavovací systémy pro LaTeX a s výsledky této analýzy následně vytvořit svůj vlastní sestavovací systém. Avšak v plánu není nic převratného.

Klíčová slova: sestavovací systémy, LaTeX, bakalářská práce

Abstract

Goal of this bachelor thesis is to analyze current build systems for LaTeX. After analyzing the build systems I would like to make my own build system with the information I obtained. However we do not plan something revolutionary.

Keywords: build systems, LaTeX, bachelor thesis

Seznam použitých zkratek a symbolů

IDE	–	Integrated Development Environment
SVN	–	Apache Subversion
GCC	–	GNU Compiler Collection
GUI	–	Graphical User Interface
WYSIWYG	–	What you see is (all) what you get

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato Bakalářská práce bude zejména pojednávat o využití sestavovacích systémech. V této práci se budu zaměřovat především na sestavovací systémy LaTeXových souborů. Překladače mají obecně za úkol překládat dokumenty (zdrojové kódy) napsané v nějakém vyšším programovacím jazyce do formy nižšího programovacího jazyka, kterému rozumí počítač tak, aby byl schopný zkompilovat daný zdrojový kód na strojové instrukce pro procesor a vytvořit spustitelný program.

Aby jsme porozuměli co jsou vlastně překladače, tak bych se vrátil k úplným začátkům historie počítačů a jejich programování. V roce 1946 byl dokončen první počítač na světě jménem Eniac. Výpočty a programování se provádělo ručně pomocí přepínačů, po roce 1948 se začaly používat děrné štítky. Takovéto „programování“ trvalo mnoho hodin až několik týdnů.

V 50. letech 20. století se objevil pokročilejší způsob programování a to jazykem symbolických adres. Je to nízkoúrovňový programovací jazyk, který už obsahuje vlastní překladač, který překládá jednoduché instrukce do strojového kódu pro procesor. Z tohoto vyplývá, že obecně překladače překládají zdrojové kódy, obsahující instrukce, z jednoho programovacího jazyka do druhého programovacího jazyka. Příkladem jazyka symbolických adres je Assembler.

Postupem času se projekty začaly zvětšovat a to vedlo ke komplikování programování pomocí jazyku symbolický adres a díky tomuto začaly vznikat vyšší programovací jazyky. Příkladem jednoho z prvních vyšších programovacích jazyků je Fortran. S příchodem mnoha programovacích jazyků přišla také otázka, jak se budou takovéto jazyky překládat? Jedna z možností je GCC překladač, který je součásti projektu GNU (viz. 2.4.2). Tento překladač je vytvořen s podporou více jazyků například Java, C, C++. Z nárůstem náročnosti projektů začaly vznikat vývojové prostředí (IDE) se zaměřením na určitý programovací jazyk. V této době jsou vývojové prostředí tvořena s podporou více jazyků. Příkladem IDE jsou Visual Studio od firmy Microsoft, NetBeans, atp. V případě velkých projektů, na kterých pracuje více lidí se využívá systému SVN, což je systém pro správu a verzování [1] zdrojových kódů.

V 80. letech 20. století Leslie Lamport [4] napsal LaTeX. LaTeX [2] je typografickým systémem, který je určen k sazbě vědeckých a matematických dokumentů vysoké typografické kvality. Systém je rovněž vhodný pro tvorbu všech možných druhů jiných dokumentů, od jednoduchých dopisů po složité knihy. Systém LaTeX je postaven na typografickém formátovacím programu TeX [1] Donalda E. Knutha. LaTeX lze užívat na širokém spektru platforem od IBM PC výše.

Cílem této práce je vytvořit sestavovací systém pro LaTeX, který by měl umět základní operace se soubory a adresáři a především si poradit s překladem větších LaTeXových projektů.

[1] Verzování je způsob uchovávání historie veškerých provedených změn obecně u jakékoliv digitální informace.

1.1 Struktura práce

V kapitole 2 se nachází stručný průzkum současných automatizovaných sestavovacích systémů. V kapitole 3 obsahuje stručný popis TEXu a LATEXu. O současných sestavovacích systémech pro LATEX si povíme v kapitole 4. V kapitole 5 si povíme o mém sestavovacím systému **TBS**. Na implementaci se zaměříme v kapitole 6. Použití TBS (uživatelská dokumentace) je náplní kapitoly 7.

2 Automatizované sestavení

Zpočátku programátoři používali k sestavení programů skripty, které se spouštěly pomocí příkazového řádku. Postupem času, když programy se stávaly rozsáhlejší a kompilování bylo složitější, vymyslel se způsob jak kompilovat programy za pomocí skriptů bez potřeby spouštět jeden skript za druhým. Vyvinula se proto vývojová prostředí, které využívají skripty pro překlad zdrojových kódů. To odstartovalo začátky automatizovaného sestavení [11], což je činnost skriptovací nebo-li automatizování různých úloh, které programátoři vykonávají v jejich denních aktivitách zahrnující věci jako:

- Překládání počítačového zdrojového kódu do binárního kódu.
- Testování.
- Nasazení kódu.
- Vytvoření dokumentace a/nebo poznámek k danému programu.

2.0.0.1 Typy:

- **Automatizace na požádání** – uživatel sám spouští skript v příkazové řádce.
- **Naplánovaná automatizace** – průběžné integrace serveru, které probíhají automaticky v daný čas (např. v noci).
- **Událostmi řízené automatizace** – probíhá za daných podmínek jako například při změně verze systému.

2.1 Make

Jedna z specifických forem automatizace sestavení je Make [12]. Make je utilita, která automaticky sestavuje spustitelné programy a knihovny ze zdrojového kódu prostřednictvím čtení souborů nazývané „makefile“, které určují jak sestavit daný program. I když IDE (vývojová prostředí) a překladače mají své funkce, které mohou být použity jako sestavovací proces, Make stále zůstává používán, hlavně v Unix-u.

2.1.0.2 Nástroje založené na bázi Make:

- Distcc (viz. 2.2.1).
- GNU make – hodně používaný nástroj s velkým výběrem doplňků.
- Make – klasický Unix nástroj (sestavovací nástroj).
- Mk – původně vytvořeno pro verzi 10 Unix a Plan 9 a následně portováno do Unix jako součást Plan 9 Port.

- MPW Make – vytvořeno pro Mac OS Classic, podobně jak pro, ale nekompatibilní, s Unix make. OS X přichází jak s GNU make tak i s BSD make. Zdarma dostupné jako součást semináře Macintosh.
- Nmake, Qmake – upravená verze Make.
- Rake – nástroj založený na programovacím jazyce Ruby.

2.1.0.3 Nástroje nezaložené na bázi Make:

- Apache Ant – populární pro Java platformu. Používá strukturovaný XML formát.
- Apache Buildr – systém, který má otevřený zdrojový kód, založený na jazyce Rake, který dává plné využití skriptování v jazyce Ruby s mnoha chtěnými integrovanými funkcemi co může sestavovací systém mít.
- Apache Maven – Java nástroj pro spravování projektů a automatizovaný sestavovací software
- A-A-P – založený na jazyce Python.
- Flowtracer – je komerční sestavovací management nástroj.
- Gradle – automatizovaný systém s otevřeným zdrojovým kódem s Groovy Rake jako hlavním specifickým jazykem (DSL), který kombinuje výhody jazyků Ant a Maven s mnoha inovativními vlastnostmi jako spolehlivým inkrementovaným sestavováním.
- MSBuild – sestavovací nástroj od Microsoftu.
- NAnt – podobný nástroj jako Ant pro .NET Framework.
- Perforce Jam – obecně pokročilý nástroj, který je podobný jako Make.
- SBT – sestavovací nástroj založený na Scala-based DSL.
- Psake – specifický jazyk a automatizovaný sestavovací nástroj napsán v PowerShell.
- Tweaker – povoluje definice úkolů být napsány v jakémkoliv jazyku (nebo v několika různých jazycích), zatímco poskytující stejné rozhraní pro všechny jazyky.
- Visual Build – software s grafickým uživatelským rozhraním pro softwarové sestavování.
- Scons, CMake (viz. 2.4.1), nebo Ant (viz. 2.3.1).

```
# Generic GNUMakefile

    # Just a snippet to stop executing under other make(1) commands
    # that won't understand these lines
    ifneq (,)
    This makefile requires GNU Make.
    endif

    PROGRAM = foo
    C_FILES := $(wildcard *.c)
    OBJS := $(patsubst %.c, %.o, $(C_FILES))
    CC = cc
    CFLAGS = -Wall -pedantic
    LDFLAGS =

    all: $(PROGRAM)

    .
    .
    .

    $clean:
rm -f .depend *.o

    .PHONY: clean depend
```

Výpis 1: Ukázka kódu jak může vypadat Makefile.

2.1.0.4 Generační nástroje sestavovacích skriptů

- Automake – programovací nástroj, který vytváří makefile soubory.
- CMake – multi-platformový nástroj, který generuje soubory pro přirozené sestavovací prostředí jako makefile pro Unix, nebo Workspace soubory pro Visual Studio.
- GNU sestavovací nástroje (např. autotools) – kolekce nástrojů pro přenosné sestavovací systémy. Ty konkrétně obsahují Autoconf a Automake, multi-platformové nástroje, které dohromady generují odpovídající lokalizované makefile soubory.
- Generate Your Projects (GYP) – vytvořeno pro Chromium[2], což je další nástroj pro generaci souborů pro přirozené sestavovací prostředí.
- Imake – je automatizovaný sestavovací systém pro X Window systém.
- Premake – nástroj založený na jazyku Lua pro vytváření makefile souborů, Visual Studio souborů, Xcode projektů, atp.

[2]Chromium je prohlížeč od společnosti Google.

2.2 Nástroje založené na Make

2.2.1 Distcc

V softwarovém vývoji, distcc [21, 13] je nástroj pro zrychlení kompilace zdrojového kódu. Pro zrychlení využívá rozdělené výpočty místo počítačové sítě. Se správnou konfigurací, distcc může dramaticky snížit kompilační čas projektů. Je navržený tak aby pracoval s programovacím jazykem C (i s C++ a Objective-C) za použití GCC jako jeho „backend“ [3], navzdory tomu, že poskytuje různé verze kompatibility s C++ a Sun Studio Suite překladačem. Distribuováno jako GNU - General Public License. Distcc je k dispozici zdarma.

2.2.2 Mk (software)

Mk [14] je nástroj, který nahrazuje Make ve verzi 9 Unix, Plan 9 z Bell Labs a Inferno. Mk je vylepšený oproti svému předchůdci. Představuje kompletně novou syntaxi, která je mnohem snadnější v orientaci a zároveň je výkonnější. Mk je naportován do Unix jako součást Plan 9. Mk je licencováno jako MIT Licence a Lucent Public Licence, které zaručují, že je software zdarma.

- Autor: Andrew G. Hume
- Operační systém: Verze 9 Unix, Plan 9 od Bell Labs a Inferno
- Typ: Automatizační sestavovací nástroj
- Licence: MIT Licence / Lucent Public Licence

2.3 Nástroje nezaložené na Make

2.3.1 Apache Ant (Another Neat Tool)

Apache Ant [15] je softwarový nástroj pro automatizování softwarových sestavovacích procesech. Je to velmi podobné nástroji Make, ale Apache Ant je implementován za pomocí jazyka Java a vyžaduje Java platformu a nejlepší využití je pro sestavování projektu psaných v Java jazyce. Největší rozdíl, kterého si můžeme povšimnout mezi Ant a Make je, že Ant využívá XML jazyka pro popis procesu sestavení, avšak Make využívá Makefile formát. Defaultně je XML soubor nazýván *build.xml*. Ant je Apache projekt. Je opensource software a spadá pod Apache Software Licenci.

- Vývoj: Apache Software Foundation
- Stabilní verze: 1.9.2 / 12. Července, 2013
- Programovací jazyk: Java

[3] Součást softwaru, se kterým uživatel nepracuje.

- Operační systém: multi-platformový
- Typ: Sestavovací nástroj
- Licence: Apache License 2.0
- Webová stránka: ant.apache.org

```
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="remove␣intermediate␣files">
        <delete dir="classes"/>
    </target>
    <target name="clobber" depends="clean" description="remove␣all␣artifact␣files">
        <delete file="hello.jar"/>
    </target>
    <target name="compile" description="compile␣the␣Java␣source␣code␣to␣class␣files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create␣a␣Jar␣file␣for␣the␣application
        ">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Výpis 2: Ukázkový kód Apache Ant, který narozdíl od Makefile využívá ke kompilaci jazyk XML.

2.3.2 NAnt

NAnt [16] je zdarma a open-source softwarový nástroj pro automatizování softwarových sestavovacích procesech. Je to velmi podobné nástroji Make, ale je směřován spíše na .NET prostředí než na Java platformu. Název pochází z faktu, že NAnt (Not Ant) není Ant. NAnt vyžaduje alespoň jednu z .NET frameworků (1.0, 1.1, 2.0, 3.5, 4.0 nebo vyšší) nebo software z třetí strany Mono platformu.

- Stabilní verze: 0.92 / 9. Června, 2012
- Napsán v: C#
- Operační systém: multi-platformový
- Typ: Automatizační sestavovací nástroj
- Licence: GPL

- Webová stránka: nant.sourceforge.net

2.3.3 Apache Buildr

Apache Buildr [17] je sestavovací systém ve formě otevřeného kódu a jeho hlavním účelem je sestavovat Java aplikace, ale zároveň je schopný mnohem více. Dává vývojáři sílu plnohodnotného skriptovacího jazyka (Ruby) zatímco psaní jeho nebo její sestavovací skripty, mnoho uniklo v sestavovacích prostředích založených na XML jako Apache Ant, nebo Apache Maven. Apache Buildr je založen na sestavovacím systému Rake, který využívá Ruby jako skriptovací jazyk. Používá některé projektové automatizační idiomy z Maven. I přesto, že princip Buildr-u je podobná jako u Maven-u, využívá schopnosti skriptovacího jazyka Ruby k poskytnutí mnohem větší flexibility a přizpůsobeného programového modulu, který dramaticky rozšiřuje možnosti vývojářů. Uživatel píše sestavovací skript v Ruby, deklaruje projekt, jeho pod-projekty, artefakty, závislosti, pravidla balíčků, atd. S ohledem na tyto deklarace je vytvořen seznam standardů, které jsou předány pro kompilaci, nebo zabalení projektu, nahrání připraveného balíčku na vzdálené úložiště, pročistit sestavený projekt, odzkoušet, atd. Apache Buildr je hlavně navržen pro Javu, ale taky umí využívat kompilery pro Groovy, Scala a díky jeho snadné rozšiřitelnosti lze využít i kompilerů třetích stran a díky tomu ho lze nazvat jako univerzální sestavovací systém.

- Vývoj: Apache Software Foundation
- Stabilní verze: 1.4.12 / May 4, 2013
- Napsán v: Ruby
- Operační systém: multi-platformový
- Licence: Apache License 2.0 v
- Webová stránka: buildr.apache.org

2.3.4 A-A-P

A-A-P [18] je počítačový program využívaný hlavně ke stahování, sestavování a instalování softwaru. Spouští tzv. „recipes“ (předpisy), které jsou podobné jako Makefile soubory, akorát mají významně mnohem více možností funkcionality. Má integrovanou podporou pro nahrávání a stahování souborů, stranou od automatického stahování distribuce. Může být také použit pro údržbu webových stránek.

A-A-P je napsán a může být skriptován v jazyce Python. Agide, A-A-P GUI IDE, je modulární vývojový framework. Editování, sestavování a ladění s různými kombinacemi programů můžou být teoreticky zkombinovány, ale jen s Vim (Vi clone) editorem, A-A-P-ovým instrukčním předpisem sestavení a GDB debugger jsou momentálně podporovány.

- Vývoj: Bram Moolenaar

- Stabilní verze: 1.092 / 14. Prosince, 2012
- Napsán v: Python
- Operační systém: Windows, POSIX
- Typ: Softwarový vývojový nástroj
- Licence: GNU GPL
- Webová stránka: www.a-a-p.org

2.4 Generační nástroje sestavovacích skriptů

2.4.1 CMake

CMake [19] je multi-platformový software program pro spravování sestavovacích procesů využivající kompiler s „nezávislou“ metodou. Je navržený k podpoře hierarchických adresářů a aplikací, které závisí na více knihovnách a pro použití ve spojení se sestavovacími prostředími jako asmake, Xcode od Applu a Microsoft Visual Studio. Má taky minimální závislosti, vyžadující jen C++ kompilátor pro jeho vlastní sestavovací systém.

- Vývoj: Andy Cedilnik, Bill Hoffman, Brad King, Ken Martin, Alexander Neundorf
- Stabilní verze: 2.8.11 / May 22, 2013
- Napsán v: C++
- Operační systém: multi-platformový
- Typ: Softwarový vývojový nástroj
- Licence: Nová BSD Licence
- Webová stránka: www.cmake.org

2.4.2 GNU sestavovací systém

GNU [20] sestavovací systém, také známý jako Autotools je souprava programovacích nástrojů navržených pro asistování při tvorbě přenosných balíčků zdrojových kódů pro mnoho systémů na bázi Unix. Tvorba přenosného softwarového programu může být složitá: C kompilátor se liší v každém systému. Díky široké různosti sestavovacích prostředí se stává toto těžko ovladatelným. Autotools je navrhnut tak, aby tento problém byl více ovladatelným. Autotools je části GNU „toolchain“ a je obecně používán v mnoha softwarech, které jsou k dispozici zdarma a balíčcích s otevřeným kódem.

2.5 Přehled nástrojů pro automatizované sestavení

Kompletní přehled nástrojů pro automatizované sestavení je uveden v tabulce 1.

Název	Operační systém	Jazyk automat. s.	Podporované jazyky	Licence
A-A-P	Unix, Windows, OS X	C	Předpisy	GNU GPL
Ant	Multi-platformní	Java	XML	Apache 2.0
AnthillPro	Multi-platformní	Podpora více jazyků	Wraps Make, Ant, Maven, MsBuild, Nant, atd.	Proprietární
Automated Build Studio	Windows	Podpora více jazyků	Grafické IDE s podporou Ant/NAnt, MSBuild, JScript, VBScript, DelphiScript	Proprietární
Bamboo	Multi-platformní	Podpora více jazyků	Průběžná integrace	Proprietární
Buildr	Multi-platformní	Java	Ruby	Apache 2.0
Capistrano	Multi-platformní	Ruby	Doménový specifický jazyk pro softwarové sest. systémy	MIT
CMake	Multi-platformní	Podpora více jazyků	Používá CMakeLists.txt soubor	Nová BSD
Continuum	Multi-platformní	Java	Neznámá	Apache 2.0
CruiseControl	Multi-platformní	Podpora více jazyků	XML	BSD
dmake	Linux, Solaris, Windows	OpenOffice, Solaris	Používá Makefile	Neznámá
Electric Commander	Multi-platformní	Podpora více jazyků	gmake, bmake, emake and ant scripts	Proprietární
FinalBuilder	Windows	Podpora více jazyků	Grafické IDE s podporou Ant/NAnt, MSBuild, JScript, VBScript, IronPython, PowerShell	Proprietární
Gradle	Multi-platformní	Java-Groovy	Groovy based DSL	Apache 2.0
Jenkins	Multi-platformní	Podpora více jazyků	Průběžná integrace	MIT
make	Multi-platformní	Podpora více jazyků	Používá Makefile	Různá
Maven	Multi-platformní	Java	Projekt Objektový model	Apache 2.0
MSBuild	Windows	Podpora více jazyků	XML	Proprietární (zdarma)
NAnt	Multi-platformní	.NET	XML	GNU GPL
nmake	Unix, Linux, Windows/Interix	Podpora více jazyků	Používá Makefile	Neznámá
SBuild	Multi-platformní	Scala, Java	Scala založená na DSL	Apache 2.0
Simple Build Tool	Multi-platformní	Scala, Java	Scala založená na DSL	BSD
Tweaker	Multi-platformní	Podpora více jazyků	Jakýkoliv (Ant, bash, batch, C, Erlang, Java, SQL, a VBScript jako default, lze využít rozšíření třetích stran)	GPLv3
Visual Build	Window	Podpora více jazyků	XML	Proprietární

Tabulka 1: Přehled nástrojů pro automatizované sestavení

3 TEX a LATEX

3.1 TEX

TEX [2] je počítačový program vytvořený profesorem Donaldem E. Knuthem [1]. Je určen pro sazbu textu a matematických rovnic, při zachování vysoké typografické úrovně výsledného dokumentu. TEX vyslovujeme „tech“; „X“ pochází z řeckého χ, proto jej vyslovujeme jako „ch“. Logo TEX přepisujeme do ASCII jako TeX.

3.2 LATEX

LATEX [2] je balík maker, který umožňuje autorům sázet a tisknout jejich díla v nejvyšší možné typografické kvalitě, přičemž autor používá profesionály předdefinovaných vzhledů dokumentů. LATEX byl původně napsán Leslie Lamportem [1]. LATEX užívá programu TEX jako sázecího stroje.

V současné době je LATEX rozšiřován týmem LATEX3, vedeným Frankem Mittelbachem [10]. Tento tým se snaží sjednotit všechny rozšiřující verze LATEXu, které postupně vznikaly od vzniku LATEXu 2.09. Aby byla nějakým způsobem odlišena stará verze od nové, tak byla označena LATEX 2_ε.

Slovo LATEX se vyslovuje „lej-tech“ nebo „la-tech“. Při odkazech na LATEX v prostředí ASCII se LATEX přepisuje jako LaTeX. LATEX 2_εse vyslovuje „lejtech tů í“ a zapisuje se jako LATEX 2_ε.

Při psaní dokumentu pomocí LATEXu je rozdíl zdali člověk píše jen jeden článek či celou knihu. Při psaní jednoho článku stačí založit jeden jediný soubor a začít psát. Literaturu k takovému článku jednoduše můžeme udělat ručně, nebo lze využít balíku BibTeX [5].

Jestliže se rozhodnete s psaním jakékoliv knihy, tak se většinou počítá s projektem, který obsahuje stovky až tisíce stran a to nemluvím ani o možné ilustraci, která se může vyskytovat v dané knize. V takovém případě se nejedná jen o jeden soubor, ale několik souborů, kde jeden z nich je hlavní a podle něho se pak řídí překladač při překladu. Pro tyto případy se využívá vývojových prostředí jako je například TeXnicCenter [3], který je vytvořený přímo pro psaní dokumentů pomocí LATEXu.

U větších projektů jako je psaní knih, se pro literaturu využívá balíku BibTeX, který nám usnadní a pomůže s formátováním odkazů na danou literaturu. Velmi užitečnou součástí knih je rejstřík. Pro generování rejstříku (indexu) lze taky použít balík, který je součástí LATEXu, jménem „makeindex“[4]. Díky těmto příkazům, které nám dovoluje LATEX použít, můžeme jednoduše tyto součásti velkých projektů vytvářet bez jakýchkoliv problémů. Také se můžeme setkat v projektech, s IT zaměřením, se zdrojovými kódy (např. C++), nebo grafy. Pro grafy a ilustrace lze použít například balík tikz [6], nebo program gnuplot [7] pracující pomocí skriptů, který dokáže spolupracovat s LATEX systémem.

[4] V systémech, které neumožňují mít jména souborů delší jak osm znaků se program jmenuje *makeidx*

Při kompilaci LaTeXového projektu vznikají „pomocné“ soubory, které většina uživatelů nepotřebují a v některých IDE a snad ve všech sestavovacích systémech lze nastavit, že po kompilaci se smažou automaticky.

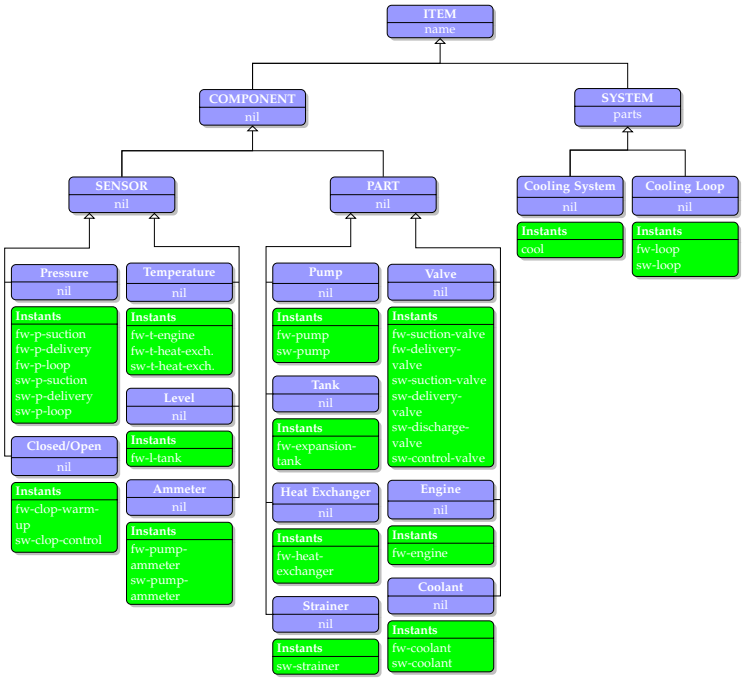
Práce na takto rozsáhlých projektech si většinou vyžaduje práce více lidí a pro lehčí domluvu a kolektivní práci jsou pro tuto činnost přizpůsobeny i programy jako je například SVN [8]. Tento verzovací program je zdarma a může být použit i komerčně. SVN je systém pro správu a verzování zdrojových kódů, náhrada za starší CVS [9]. Snaží se zachovat podobný způsob a styl práce, ale odstranit nedostatky CVS jako například nemožnost přesunu nebo kopírování adresářů, časová a prostorová náročnost větvení a tagování a podobně. Jednou z výhod systému Subversion je existence velmi dobré dokumentace (zatím v angličtině) – nazývá se Version Control with Subversion a je volně dostupná. Další je existence více přístupových metod k repozitáři. Subversion je, tak jako CVS, založeno na principu centrálního repozitáře.

```
\documentclass{article}
\ title {Ukazka}
\author{Michal Tomanek}
\date{Duben 2014}
\begin{document}
    \maketitle
    Hello world!
\end{document}
```

Výpis 3: Ukázka LaTeXového kódu pro vypsání jednoduchého „Hello World!“.

```
\begin{document}
\ tikzstyle {abstract}=[rectangle, draw=black, rounded corners,
        text centered, anchor=north, text=white, text width=3cm]
\ tikzstyle {comment}=[rectangle, draw=black, rounded corners,
        text centered, anchor=north, text=white, text width=3cm]
\ tikzstyle {myarrow}=[->, >=open triangle 90, thick]
\ tikzstyle {line}=[-, thick]
\begin{center}
\begin{tikzpicture}[node distance=2cm]
    \node (Item) [abstract, rectangle split , rectangle split parts=2]
        {
            \textbf{ITEM}
            \nodepart{second}name
        };
    \node (AuxNode01) [text width=4cm, below=of Item] {};
    .
    .
    .
    \draw[myarrow] (CoolingSystem.north) -- ++(0,0.8) -| (System.south);
    \draw[line] (CoolingSystem.north) -- ++(0,0.8) -| (CoolingLoop.north);
\end{tikzpicture}
\end{center}
\end{document}
```

Výpis 4: Ukázka LaTeXového kódu pro vykreslení třídního diagramu pomocí balíku *tikz*.



Obrázek 1: Grafická podoba ukázkového kódu pro grafické zobrazení třídního diagramu.

3.3 Výhody a nevýhody

Častým tématem vášnivých diskusí milovníků WYSIWYG systému a milovníků LaTeXu jsou výhody případně nevýhody LaTeXu oproti normálním textovým procesorům. Nejlepší věc, kterou lze udělat, když taková diskuse začíná, je držet se zpět, jelikož se taková diskuse velice často vymkne člověku z rukou. Avšak jsou chvíle, kdy se nedá nikam utéct...

Oproti ostatním textovým procesorům se LaTeX vyznačuje následujícími výhodami [2]:

- Je k dispozici několik profesionálně vytvořených formátů, se kterými dokumenty vypadají „jako profesionálně vytištěné“.
- Zvlášť dobře je podporována sazba matematických vzorců.
- Uživatel musí zadávat jen několik lehce srozumitelných příkazů, které se týkají logické struktury dokumentu, a (téměř) nepotřebuje zabývat se technickými detaily tisku.
- Bez velké námahy mohou být vytvořeny také složité struktury jako poznámky pod čarou, seznamy literatury, obsahy, tabulky atd. stejně jako odkazy na stránku, čísla kapitol tabulek, obrázků, rovnic atd.
- Pro mnoho typografických funkcí, které nejsou přímo podporovány základním LaTeXem existují snadno připojitelné nadstavby, tzv. balíky maker. Například lze po-

užívat balík umožňující práci z grafickým formátem ve formě PostScriptu či balík pro sazbu bibliografie splňující přesně daný standard. Většina podobných balíků je popsána v *The LaTeXCompanion* [10].

- TeX, základní program pro práci LaTeXu a navíc TeX i LaTeX jsou i přes své vysoké kvality zdarma.

LaTeX má i své nevýhody:

- Dokonalejší algoritmy pro formátování kladou větší nároky na čas zpracování a paměť než u jednodušších programů pro zpracování textu. Kompletní instalace zabírá poměrně mnoho diskového prostoru. Vzhledem k tomu, že v současné době se objevují textové procesory užívající více i méně diskové kapacity než běžný LaTeX, není tento zápor nutné brát v úvahu. V tomto případě je nutno brát v potaz, že LaTeX se stará o veškerou typografii.
- Výstup textu je možný pouze na grafických zařízeních (laserové, inkoustové nebo jehličkové tiskárny, grafické obrazovky), nikoliv na levných znakově orientovaných rychlotiskárnách.
- Ačkoli lze v předem připravených dokumentárních stylech snadno přizpůsobit jednotlivé parametry, zásadní změny předem stanovených formátů vyžadují větší námahu (návrh nového dokumentárního stylu).[5]

[5] Říká se, že právě tento problém má vyřešit vznikající systém LaTeX3

4 Sestavovací systémy pro LaTeX

4.1 Latexmk

Latexmk [22] je jeden z nejpoužívanějších LaTeXových sestavovacích systému. Latexmk zcela automatizuje proces překladu LaTeXových dokumentů. K zjištění, které soubory má přeložit, používá log soubor, který analyzuje a zjistí veškeré informace, které potřebuje. Volitelně může také analyzovat list vstupních a výstupních souborů, které jsou generovány funkcí „bold-recorder“, kterou obsahuje moderní verze latexu (pdflatex, xelatex, lualatex, atd.). Další možností jsou tzv. Standardně je u latexmk výstupní soubor „.dvi“.

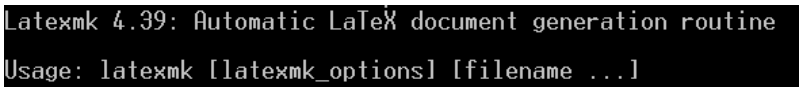
4.2 Arara

Arara [23] je sestavovací systém založený na jazyku Java a XML. Arara nepracuje na stejném principu jako latexmk, tudíž neanalyzuje log soubor aby zjistila co má dělat, ale je spíše založená na analýze instrukcí, které se nacházejí uvnitř TeX souboru. Pro některé uživatelé toto řešení nemusí být zrovna nejlepší, protože musí explicitně říct co má Arara dělat. Například:

- Kolikrát se má pdflatex spustit.
- Co vše se má spustit (bibtex, makeindex), aby se správně uživatelův LaTeXový projekt správně přeložil i se všemi referencemi.
- Styl přeložení, atd...

4.3 Rubber

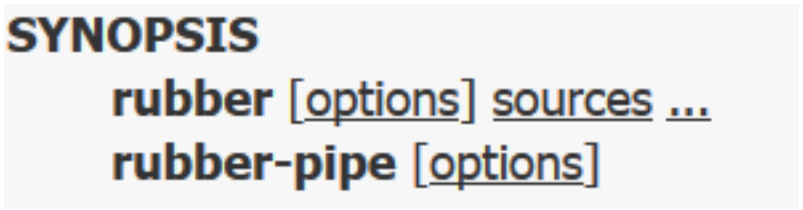
Rubber [24] je další sestavovací systém pro LaTeX. Avšak vstupní soubor pro překlad nemusí být jen s koncovkou „.tex“, ale Rubber podporuje i formáty CWEB, nebo Literate Haskell dokumenty. Pokuď narazí během kompilace na nějaký „error“, kompilace automaticky skončí. Oproti Latexmk a Arara umí zipovat vygenerované soubory.



Obrázek 2: Použití sestavovacího systému latexmk.

```
1 % arara: pdflatex: { shell: yes }
2 % arara: makeindex: { style: mystyle }
3 % arara: pdflatex: { shell: yes }
4 % arara: pdflatex: { shell: yes }
5 \documentclass{book}
6 ...
```

Obrázek 3: Použití sestavovacího systému Arara.



Obrázek 4: Použití sestavovacího systému Rubber.

5 TBS - TEXBuild System

Jak jsme si řekli a ukázali, tak sestavovací systémy pro LaTeX existují. Úkolem této práce je na základě existujících sestavovacích systému vytvořit svůj vlastní TeXový sestavovací systém, avšak v plánu není nic převratného. TBS umožňuje překlad jen do pdf formátu oproti existujícím systémům, nicméně má také prvky, které neobsahují existující systémy.

- Vstupní XML skript, který si lze nastavit podle toho jak uživatel potřebuje.
- Obsahuje funkce, díky nimž může uživatel svou složku „zazipovat“ či může jakýkoliv zip soubor rozbalit.
- Uživatel má možnost provádět základní operace se soubory i adresáři.
- Defaultní výstup jako PDF soubor.

5.1 Požadavky

TBS by měl umožňovat především překlad TeXových souborů. Bylo by dobré, kdyby uměl základní operace se soubory a složkami. Myšleno: kopírování, přesouvání, mazání a vytváření složek i souborů. Bylo by dobré kdyby také zvládal kompresi a dekompresi souborů a následně vytvořil Zip soubor. Další součástí by mohla být komunikace se vzdáleným úložištěm pomocí SVN. Jelikož počítač sám od sebe neumí uklidit, TBS bude taktéž schopen umět po sobě uklidit.

5.2 Návrh skriptovacího jazyka

K tomu aby TBS mohl fungovat tak jak si uživatel přeje, musí mít k dispozici skript. Jazyk vstupního skriptu jsem zvolil značkovací jazyk XML, protože je to strukturovaný jazyk (strom) a přehledný. Jak bylo řečeno, XML je značkovací jazyk a skládá se tedy ze značek (tagů). Každý tag může mít za úkol něco jiného a může taky mít své atributy. Kódování jazyka je UTF-8 a je tzv. case sensitive (citlivý na malá/velká písmena).

5.3 XML tagy

XML tagy jsou v tomto projektu velmi důležité a je potřeba aby se všechny psaly malými písmeny, protože XML skript je senzitivní na malá/velká písmena. Nebude-li to dodrženo, program nebude pracovat tak jak má.

5.3.1 Project

Project je hlavní, může být kořenový, element XML skriptu od kterého se všechno bude načítat do TBS a následně vykonávat. Může obsahovat několik svých potomků s názvem „target“, ale hlavně obsahuje dva atributy „default“ a „texname“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény. Příklad lze vidět na obrázku 5, strana 26.

- **Default** - určuje, který target se bude vykonávat jako defaultní.
- **Texname** - udává jaké má jméno TEXový soubor, který se má přeložit. Jméno TEXového souboru může být definováno s koncovkou „.tex“, ale také nemusí. V případě neuvedení koncovky, TBS bude hledat v daném adresáři soubor s koncovkou „.tex“ a nenajde-li takovýto soubor, vypíše v příkazové řádce chybovou hlášku. Pokuď tedy zvolíte „name=„main““, tak pak TBS bude hledat soubor „main.tex“.
- **Potomci** - target.

5.3.2 Target

Target je element, který se může opakovat a má také své potomky. Obsahuje jeden atribut „name“. Nesmí obsahovat více atributů, nebo atribut s jiným jménem. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - určuje jméno daného targetu a je velmi důležité, aby následně TBS mohl zjistit, do kterého „target-u“ se má podívat a začít podle toho pracovat.
- **Potomci** - mkdir, mvdir, rmdir, cpdir, ctfile, mvfile, cpfile, rmfile, zpfile, uzfile, pdflatex, pdfviewer, clean, cleanwork, svncom.

5.3.3 Pdflatex

Pdflatex určuje zdali se má provést překlad TEXových souborů. Může obsahovat jediný atribut „mode“. Nesmí obsahovat více atributů, nebo atribut s jiným jménem a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Mode** - určuje v jakém módu se spustí pdflatex. Např. nonstopmode, errorstopmode, batchmode (silent), scrollmode atp.

  POZOR! Mode může obsahovat jakýkoliv volitelný příkaz, který dovoluje *pdflatex*. Tedy pro funkci *nonstopemode* je nutné napsat příkaz - *„-interaction=nonstopemode“*. Volitelné příkazy „pdflatex-u“ si lze nechat vypsat v přikazové řádce pomocí příkazu „pdflatex -help“.

5.3.4 Mvdir

Mvdir je element pro funkci přesunutí složky. Obsahuje dva atributy „name“ a „mvto“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - určuje, která složka se má přesunout.
- **Mvto** - zkratka pro „move to“, udává kam se má daná složka přesunout.

5.3.5 Cpdir

Cpdir je element pro zkopírování složky. Obsahuje tři atributy „name“, „cpto“ a „subdir“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - jméno kopírované složky.
- **Cpto** - zkratka pro "copy to", udává kam se má daná složka zkopírovat.
- **Subdir** - určuje zdali se mají i podsložky zkopírovat. Atribut může nabývat hodnot *true* (pravda) a *false* (nepravda). Jestliže atribut nebude vyplněn, je defaultně nastavena hodnota *false*.

5.3.6 Rmdir

Rmdir element pro smazání složky. Obsahuje jeden atribut „name“. Nesmí obsahovat více atributů, nebo atribut s jiným jménem a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - jméno složky, která se má smazat.

5.3.7 Mkdir

Mkdir je element pro vytvoření složky. Obsahuje jeden atribut „name“. Nesmí obsahovat více atributů, nebo atribut s jiným jménem a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - jméno složky, která se má vytvořit.

5.3.8 Mvfile

Mvfile je element pro přesunutí souboru(ů). Obsahuje dva atributy „mask“ a „mvto“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Mask** - určuje, co všechno se má přesunout. Lze využití regulárních výrazů „*,?“. Kde „*“ představuje libovolný počet znaků v názvu a „?“ představuje vždy právě jeden libovolný znak. Například:
  - *mask*=„*.txt“ - přesune všechny soubory s koncovkou „.txt“, tedy všechny textové soubory.
  - *mask*=„???.*“ - přesune všechny soubory, které nemají delší jméno jak tři znaky s libovolnou koncovkou.
- **Mvto** - zkratka pro „move to“, udává jméno cílového adresáře.

5.3.9 Cpfile

Cpfile je element pro zkopírování souboru(ů). Obsahuje tři atributy „mask“, „cpto“ a „overwrite“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Mask** - určuje, co všechno se má smazat. Lze využít opět regulárních výrazů.
- **Cpto** - zkratka pro "copy to". Udává kam se má daná složka zkopírovat.
- **Overwrite** - pokuď by se nacházel soubor se stejným jménem v cílovém adresáři, tento atribut rozhodne zdali se má soubor přepsat či nikoliv. Lze nastavit na true/-false (pravda/nepravda). Jestliže atribut nebude vyplněn, je defaultně nastavena hodnota *false*.

5.3.10 Rmfile

Rmfile je element pro smazání souboru(ů). Obsahuje jeden atribut „mask“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Mask** - určuje, co všechno se má smazat. Lze využití opět regulárních výrazů.
- **Subdir** - určuje zdali se má/mají soubor(y) smazat i v podsložkách. Jestliže atribut nebude vyplněn, je defaultně nastavena hodnota *false*.

5.3.11 Ctfile

Ctfile je element pro vytvoření souboru. Obsahuje tři atributy „name“, „content“ a „overwrite“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - jméno souboru, který se má vytvořit.
- **Content** - obsah, který následně bude ve vytvořeném souboru.
- **Overwrite** - pokuď by se nacházel soubor se stejným jménem v cílovém adresáři, tento atribut rozhodne zdali se má soubor přepsat či nikoliv. Lze nastavit na true/-false (pravda/nepravda). Jestliže atribut nebude vyplněn, je defaultně nastavena hodnota *false*.

5.3.12 Zpfile

Zpfile je element pro kompresi souboru(ů). Obsahuje dva atributy „name“ a „subdir“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - název komprimovaného (zip) souboru.

- **Subdir** - určuje zdali se mají „zazipovat“ i podsložky. Lze nastavit na true/false (pravda/nepravda). Jestliže atribut nebude vyplněn, je defaultně nastavena hodnota *false*.

5.3.13 Uzfile

Uzfile je element pro dekompresi souboru(ů). Obsahuje dva atributy „name“ a „whereto“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Name** - určuje, který zip soubor se má „odzipovat“.
- **Whereto** - kam se má rozbalit daný zip soubor.

5.3.14 Clean

Clean je element pro úklid (smazání) pdf souborů. Neobsahuje žádné atributy a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

5.3.15 Cleanwork

Cleanwork je element pro úklid nadefinovaných souboru(ů). Komu by nevyhovovalo takto nadefinovaný seznam souborů, může si vytvořit svůj vlastní v XML skriptu pomocí tagu „rmfile“. Neobsahuje žádné atributy a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26. Jak nadefinovat svůj vlastní seznam, lze vidět na 6, strana 26.

- **Nadefinovaný list souborů** - *.bak, *.aux, *.idx, *.ilg, *.ind, *.log, *.toc, *.01, *.02, *.bbl, *.blg, *.tps, *.txt, *.ind, *.toc, *.lot, *.lol, *.out.

5.3.16 Svncom

Svncom je element pro komunikaci se vzdáleným depozitářem. Pracuje na principu Apache Subversion. Obsahuje tři atributy „command“, „path“ a „exepath“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **Command** - představuje proměnou, podle které se bude vědět co se má vykonávat. Takových to příkazů může být více a lze je najít na webových stránkách [26]. Avšak čtyři jsou nejdůležitější znát a to jsou: checkout, update, commit a add.
- **Path** - představuje cestu k adresáři, se kterým chceme pracovat.
- **Exepath** - cesta k Tortoise exe souboru. Instalační cesta se může lišit podle vašeho uvážení, tudíž si musíte nastavit celou cestu k „tortoiseproc.exe“.

5.3.17 Pdfviewer

Pdfviewer je element pro otevírání a náhled pdf souborů. Obsahuje dva atributy „name“ a „exepath“. Nesmí obsahovat více atributů, nebo atributy s jinými jmény a nemá žádné potomky. Příklad lze vidět na obrázku 5, strana 26.

- **name** - lze si navolit určitý soubor, který se bude otevírat. Lze psát název souboru s koncovkou „.pdf“, ale není to nutné. Jestliže tento atribut nebude použit, tak se defaultně bude otevírat soubor se jménem, který je uveden pro překlad (LaTeX).
- **Exepath** - cesta k jakémukoliv exe souboru, který umí otevírat pdf soubory. Cesta se může lišit podle vašeho uvážení kde ho chcete nainstalovat, tudíž si musíte taky nastavit celou cestu.

```
<?xml version="1.0"?>
<project default="run"  texfile ="main.tex">
    <target name="run">
        <mkdir name="main" />
        <mvdir name="pokus" mvto="zaloha" />
        <rmdir name="zaloha" />
        <cpdir name="" cpto="" subdir="true/false"  />

        <mvfile mask="main.*" mvto="main" />
        <cpfile mask="*.pdf" cpto="zaloha" overwrite="true" />
        <rmfile mask="*.txt" subdir="false"  />
        <ctfile name="log" content="TBS version 1.0.3, Date: 2014/05/01" overwrite="false"/>

        <zpfile name="zip" subdir="true" />
        <uzfile name="zip" whereto="zip" />

        <pdflatex mode="-interaction=batchmode"/>
        <pdfviewer name="main" exepath="C:\Program Files (x86)\Adobe\Reader 11.0\Reader\
            AcroRd32.exe" />

        <svncom command="update" path="C:\Users\Desktop\svn" exepath="C:\Program Files\
            TortoiseSVN\bin\tortoiseproc.exe" />

        <clean />
        <cleanwork />
    </target>
</project>
```

Výpis 5: Ukázka XML skriptu.

```
<target name="mydelete">
        <rmfile mask="main.aux" subdir="false" />
        <rmfile mask="*.log" subdir="true" />
        <rmfile mask="m*.txt" subdir="false" />
        <rmfile mask="m???.lol" subdir="true" />
    </target>
```

Výpis 6: Ukázka vlastního „mazacího“ seznamu.

6 Implementace

Pro vývoj TBS jsem zvolil platformu .NET Framework od Microsoftu a jako vývojové prostředí jsem použil Microsoft Visual Studio 2013 s podporou jazyka C#, ve kterém je TBS naprogramován. Jak jsme si řekli v minulé kapitole, že XML skript obsahuje tagy, tak při návrhu jsem postupoval tak, že každý tag bude představovat jednu třídu a jeho atributy budou představovat vstupní atributy pro dané třídy. TBS tedy funguje jako překladač vstupního XML skriptu a následně vykonává to co se nachází v daném skriptu. Při tomto postupu vzniká spoustu tříd s elementárními funkcemi, kde lze velmi dobře využít návrhového vzoru Kompozit [25]. Návrhový vzor Kompozit představuje řešení, jak uspořádat jednoduché objekty a z nich složené (kompozitní) objekty. Snahou vzoru je, aby k oběma typům objektů (jednoduchým a složeným) bylo možné přistoupit jednotným způsobem. Použitím vzoru Kompozit lze uspořádat jednoduché objekty a kompozitní objekty do stromové hierarchické struktury.

Jak lze vidět na obrázku 5 třídu „Component“, tak v mém projektu takovouto třídu představuje interface Command. Takováto třída či interface obsahují metody, které musí následně implementovat i všechny podtřídy, ať už složené či jednoduché. Dále je možné vidět „Leaf“ (list) a „Composit“ (větev). Tzv. „leaf“ je právě jednoduchý objekt, který obsahuje jednu metodu. Tzv. „Composite“ je složený objekt, který má možnost se dále větvit na další listy, nebo i větve. Důležité je aby šlo ke všem takovým objektům přistupovat jednotným způsobem.

```
namespace TBS
{
    public interface Command
    {
        void Execute();
    }
}
```

Výpis 7: Implementovaný interface pro využití návrhového vzoru Kompozit.

TBS funguje na principu: najdi, znáš?, proveď. Nejprve musí najít v XML skriptu kořenový element *project*, který se načítá v samostatné třídě a následně hledá svého potomka *target*, tedy „složený“ objekt.

```
XmlNode root = doc.DocumentElement;

            if (root.Name == "project")
            {
                ProjectNode p = new ProjectNode(root, xmlTarget);
                p.procProjectNode();
            }
```

Výpis 8: Nalezení kořenového elementu *project*.

Target se také načítá v samostatné třídě a musí se porovnávat i jméno (*name*), aby se vykonal správný *target*.

```
foreach (XmlNode node in _root.ChildNodes)
        {
            if (varBool) { break; }

            if (node.Name == "target" && node.Attributes[0].Value == _xmlTarget)
            {
                TargetNode tn = new TargetNode(node, texFile);
                tn.procTargetNode();
                varBool = true;
            }
        }
```

Výpis 9: Nalezení elementu *target* a porovnávání jeho *name*.

Po té co se najde správný *target*, tak se už hledají i jeho potomci, tedy „jednoduché“ objekty jak můžeme vidět na následujícím kódu. V ukázce můžeme vidět jeden „case“ (element) a to *CPDir*.

```
public void procTargetNode()
    {
        foreach (XmlNode targetNode in _node.ChildNodes)
        {
            switch (targetNode.Name)
            {

                case "cpdir":
                    target.Add(new CPDir(targetNode));
                    break;
                    }
                    target.Execute();
              }
        }
```

Výpis 10: Nalezení elementu *target* a hledání jeho potomků.

V každé třídě, která představuje jeden tag (list) a tvoří tím jednu elementární akci si v konstruktoru načítám daný tag a jeho atributy. Atributy je nutno porovnávat aby se zjistilo zdali souhlasí a jestli tam třeba nedošlo k chybě jak lze vidět na další ukázce kódu.

```
public CPDir(XmlNode targetNode)
    {
        for (int attInd = 0; attInd < targetNode.Attributes.Count; attInd++)
        {
            varName = targetNode.Attributes[attInd].Name;

            if (varName == "name")
            {
                sourceDirName = targetNode.Attributes[attInd].Value;
            }
            else if (varName == "cpto")
            {
                destDirName = targetNode.Attributes[attInd].Value;
            }
```

```
            else if (varName == "subdir")
            {
                copySubDirs = true;
            }
            else
            {
                Console.WriteLine("Wrong␣argument(s)␣in␣cpdir␣element.");
            }
        }
    }
```

Výpis 11: Načítání atributů tagu, v tomto případě *CPDir*.

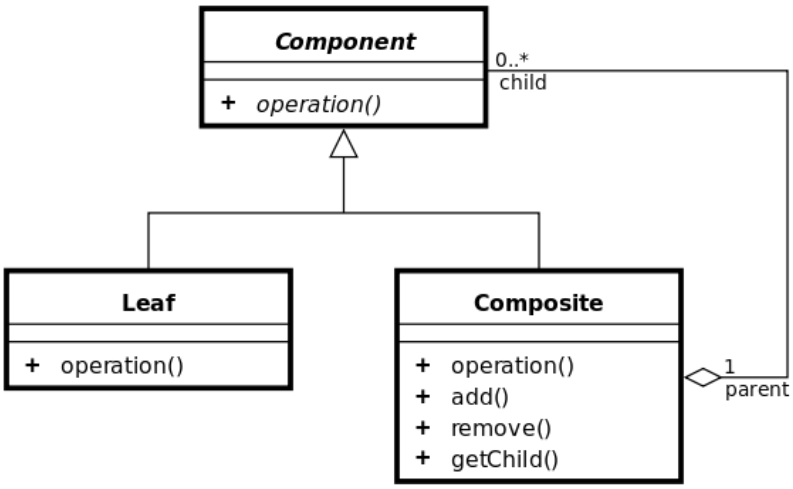
Po načtení se vytváří nová instance takovéto třídy, které se přidává do listu akcí ve třídě Target, které se na konci načítání XML skriptu provedou všechny najednou, avšak postupně tak jak je napsáno ve XML skriptu.

```
namespace TBS
{
    public class Target : Command
    {
        private List<Command> _commands = new List<Command>();

        public void Add(Command command)
        {
            _commands.Add(command);
        }

        public void Execute()
        {
            foreach(Command command in _commands)
            {
                command.Execute();
            }
        }
    }
}
```

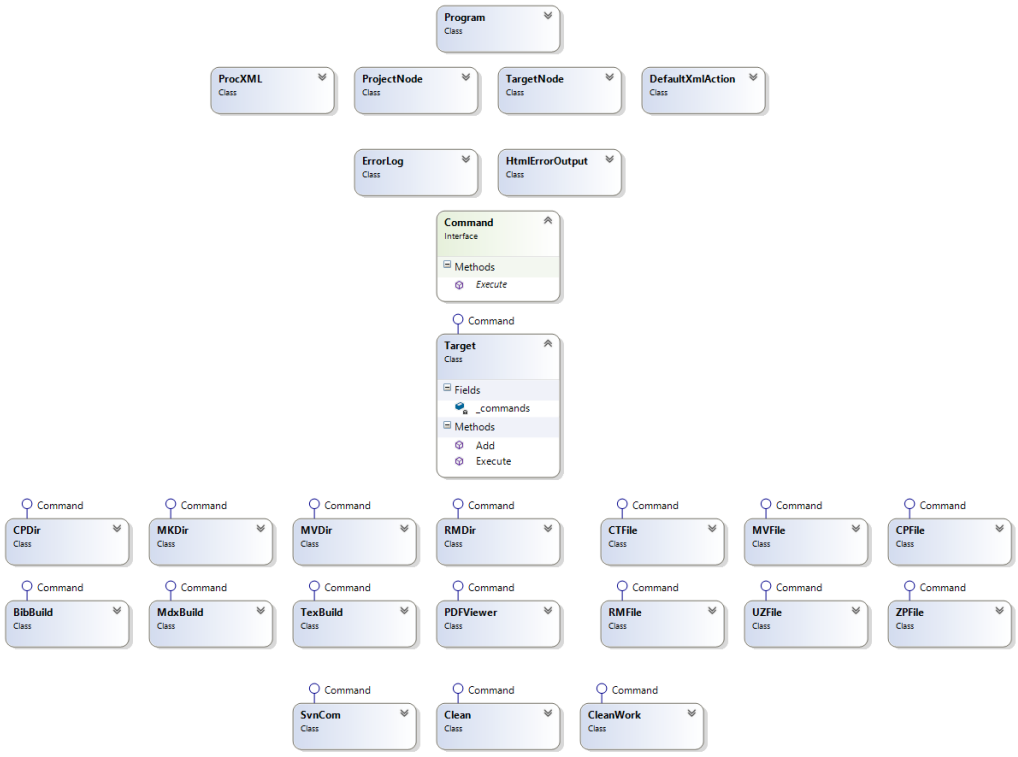
Výpis 12: Třída *target*, která představuje *Composite*, součást návrhového vzoru Kompozit.

Na třídním diagramu 6 se můžete podívat jak je implementován návrhový vzor Kompozit a následně ho porovnat s obrázkem 5.

Samotný překlad LaTeXu probíhá pomocí programu *pdflatex*, který je součástí distribuce TeXLive [27]. TBS v tomto případě předává jméno souboru, které je udáno v XML skriptu, programu *pdflatex* společně s možným módem, ve kterém následně *pdflatex* bude překládat.



Obrázek 5: Návrhový vzor Kompozit.



Obrázek 6: TBS – třídní diagram.

7 Použití programu

Pro používání TBS je nutná znalost základních operací s příkazovou řádkou. Uživatel si musí najít nejdříve adresář, ve kterém se nachází jeho TEXový soubor, který chce přeložit. Musí se taky ujistit, že v daném adresáři se nachází XML skript a *tbs.exe*. Je několik možností jak spustit TBS program. Pokuď uživatel zvolí možnost spustit jen *tbs.exe*, tak se budou vykonávat defaultní akce, což znamená, že *tbs.exe* se bude snažit najít v daném adresáři XML skript se jménem *tbs.xml*. Pokuď *tbs.xml* nenajde, vypíše v příkazové řádce chybovou hlášku. Jestli soubor najde, tak se bude řídit podle „default“ hodnoty, kterou najde v *tbs.xml*. Pokuď se v dané složce už *tbs.xml* nachází tak argumentem může být také jméno „target-u“, který se má vykonat např: *tbs.exe mydelete*. V takovém případě bude *tbs.exe* ignorovat default v XML skriptu a bude se řídit podle vstupního argumentu a podle něho také provede daný „target“. Lze si také vypsat nápovědu „tbs.exe help“.

Předtím než uživatel spustí *tbs.exe* je nutné si nastavit v XML skriptu co vlastně chce všechno aby se provedlo. Uživatel si může nastavit hned několik variací „target-ů“, které budou každý dělat něco jiného a následně je volat jednoduše jeden po druhém v příkazové řádce. Na následném kódu lze shlédnout ukázku použití příkazu *tbs.exe help*.

```
C:\Users\Michal\Desktop\BP2>tbs.exe help
Usage: 1 or 2 arguments -> [optional] or [xml script] [optional]
Available arguments:
help     - shows help
optional - name of target you want to run from your xml script.
```

Závěr

Cílem této bakalářské práce bylo provést analýzu dosavadních sestavovacích systémů a následně vytvořit vlastní sestavovací systém se specifickým zaměřením na LaTeX. Po analýze jsem pro vývoj zvolil platformu .NET Framework od Microsoftu a jako vývojové prostředí jsem použil Microsoft Visual Studio 2013 s podporou jazyka C#, ve kterém je TBS napsán a jako vstupní skript jsem zvolil jazyk XML.

Bylo dosaženo plně funkčního sestavovacího systému se všemi body, které byly v plánu. TBS může být taktéž jednoduše rozšířen o další funkční prvky, protože při jeho implementaci bylo využito jednak objektově orientované programování a dále byl využit návrhový vzor Kompozit. Tudíž další funkcionalitu lze snadno implementovat jako potomky základních tříd a rozhraní v tomto návrhovém vzoru.

8 Literatura

[1] Donald E. Knuth. The TEXbook, Volume A of Computers and Typesetting, Addison-Wesley Publishing Company (1984), ISBN 0-201-13448-9.

[2] Oetiker, Tobias, Hubert Partl, Irene Hyna, Elisabeth Schlegl, Michal Kočer a Pavel Sýkora. Ne příliš stručný úvod do systému LaTeX2e . online, 1996. Dostupné z: http://www.cs.vsb.cz/dvorsky/Download/Elp/lshort2e.pdf

[3] *TeXnicCenter* [online]. [cit. 2014-03-03]. Dostupné z: http://www.texniccenter.org/

[4] LaTeX. *LaTeX* [online]. [cit. 2014-03-03]. Dostupné z: http://en.wikipedia.org/wiki/LaTeX

[5] BibTeX. *BibTeX* [online]. [cit. 2014-03-05]. Dostupné z: http://www.bibtex.org/

[6] Texample. *Texample* [online]. [cit. 2014-03-05]. Dostupné z: http://www.texample.net/tikz/examples/

[7] Gnuplot. *Gnuplot* [online]. [cit. 2014-03-05]. Dostupné z: http://www.gnuplot.info/

[8] Apache Subversion (SVN). *Apache Subversion* [online]. [cit. 2014-03-05]. Dostupné z: http://en.wikipedia.org/wiki/Apache_Subversion

[9] CVS. *CVS* [online]. [cit. 2014-03-05]. Dostupné z: http://cs.wikipedia.org/wiki/CVS

[10] Michel Goossens, Frank Mittelbach and Alexander Samarin. The LATEX Companion. Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8.

[11] Build Automation. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/Build_automation

[12] Make. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/Make_(software)

[13] Distcc. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/Distcc

[14] Mk. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/Mk_(software)

[15] Apache Ant. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/Apache_Ant

[16] NAnt. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/NAnt

[17] Apache Buildr. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/Apache_Buildr

[18] A-A-P. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/A-A-P

[19] CMake. *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/CMake

[20] GNU (Autotools). *Wikipedia* [online]. [cit. 2013-04-16]. Dostupné z: http://en.wikipedia.org/wiki/GNU_Build_Tools

[21] Distcc. *Distcc* [online]. [cit. 2014-03-05]. Dostupné z: https://code.google.com/p/distcc/

[22] Latexmk. *Latexmk* [online]. [cit. 2014-04-07]. Dostupné z: http://users.phys.psu.edu/~collins/software/latexmk-jcc/latexmk-439.pdf

[23] *Arara* [online]. [cit. 2014-04-17]. Dostupné z: http://ctan.mirrorcatalogs.com/support/arara/doc/arara-usermanual.pdf

[24] *Rubber* [online]. [cit. 2014-04-17]. Dostupné z: http://manpages.ubuntu.com/manpages/precise/en/man1/rubber.1.html

[25] *Composite Pattern* [online]. [cit. 2014-04-07]. Dostupné z: http://objekty.vse.cz/Objekty/Vzory-Composite

[26] *Appendix D. Automating TortoiseSVN* [online]. [cit. 2014-04-17]. Dostupné z: http://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-automation.html

[27] *TeX Live* [online]. [cit. 2014-04-28]. Dostupné z: https://www.tug.org/texlive/

A Systémové požadavky

TBS byl odzkoušen na následovných systémech, programech a jejich verzí:

- Windows 8.1
- Adobe Reader 11.0
- TortoiseSVN 1.8.6
- Microsoft Visual Studio 2013 (.NET Framework 4.5+)