



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZOVANÁ SYNTÉZA STROMOVÝCH STRUK- TUR Z REÁLNÝCH DAT

AUTOMATED SYNTHESIS OF TREE STRUCTURES FROM REAL DATA

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. DUŠAN ŽELIAR

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. ALEŠ SMRČKA, Ph.D.

BRNO 2019

Abstrakt

Táto diplomová práca sa zaoberá problematikou analýzy štrukturovaných stromových dát. Cieľom práce je návrh a implementácia nástroja pre automatizovanú detekciu závislostí vzoriek reálnych dát, pričom zohľadňuje ich stromovú štruktúru a hodnoty uzlov. Abstrakcia konkrétnych hodnôt uzlov je dosiahnutá pomocou klasterizácie. Nástroj vytvorí predpis pre syntézu umelých dát, ktoré sú významom a štruktúrou podobné reálnym vzorkám. Nástroj je súčasťou platformy Testos.

Abstract

Do tohoto odstavce bude zapsán výťah (abstrakt) práce v anglickém jazyce.

Klíčové slova

testovanie založené na dátach, syntéza, analýza, stromové štruktúry, XML, JSON, Testos

Keywords

data-driven testing, synthesis, analysis, tree structures, XML, JSON, Testos

Citácia

ŽELIAR, Dušan. *Automatizovaná syntéza stromových štruktúr z reálných dát*. Brno, 2019. Diplomová práca. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Aleš Smrčka, Ph.D.

Automatizovaná syntéza stromových struktur z reálných dat

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....

Dušan Želiar
5. januára 2019

Podakovanie

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant, apod.).

Obsah

1	Úvod	2
2	Testovanie softwaru	4
2.1	Úrovne testovania	4
2.2	Dynamické testovanie	5
2.3	Testovanie založené na dátach	6
2.4	Editácia a uloženie testovacích dát	6
2.5	Ekvivalenčné triedy a kritéria pokrytia	8
2.6	Grafy príčin a dôsledkov	9
2.7	Testos	11
3	Štrukturované dáta	13
3.1	Grafová reprezentácia	13
3.2	Porovnávanie dát	13
3.3	Klasterizácia	13
3.4	Prehľad používaných formátov	13
3.5	Problém generovania	13
3.6	Existujúce generátory	13
4	Záver	14
	Literatúra	15
A	Jak pracovať s touto šablónou	16
B	Psaní anglického textu	20
C	Checklist	24
D	L^AT_EXpro začátečníky	28

Kapitola 1

Úvod

Tento text slouží jako ukázkový obsah šablony a současně rekapituluje nejdůležitější informace z předpisů a poskytuje další užitečné informace, které budete potřebovat pro tvorbu technické zprávy ke svojí práci. Než se šablonou budete dále pracovat, je třeba vědět, jak ji správně použít. To je stručně uvedeno v příloze A.

I když některým studentům pro napsání dobré diplomové práce (bakalářská práce je také diplomová – dostává se za ni diplom) stačí znát a dodržovat oficiální formální požadavky uvedené ve směrnících a typografické zásady, často je výhodné před započatím psaní zjistit, jaké jsou osvědčené postupy pro psaní odborného textu a jak si práci usnadnit. Někteří vedoucí svým studentům připravili popisy osvědčených postupů, které vedly k desítkám úspěšně obhájených prací. Výběr nejzajímavějších postupů, které měli autoři této šablony k dispozici ve chvíli její tvorby, je v níže uvedených kapitolách. Má-li Váš vedoucí svoji stránku s doporučenými postupy, tyto kapitoly můžete vynechat a řídit se pokyny svého vedoucího. Pokud takovou stránku nemá, může být přečtení níže uvedeného textu vhodnou přípravou na konzultaci o plánované struktuře a náplni textu práce.

Diplomová práce je rozsáhlé dílo a tomu odpovídá i technická zpráva. Ne každý je schopen si sednout a jednoduše ji napsat. Je třeba vědět, kde začít a jak postupovat. Jedním z možných přístupů je začít psaním klíčových slov a abstraktu, abyste si ujasnili, co je v práci nejdůležitější. O tom pojednává kapitola ??.

Po sepsání abstraktu se lze pustit do psaní samotného textu technické zprávy. Typicky si nejprve připravíme základní strukturu práce, kterou pak budeme plnit textem. Kapitola ?? se zabývá základními informacemi a radami pro psaní odborného textu, které Vám pomohou vyhnout se začátečnickým chybám, a stanovením nadpisů kapitol a přibližných rozsahů jednotlivých částí práce. V závěru kapitoly je pak uveden přístup, kterým si lze psaní technické zprávy značně usnadnit.

Diplomové práce v oblasti informačních technologií mají určitou typickou strukturu. Po úvodu bude následovat kapitola či kapitoly zabývající se shrnutím současného stavu, který bude v následujících kapitolách zhodnocen a bude navrženo řešení, které bude implementováno a otestováno. V závěru pak budou výsledky vyhodnoceny a bude navržen budoucí vývoj. I když se názvy a rozsahy kapitol v různých pracích liší, vždy tam lze najít kapitoly odpovídající této struktuře. Kapitola ?? se zabývá obsahy typických kapitol, které se v diplomových pracích z oblasti IT vyskytují. Většina studentů ve svojí práci pravděpodobně využije pouze určitou podmnožinu popsanych kapitol, která je pro jejich práci relevantní. Uvedené popisy a rady mohou pomoci jak s rozhodnutím, zda danou kapitolu uvést, tak i s vnitřní strukturou a samotným obsahem kapitoly.

Za závěrečnou kapitolou práce vždy následuje seznam použité literatury. Citacemi, které tento seznam tvoří, a odkazy na ně se zabývá kapitola ???. Byť to tak nezkušený student nemusí vnímat, je seznam použité literatury a odkazy na něj pro práci zcela zásadní. Hodnocení práce s literaturou a citací tvoří jednu z důležitých částí posudku oponenta a bude-li chybět jediná položka, může to vést k hodnocení stupněm F, následnému disciplinárnímu řízení za plagiátorství a k vyloučení z nedokončeného studia. Nesprávná práce se zdroji může mít i další důsledky – v roce 2018 stála křesla dva členy české vlády. Proto prosím citacím věnujte odpovídající pozornost.

Po dokončení textu je nutné zjistit, jaké požadavky jsou kladeny na vysokoškolskou kvalifikační práci na FIT VUT v Brně, a dořešit případné nedostatky. Formální požadavky jsou uvedeny ve směrnících a na webových stránkách, které jsou zmíněny v kapitole ??. Tato kapitola obsahuje i požadované rozsahy jednotlivých typů prací a další vybrané informace z předpisů a doporučení. V závěru kapitoly je uveden přehled nejčastějších chyb, se kterými se oponenti setkávají a kterým byste se měli vyhnout. Hodnocení formální úpravy práce je pak další z důležitých součástí posudku oponenta.

Po odstranění formálních nedostatků lze práci odevzdat. Před odevzdáním práce si můžete projít kontrolní seznam (tzv. „checklist“) uvedený v příloze C. Samotné odevzdání listinné i elektronické verze práce je pak popsáno v kapitole ??.

V závěrečné kapitole 4 je pak uvedeno shrnutí toho, co se lze přečtením tohoto textu dozvědět, a to nejdůležitější, na co je třeba myslet před odevzdáním práce.

Kapitola 2

Testovanie softwaru

Testovanie softvéru je súbor procesov analyzujúcich softvér za účelom vyhodnotenia jeho vlastností a detekcie rozdielov medzi aktuálnym a požadovaným stavom [1]. Táto kapitola najskôr predstavuje základné úrovne a druhy testovania. Následne popisuje testovanie založené na dátach a s tým spojenú rozhodovaciu tabuľku.

[2][4][5][8][3][6][1].

2.1 Úrovne testovania

Testy sú vytvárané na základe špecifikácií a požiadaviek, dizajnových artefaktov alebo zdrojového kódu. Rôzne úrovne testovania sprevádzajú rozdielne vývojárske aktivity [2].

Jednotkové testovanie

Jednotkové testovanie je zamerané na jednotky, predstavujúce najmenšie testovateľné komponenty testovaného systému. Zmyslom jednotkového testovania je validácia správania jednotky voči jej dizajnu. Zameriava sa na chyby na najnižšej úrovni. Testy vytvárajú samotný programátori počas vývoja.

Integračné testovanie

Integračné testovanie je cielené na korektnú komunikáciu medzi rozhraniami jednotiek, ktoré sú pre tento účel zlučované do podsystémov. Úlohou integračného testovania nie je nájdenie chýb v jednotlivých integrovaných jednotkách, ale overenie ich korektnej integrácií. Predpokladá sa, že tieto chyby boli eliminované pri jednotkovom testovaní. Odhaľuje chyby v rozhraniach a stavoch jednotiek. Pri väčšom množstve rozhraní je vhodné zvoliť špecifický prístup k integračným testom. Obvyklé stratégie sú zdola nahor, zhora dole, funkcionálna integrácia a veľký tresk [4]. Obvykle ich tvoria vývojári alebo tester v rámci tímu.

Systémové testovanie

Systémové testovanie je zamerané na nájdenie chýb vo vlastnostiach plne integrovaného systému. Testovaný systém je tvorený komponentami, ktoré už úspešne prešli integračnými testami. Cieľom je detekcia nekonzistentností medzi týmito komponentami a systému ako celku voči špecifikácií požiadavkov. Systémové testovanie vykonáva oddelená skupina testerov mimo vývojový tím.

Akceptačné testovanie

Akceptačné testovanie je proces s účelom overenia softvéru voči počiatočným stanoveným požiadavkom zákazníka jeho aktuálnych potrieb. Často sa na ich vytváraní podieľa expert na doménu, pre ktorý sa softvér vyvíja. Zvyčajne je vytvorený zákazníkom alebo koncovým užívateľom a overuje, že dané riešenie pre užívateľa funguje [5].

2.2 Dynamické testovanie

Existuje mnoho prístupov k testovaniu softvéru. Na najvyššej úrovni sa testovanie rozdeľuje na *statické* a *dynamické* [2]. Techniky, ktoré analyzujú a skúmajú program bez nutnosti jeho spusteniam za účelom verifikácie spadajú do skupiny statického testovania. Zahŕňa posudzovanie dokumentov, kódu a jeho statickú analýzu (základná statická analýza zvyčajne prebieha na úrovni kompilátorov). Druhá skupina techník spadá do skupiny dynamického testovania, ktorá je zameraná na analýzu dynamického správania kódu s cieľom jeho validácie. Prerekvizitou je úspešná kompilácia a spustenie kódu. Zahŕňa prácu so softvérom, kedy pre špecifické vstupy overuje a analyzuje správnosť výstupov.

Dynamické testovanie môže byť ďalej rozdelené na *funkcionárne* a *nefunkcionárne*. Kým funkcionárne testovanie adresuje splnenie požiadaviek, nefunkcionárne je mierené na ostatné oblasti ako bezpečnosť, výkonnosť, použiteľnosť, správa pamäti a iné. Podľa znalosti kódu sa delí na *black-box*, *white-box* a *grey-box* testovanie.

Táto práca sa zameriava na testovanie založené na dátach, ktoré vychádza z funkcionálneho black-box testovania, ale výsledný nástroj môže byť prínosný aj pre iné prístupy.

Black-box testovanie

Black-box testovanie (tiež známe ako *testovanie založené na dátach*) zoskupuje techniky tvorby testovacích prípadov na základe špecifikácií podľa analýzy popisu softvéru bez znalosti jeho vnútornej štruktúry [5]. Ich hlavným zameraním je odhalenie okolností, pri ktorých sa systém správa odlišne od špecifikácií. Testovacie dáta závisia na popise očakávaní od testovaného softvéru napríklad vo forme manuálu či popisu procesu.

Black-box testovanie môže byť využité na všetkých úrovniach testovania. Pre nižšie úrovne jednotkového a integračného testovania sa dá využiť ako počiatočný bod pre tvorbu testov na základe dizajnu alebo aj požiadaviek. Veľmi užitočné sú na vyšších úrovniach, teda systémová a akceptačná, kde sú testy založené na požiadavkách [4].

White-box testovanie

Techniky white-box testovania vytvárajú testovacie prípady podľa vnútornej štruktúry komponenty alebo systému. Hlavný dôraz kladú na vetvy, jednotlivé podmienky a výrazy tradične v zdrojovom kóde. Primárne sa využívajú v jednotkovom a integračnom testovaní. Všetky testovacie techniky tohoto druhu od testera vyžadujú znalosť danej štruktúry, teda programovacieho jazyka [4].

Grey-box testovanie

Mezi white-box a black-box testovaním je mnoho úrovní grey-box testovania ktoré predstavuje ich kombináciu. Testovacie prípady sú tvorené so znalosťou architektúry, algoritmov, vnútorných stavov alebo iného vysoko úrovňového popisu správania.

2.3 Testovanie založené na dátach

Jednoduché automatizované testovacie skripty obsahujú pevne dané testovacie dáta. Zmena týchto dát obvykle vyžaduje zmenu v zdrojovom kóde skriptu, čo môže viesť k viacerým komplikáciám. Ak je test neprehľadný, dlhý alebo neštrukturovaný, jednoduchá zmena v dátach je náročná aj pre skúsených expertov. Rovnako vzniká riziko zavedenia novej chyby. Pri tvorbe nových testov, líšiacich sa len v testovacích dátach, často dochádza k skopírovaniu kódu a následnej modifikácii dát. Pritom nastáva duplicita kódu a testovacie prípady sú ťažko udržiateľné.

Pri veľkých testovacích sadách sú pre spomenuté problémy skripty s pevne danými dátami len ťažko použiteľné. *Testovanie založené na dátach (Data-driven testing)* je metodológia, v ktorej sa opakovane vykonávajú rovnaké kroky skriptu s použitím externých dátových zdrojov. Takéto dáta musia byť ľahko editovateľné aj testerom bez znalosti zdrojového kódu. Umožňuje mu tým sústrediť sa len na tvorbu testovacích prípadov. Výhody metodológie sú zreteľné najmä pri aplikáciách s častými zmenami funkcionality. Hlavné výhody sú nasledovné [7]:

- Testy založené na dátach dosahujú vysoké pokrytie kódu testovacími prípadmi a zároveň minimalizujú množstvo kódu, ktoré je potrebné napísať a udržiavať
- Uľahčuje vytváranie a spúšťanie veľkého množstva testovacích podmienok
- Testovacie dáta môžu byť navrhnuté a vytvorené pred tým, ako je aplikácia pripravená na testovanie
- Rozhodovacie dátové tabuľky môžu byť použité pri manuálnom testovaní



Obr. 2.1: Diagram základnej štruktúry testovania založeného na dátach .

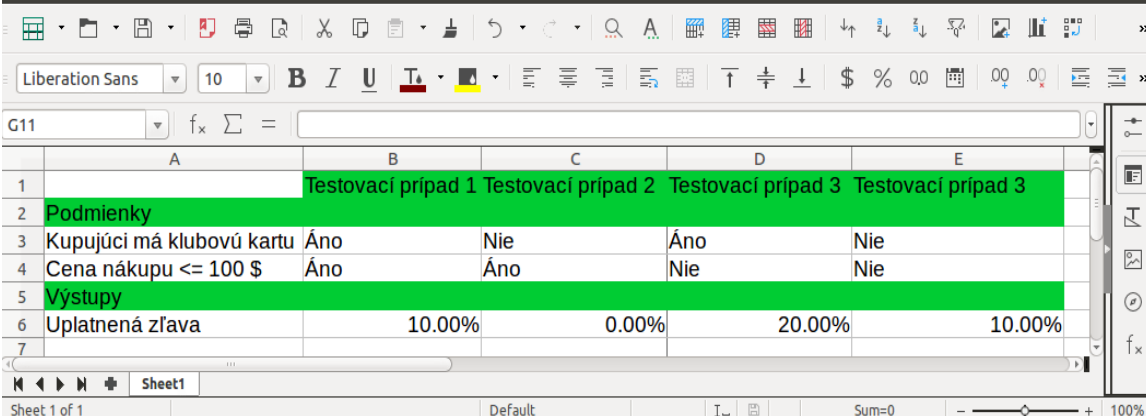
2.4 Editácia a uloženie testovacích dát

Využívané testovacie dáta sa obecnne skladajú z kombinácie vstupných a očakávaných výstupných dát. Daný typ dát sa najlepšie popisuje formou *rozhodovacích tabuliek*. Rozhodovacia tabuľka v najjednoduchšej forme poskytuje vstupy ako aj očakávané výstupy na jednom riadku. Pri tvorbe tabuľky je dôležitá správna identifikácia všetkých vstupných dát

a ich rozdelenie do *domén*. Výber konkrétnych hodnôt závisí od zvoleného prístupu [5]. Najčastejšie sa využívajú nasledovné prístupy:

- *Testy pokrývajúce logiku*. Testovacie prípady spoločne dosahujú všetky definované výstupy a rovnako zaručujú vykonanie všetkých častí kódu minimálne raz.
- *Rozdelenie na ekvivalenčné triedy* 2.5. Vstupy sa rozdeľujú do tried za účelom redukcie počtu testov. Predpokladá sa, že test na jednom prvku triedy reprezentuje všetky ostatné.
- *Analýza hraničných hodnôt*. Prístup využíva ekvivalenčné triedy, jednotlivých reprezentantov ale nevolí náhodne. Keďže najčastejšie chyby sa vyskytujú pri hraničných hodnotách, konkrétne hodnoty volí z hraničných oblastí. Zohľadňuje pritom vstupné aj výstupné dáta.
- *Grafy príčin a dôsledkov (angl. Cause-Effect Graph)* 2.6. Vytvárajú logickú grafovú reprezentáciu medzi požiadavkami a výsledkami testov. Pomáhajú pri výbere účelných a úplných testov.

Vzhľadom k charakteru dát sa k ich editácii prirodzene ponúka využitie tabuľkových procesorov (anglicky *spreadsheet*). Prácu s danými programmi obvykle zvládajú testeria ako aj ľudia z oblasti biznisu, čo uľahčuje ich rýchle zapojenie. Dané programy sa často používajú aj na jednoduchý manažment testov pre manuálne testovanie. V tomto prípade sa dáta môžu zdieľať s automatizovanými testami a predchádzať tak ich zbytočnej redundancii. Príklad tabuľky je uvedený na obrázku 2.2.



	A	B	C	D	E
1		Testovací prípad 1	Testovací prípad 2	Testovací prípad 3	Testovací prípad 3
2	Podmienky				
3	Kupujúci má klubovú kartu	Áno	Nie	Áno	Nie
4	Cena nákupu <= 100 \$	Áno	Áno	Nie	Nie
5	Výstupy				
6	Uplatnená zľava	10.00%	0.00%	20.00%	10.00%
7					

Obr. 2.2: Príklad rozhodovacej tabuľky pre uplatnenie zľavy vytvorenej v tabuľkovom editore.

Formáty uloženia tabuliek spadajú do viacerých kategórií. Prostý databázový súbor (anglicky *flat file database*) je jednoduchá databáza (väčšinou tabuľka) uložená v textovom súbore ve forme prostého textu. Obvykle používané formáty sú napríklad hodnoty oddelené čiarkami (CSV), hodnoty oddelené tabulátormi (CSV, TXT) alebo iný špecifický formát (variácie XLS). Rovnako sú stále viac využívané štrukturované formáty ako XML a Json. Súčasné programovacie jazyky majú knižnice pre ich načítanie a spracovanie, čo výrazne uľahčuje ich využitie. Prosté databázové súbory môžu mať problémy pri výraznom rozširovaní. Rovnako je v nich neprehľadné uchovávanie rôznych konfigurácií a verzií.

Pre veľké množstvo testovacích dát je vhodné využitie relačnej databázy. Umožňuje editáciu prostredníctvom skriptov ako aj pomocou grafických editorov.

2.5 Ekvivalenčné triedy a kritéria pokrytia

Zmyslom tvorby testovacích prípadov je nájdenie vstupov, ktoré najlepšie pokrývajú zvolené kritérium pokrytia. V závislosti od zložitosti testovaného softvéru môže byť množstvo možných vstupov potenciálne nekonečné, preto je zvolenie vhodnej množiny testovacích dát náročné [4].

Rozdelenie vstupných domén na ekvivalenčné triedy

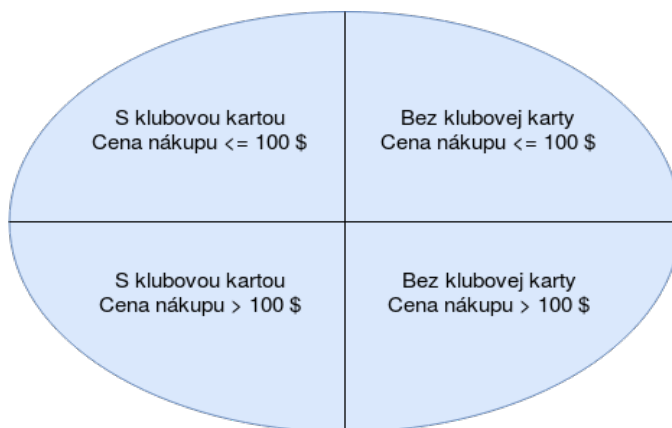
Kľúčová je správna identifikácia vstupných domén. *Vstupná doména* testovaného systému je definovaná množinou všetkých vstupných hodnôt, ktoré nadobúda. V závislosti na testovacej úrovni a testovaného artefaktu sú to obvykle parametre metód, statické a globálne premenné, objekty reprezentujúce stav systému, užívateľské vstupy a argumenty programu [2]. Vstupná doména je následne rozdelená na *ekvivalenčné triedy* (označované aj ako bloky). Pojem ekvivalencie sa vzťahuje k predpokladu, že všetky hodnoty v jednej triede obsahujú z pohľadu testovania rovnako užitočné hodnoty. Každý prvok patrí práve do jednej triedy a jediný prvok z ekvivalenčnej množiny reprezentuje všetky prvky. Keď overíme testovací prípad pre jeden prvok, predpokladáme, že sme overili všetky prvky z danej množiny.

Pri rozpade domény D podľa rozdelenia q vznikajú vzájomne disjunktné ekvivalenčné triedy (bloky) B_q definované nasledovne:

$$b_i \cap b_j = \emptyset, i \neq j; b_i, b_j \in B_q$$

a spolu pokrývajú doménu D

$$\bigcup_{b \in B_q} b = D$$



Obr. 2.3: Rozpad domén z tabuľky 2.2 na ekvivalenčné triedy

Kritéria pokrytia

Po identifikácii vstupných blokov je ďalší krok vytvorenie konkrétnej testovacej sady. Efektívne testovanie množstva blokov vyžaduje vhodný výber ich kombinácií. Stratégie zvolenia kombinácií sú dané konkrétnym krytériom pokrytia. *Kritérium pokrytia* (angl. *Coverage*

criterion) je pravidlo alebo predpis pre systematické generovanie požiadavkov na test. *Pokrytie* (angl. *coverage*) je miera udávajúca, ako veľmi daná testovacia sada skúma testovaný systém. Obvykle sa udáva v percentách a viaže sa na konkrétne kritérium [2].

- *Kritérium pokrytia všetkých kombinácií* (angl. *All Combinations Coverage*). Kritérium vyžaduje pokrytie všetkých kombinácií blokov zo všetkých domén. Testovacie prípady spoločne dosahujú všetky definované výstupy a rovnako zaručujú vykonanie všetkých častí kódu minimálne raz. Reálne sa dá použiť len pri minimálnom množstve blokov.
- *Kritérium pokrytia všetkých párov blokov* (angl. *Pair-Wise Coverage*). Kritérium vyžaduje kombináciu každého bloku každej domény s každým blokom každej inej domény, teda všetkých dvojíc blokov z rôznych domén. Generalizáciou kritéria je *Kritérium pokrytia všetkých n-tíc blokov* (angl. *T-Wise Coverage*).
- *Kritérium pokrytia bazových blokov* (angl. *Base Choice Coverage*). Pre každú doménu je zvolený bazový blok, zvyčajne ide o najčajstejší alebo najdôležitejší blok. Kritérium vyžaduje kombinácie všetkých bazových blokov každej domény a pokrytie každého nebazového bloku. Vhodne zvolené kombinácie bazových blokov výrazne redukujú celkový počet testovacích prípadov. Vyšší stupeň predstavuje *Kritérium pokrytia viacerých bazových blokov* (angl. *Multiple Base Choices Coverage*).
- *Kritérium pokrytia každého bloku* (angl. *Each Choice Coverage*). Kritérium vyžaduje pokrytie každého bloku pre každú doménu. Minimálny počet testov sa rovná počtu blokov. Kritérium nie je veľmi efektívne a samotnú voľbu testovacích prípadov necháva na testerovi. Nevyžaduje žiadnu kombináciu hodnôt a preto je považované za slabé.

2.6 Grafy príčin a dôsledkov

Slabinou predstaveného prístupu založeného na rozklade na ekvivalenčné triedy je absencia kombinácií vstupov. Riešenie ponúkajú spomenuté kritéria pokrytia, ale počet kombinácií vstupných dát je napriek tomu obvykle príliš vysoký. *Graf príčin a dôsledkov* (angl. *Cause-Effect Graphing, CEG*) je grafický spôsob znázornenia prepojenia vstupov (*príčiny*, angl. *causes*) s nimi asociovanými výstupmi (*dôsledky*, angl. *effects*). Graf je formálne vyjadrenie boolovských požiadavkov a umožňuje systematický spôsob výberu podmnožiny testovacích prípadov. Výhody prístupu sú nasledovné:

- Redukcia počtu kombinácií vstupov
- Odhalenie nejasností a nekompletnosti špecifikácie
- Zrozumiteľný a jasne čitateľný formát
- Zlepšenie celkového chápania systému a jeho dôležitých faktorov
- Pomoc pri hľadaní zdroja konkrétnej príčiny a dôsledkov

Pre tvorbu testovacích prípadov je použitý nasledovný proces [5]:

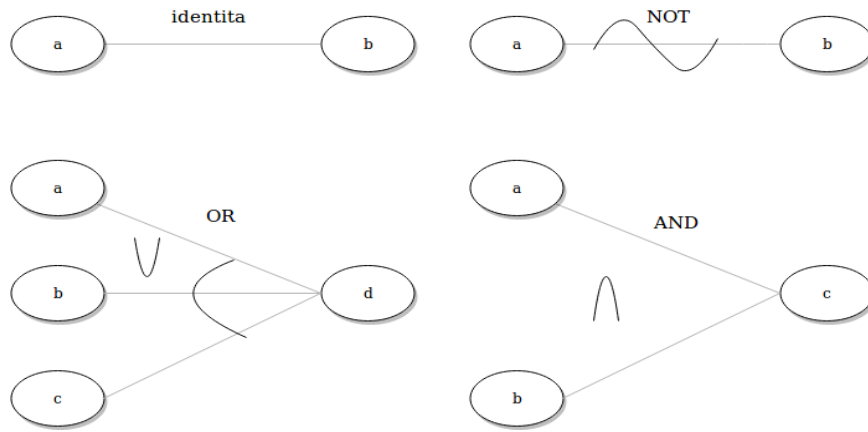
1. *Rozdelenie požiadavkov do skupín primeranej veľkosti*. Veľké množstvo požiadavkov by vyústilo do príliš veľkého grafu, ktorý sa stane nepoužiteľným.
2. *Identifikácia príčin a dôsledkov*. Príčina je priamo vstupná podmienka alebo jej ekvivalenčná trieda. Dôsledok je výstupná podmienka alebo zmena v systéme.

3. Analýza významu požiadavkov a vytvorenie grafu príčin a dôsledkov.
4. Identifikácia obmedzení medzi príčinami a dôsledkami.
5. Konvertovanie grafu do rozhodovacej tabuľky.
6. Každý stĺpec v tabuľke reprezentuje testovací prípad.

Notácia grafu

Základná notácia grafu je ukázaná na obrázku 2.4. Každý uzol môže nadobúdať hodnoty 0 a 1 reprezentujúce absenciu a prítomnosť stavu. Celkovo syntax pokrýva štyri prípady [5]:

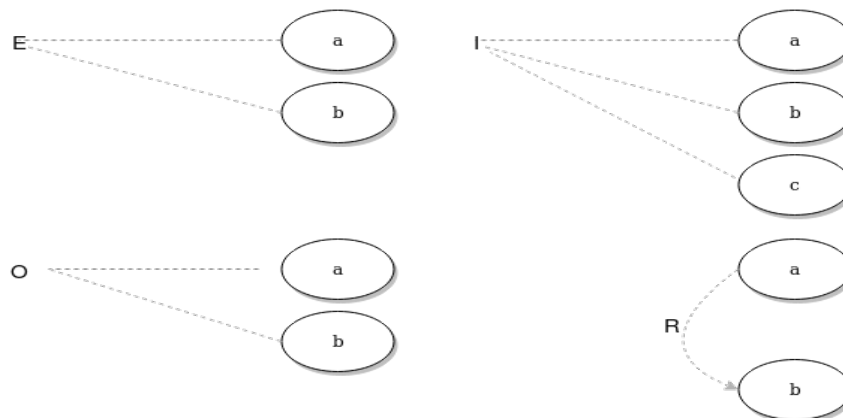
- Funkcia *identity*: ak $a = 1$ tak $b = 1$, inak $b = 0$
- Funkcia *NOT*: ak $a = 1$ tak $b = 0$, inak $b = 1$
- Funkcia *OR*: ak a , b alebo c je rovné 1, tak $d = 1$
- Funkcia *AND*: ak $a = 1$ a súčasne $b = 1$, tak $c = 1$, inak $c = 0$



Obr. 2.4: Základné symboly CEG grafu [5].

Niektoré kombinácie medzi príčinami a dôsledkami sú neuskutočniteľné. Grafová reprezentácia na obrázku 2.5 preto umožňuje obmedzenia popísať nasledovne [5]:

- Obmedzenie *E*: Najviac jeden z uzlov a a b je súčasne rovný 1
- Obmedzenie *I*: Aspoň jeden z uzlov a , b a c je vždy rovný 1
- Obmedzenie *O*: Práve jeden z uzlov a a b je rovný 1
- Obmedzenie *R*: Aby a moholo byť 1, b musí byť 1



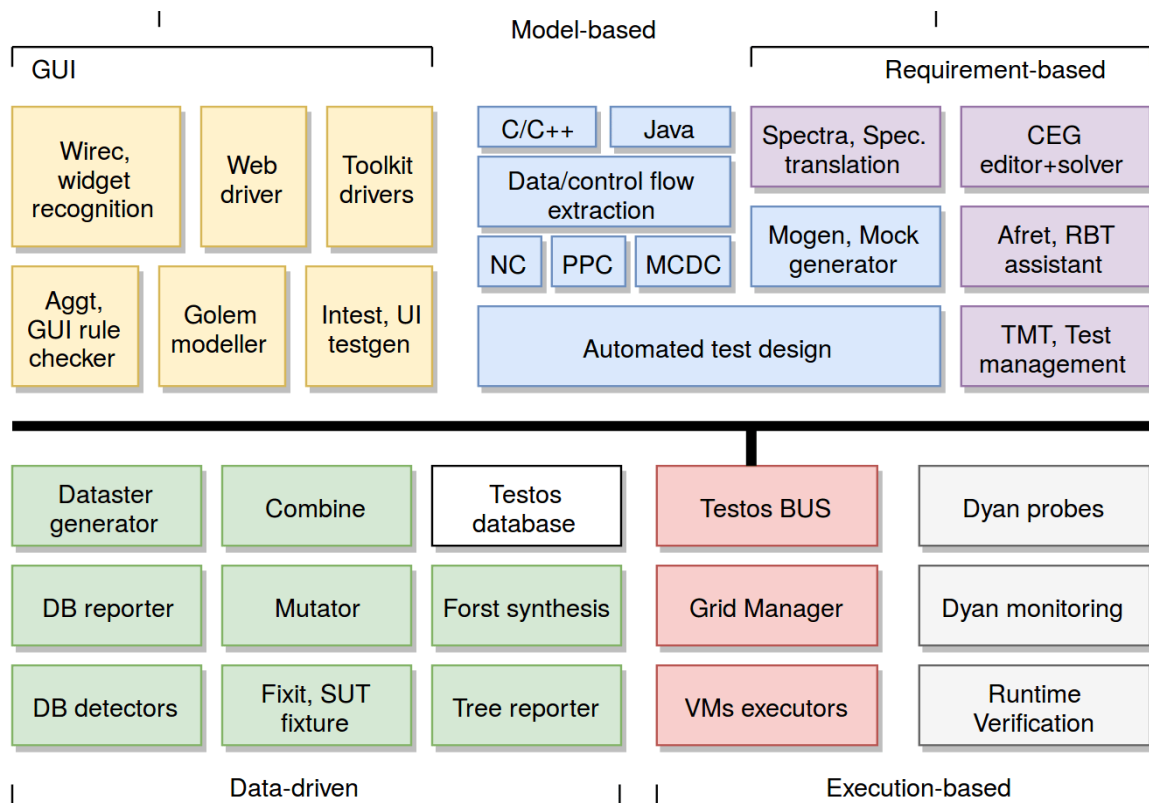
Obr. 2.5: Základné symboly CEG grafu [5].

2.7 Testos

Platforma *Testos* (Test Tool Set) [6] je projekt, ktorého hlavný cieľ je vytvorenie ucelenej sady nástrojov podporujúce automatizované testovanie softvéru. Platforma sa sústreďuje na všetky úrovne testovania a podľa zamerania je rozdelená na oblasti:

- Testovanie grafického užívateľského rozhrania (*GUI*)
- Testovanie založené na modeloch (*Model-based*)
- Testovanie založené na požiadavkách (*Requirement-based*)
- Testovanie založené na dátach (*Data-based*)
- Dynamická analýza (*Execution-based*)

Oblasť testovania založeného na dátach aktuálne obsahuje nástroje pre generovanie testovacích dát pre databázy (náhodné dáta, kombinácie dát a ich mutácie), detektory databázovej štruktúry a detektory štrukturovaných dát. Nástroj vytvorený v tejto práci patrí do rovnakej oblasti, v rámci nej priamo komunikuje a využíva ostatné nástroje Testos.



Obr. 2.6: Platforma Testos [6].

Kapitola 3

Štrukturované dáta

Disponovanie vhodnými *testovacími dátami* je pre proces testovania rovnako dôležité ako konkrétne prípady. S narastajúcou veľkosťou a komplexnosťou systémov je získanie týchto dát stále obtiažnejšie. V tejto kapitole je najskôr uvedená obecná charakteristika štrukturovaných dát a spôsob ich grafovej reprezentácie. Následne sa venuje metódam ich porovnania z pohľadu štruktúry a sémantiky, pričom rozoberá problematiku klasterizácie. Nakoniec poskytne prehľad používaných formátov a existujúcich riešení získania testovacích dát.

3.1 Grafová reprezentácia

3.2 Porovnávanie dát

3.3 Klasterizácia

3.4 Prehľad používaných formátov

3.5 Problém generovania

3.6 Existujúce generátory

Kapitola 4

Závěr

V tomto textu bylo uvedeno, jak začít s tvorbou bakalářské či diplomové práce, napsat abstrakt, připravit základní strukturu práce a co uvést do jednotlivých kapitol. Při tom bylo vysvětleno, že bakalářská práce je také diplomová a je třeba k ní přistupovat stejně zodpovědným způsobem. Následně byla věnována pozornost bibliografickým citacím a formální stránce práce. V předposlední kapitole jsou uvedeny důležité informace k odevzdání v listinné i v elektronické podobě.

Je třeba zdůraznit, že diplomová práce je unikátním individuálním dílem, které vzniká pod vedením zkušeného odborníka. Ať už je v této šabloně uvedeno cokoli, závazné jsou pouze oficiální pokyny na stránkách fakulty. Pro konkrétní diplomovou práci je potřeba vždy zvažovat, co je z výše uvedeného textu relevantní a co nikoliv a řídit se především pokyny vedoucího, který rozumí dané problematice a je tak schopen poskytnout ty nejlepší rady, co lze k práci dostat.

I přes velkou snahu nikdy není možné do šablony zahrnout všechny prvky, co budou při tvorbě práce potřeba, a zaručit, že po doplnění textu, obrázků, literatury apod. bude vše v pořádku pro všechny možné diplomové práce. Bude-li někde delší text, než se předpokládalo, a zalomí-li se na dva řádky, bude-li v literatuře položka, se kterou nebyl otestován využitý styl, a v dalších případech může být výsledek neuspokojivý a může být potřeba do šablony zasáhnout a chybu, která se projevuje třeba jen pro jednu práci ze sta, opravit. Výsledné PDF a následně i vytištěnou papírovou verzi je tedy vždy nutné pečlivě zkontrolovat a nespolehat se na to, že „tohle přece generuje šablona, tak to musí být správně“. Najdete-li v šabloně nějaké chyby nebo budete-li mít návrhy na její vylepšení, napište prosím na e-mail sablona@fit.vutbr.cz a pomozte nám s jejím vylepšováním. Veškeré připomínky a návrhy jsou vítány.

S kontrolou výsledku může výrazně pomoci vedoucí práce. Nelze však předpokládat, že vedoucí poslední noc před odevzdáním bude sedět v práci připraven na kontrolu desítek stran textu. Proto je nutné mít vše připravené v předstihu a konzultovat průběžně. Kritický pohled vedoucího pak umožní dosažení kvalitního výsledku a aktivita, kterou uvidí, přispěje k pozitivnímu hodnocení práce z jeho strany.

Na závěr bych jménem autorů této šablony popřál všem, kteří právě vytvářejí svoje diplomové práce nebo se k jejich tvorbě připravují, úspěšné dokončení a obhájení práce.

Literatúra

- [1] IEEE Standard for Software and System Test Documentation. *IEEE Std 829-2008*, July 2008: s. 1–150, doi:10.1109/IEEESTD.2008.4578383.
- [2] Ammann, P.; Offutt, J.: *Introduction to Software Testing*. Cambridge University Press, 2008, ISBN 978-0-511-39330-3.
- [3] Emmi, M.; Majumdar, R.; Sen, K.: *Dynamic Test Input Generation for Database Applications*. In Proceedings of the 2007 International Symposium on Software Testing and Analysis, *ISSTA '07, New York, NY, USA: ACM, 2007, ISBN 978-1-59593-734-6*, s. 151–162.
URL <http://doi.acm.org/10.1145/1273463.1273484>
- [4] Hass, A. M. J.: *Guide to Advanced Software Testing*. Artech House, 2008, ISBN 978-1-59693-285-2.
- [5] Myers, G. J.; Badgett, T.; Sandler, C.: *The Art of Software Testing*. John Wiley, 3 vydanie, 2011, ISBN 978-1-118-03196-4.
- [6] Testos: Domovská stránka projektu Testos. FIT VUT v Brně, 2018, [Online; navštívené 30.12.2018].
URL <http://testos.org>
- [7] Unmesh, G.: *Selenium Testing Tools Cookbook*. Packt Publishing, 2012, ISBN 1849515743, 9781849515740.
- [8] Zhao, R.; Li, Z.; Wang, Q.: *Test Generation for Programs with Binary Tree Structure as Input*. International Journal of Software Engineering and Knowledge Engineering, ročník 25, č. 07, 2015: s. 1129–1151.
URL <https://doi.org/10.1142/S0218194015500205>

Príloha A

Jak pracovat s touto šablonou

V této příloze je uveden popis jednotlivých částí šablony, po kterém následuje stručný návod, jak s touto šablonou pracovat. Pokud po jejím přečtení k šabloně budete mít nějaké dotazy, připomínky apod., neváhejte a napište na e-mail sablona@fit.vutbr.cz.

Popis částí šablony

Po rozbalení šablony naleznete následující soubory a adresáře:

bib-styles Styly literatury (viz níže).

obrazky-figures Adresář pro Vaše obrázky. Nyní obsahuje placeholder.pdf (tzv. TODO obrázek, který lze použít jako pomůcku při tvorbě technické zprávy), který se s prací neodevzdává. Název adresáře je vhodné zkrátit, aby byl jen ve zvoleném jazyce.

template-fig Obrázky šablony (znak VUT).

fitthesis.cls Šablona (definice vzhledu).

Makefile Makefile pro překlad, počítání normostran, sbalení apod. (viz níže).

projekt-01-kapitoly-chapters.tex Soubor pro Váš text (obsah nahraďte).

projekt-20-literatura-bibliography.bib Seznam literatury (viz níže).

projekt-30-prilohy-appendices.tex Soubor pro přílohy (obsah nahraďte).

projekt.tex Hlavní soubor práce – definice formálních částí.

Výchozí styl literatury (czechiso) je od Ing. Martínka, přičemž slovenská a anglická verze (slovakiso a englishiso) jsou jeho překlady s drobnými modifikacemi. Oproti normě jsou v něm určité odlišnosti, ale na FIT je dlouhodobě akceptován. Alternativně můžete využít styl od Ing. Radima Loskota nebo od Ing. Radka Pyšného¹. Alternativní styly obsahují určitá vylepšení, ale zatím nebyly řádně otestovány větším množstvím uživatelů. Lze je považovat za beta verze pro zájemce, kteří svoji práci chtějí mít dokonalou do detailů a neváhají si nastudovat detaily správného formátování citací, aby si mohli ověřit, že je vysázený výsledek v pořádku.

¹BP Ing. Radka Pyšného <http://www.fit.vutbr.cz/study/DP/BP.php?id=7848>

Makefile kromě překladu do PDF nabízí i další funkce:

- přejmenování souborů (viz níže),
- počítání normostran,
- spuštění vlny pro doplnění nezlomitelných mezer,
- sbalení výsledku pro odeslání vedoucímu ke kontrole (zkontrolujte, zda sbalí všechny Vámi přidané soubory, a případně doplňte).

Nezapomeňte, že vlna neřeší všechny nezlomitelné mezery. Vždy je třeba manuální kontrola, zda na konci řádku nezůstalo něco nevhodného – viz Internetová jazyková příručka².

Pozor na číslování stránek! Pokud má obsah 2 strany a na 2. jsou jen „Přílohy“ a „Seznam příloh“ (ale žádná příloha tam není), z nějakého důvodu se posune číslování stránek o 1 (obsah „nesedí“). Stejný efekt má, když je na 2. či 3. stránce obsahu jen „Literatura“ a je možné, že tohoto problému lze dosáhnout i jinak. Řešení je několik (od úpravy obsahu, přes nastavení počítadla až po sofistikovanější metody). **Před odevzdáním proto vždy přezkontrolujte číslování stran!**

Doporučený postup práce se šablonou

1. **Zkontrolujte, zda máte aktuální verzi šablony.** Máte-li šablonu z předchozího roku, na stránkách fakulty již může být novější verze šablony s aktualizovanými informacemi, opravenými chybami apod.
2. **Zvolte si jazyk,** ve kterém budete psát svoji technickou zprávu (česky, slovensky nebo anglicky) a svoji volbu konzultujte s vedoucím práce (nebyla-li dohodnuta předem). Pokud Vámi zvoleným jazykem technické zprávy není čeština, nastavte příslušný parametr šablony v souboru `projekt.tex` (např.: `documentclass[english]{fitthesis}`) a přeložte prohlášení a poděkování do angličtiny či slovenštiny.
3. **Přejmenujte soubory.** Po rozbalení je v šabloně soubor `projekt.tex`. Pokud jej přeložíte, vznikne PDF s technickou zprávou pojmenované `projekt.pdf`. Když vedoucímu více studentů pošle `projekt.pdf` ke kontrole, musí je pracně přejmenovávat. Proto je vždy vhodné tento soubor přejmenovat tak, aby obsahoval Váš login a (případně zkrácené) téma práce. Vyhněte se však použití mezer, diakritiky a speciálních znaků. Vhodný název může být např.: „`xlogin00-Cisteni-a-extrakce-textu.tex`“. K přejmenování můžete využít i přiložený Makefile:

```
make rename NAME=xlogin00-Cisteni-a-extrakce-textu
```
4. Vyplňte požadované položky v souboru, který byl původně pojmenován `projekt.tex`, tedy typ, rok (odevzdání), název práce, svoje jméno, ústav (dle zadání), tituly a jméno vedoucího, abstrakt, klíčová slova a další formální náležitosti.
5. Nahraďte obsah souborů s kapitolami práce, literaturou a přílohami obsahem svojí technické zprávy. Jednotlivé přílohy či kapitoly práce může být výhodné uložit do samostatných souborů – rozhodnete-li se pro toto řešení, je doporučeno zachovat konvenci pro názvy souborů, přičemž za číslem bude následovat název kapitoly.

²Internetová jazyková příručka <http://prirucka.ujc.cas.cz/?id=880>

6. Nepotřebujete-li přílohy, zakomentujte příslušnou část v `projekt.tex` a příslušný soubor vyprázdněte či smažte. Nesnažte se prosím vymyslet nějakou neúčelnou přílohu jen proto, aby daný soubor bylo čím naplnit. Vhodnou přílohou může být obsah přiloženého paměťového média.
7. Zadání, které si stáhnete v PDF z IS FIT (odkaz „Zadání pro vložení do práce“ či „Thesis assignment“), uložte do souboru `zadani.pdf` a povolte jeho vložení do práce parametrem šablony v `projekt.tex` (`\documentclass[zadani]{fitthesis}`).
8. Nechcete-li odkazy tisknout barevně (tedy červený obsah – bez konzultace s vedoucím nedoporučuji), budete pro tisk vytvářet druhé PDF s tím, že nastavíte parametr šablony pro tisk: (`\documentclass[zadani,print]{fitthesis}`). Budete-li tisknout barevně, místo `print` použijte parametr `cprint`. Barevné logo se nesmí tisknout černobíle!
9. Vzor desek, do kterých bude práce vyvázána, si vygenerujte v informačním systému fakulty u zadání. Pro disertační práci lze zapnout parametrem v šabloně `cover` (více naleznete v souboru `fitthesis.cls`).
10. Nezapomeňte, že zdrojové soubory i (obě verze) PDF musíte odevzdat na CD či jiném médiu přiloženém k technické zprávě.

Obsah práce se generuje standardním příkazem `\tableofcontents` (zahrnut v šabloně). Přílohy jsou v něm uvedeny úmyslně.

Pokyny pro oboustranný tisk

- **Oboustranný tisk je doporučeno konzultovat s vedoucím práce.**
- Je-li práce tištěna oboustranně a její tloušťka je menší než tloušťka desek, nevypadá to dobře.
- Zapíná se parametrem šablony: `\documentclass[twoside]{fitthesis}`
- Po vytištění oboustranného listu zkontrolujte, zda je při prosvícení sazební obrazec na obou stranách na stejné pozici. Méně kvalitní tiskárny s duplexní jednotkou mají často posun o 1–3 mm. Toto může být u některých tiskáren řešitelné tak, že vytisknete nejprve liché stránky, pak je dáte do stejného zásobníku a vytisknete sudé.
- Za titulním listem, obsahem, literaturou, úvodním listem příloh, seznamem příloh a případnými dalšími seznamy je třeba nechat volnou stránku, aby následující část začínala na liché stránce (`\cleardoublepage`).
- Konečný výsledek je nutné pečlivě přezkontrolovat.

Styl odstavců

Odstavce se zarovnávají do bloku a pro jejich formátování existuje více metod. U papírové literatury je častá metoda s použitím odstavcové zarážky, kdy se u jednotlivých odstavců textu odsazuje první řádek odstavce asi o jeden až dva čtverčíky (vždy o stejnou, předem zvolenou hodnotu), tedy přibližně o dvě šířky velkého písmene M základního textu. Poslední řádek předchozího odstavce a první řádek následujícího odstavce se v takovém případě

neoddělují svislou mezerou. Proklad mezi těmito řádky je stejný jako proklad mezi řádky uvnitř odstavce. [?]

Další metodou je odsazení odstavců, které je časté u elektronické sazby textů. První řádek odstavce se při této metodě neodsazuje a mezi odstavce se vkládá vertikální mezera o velikosti 1/2 řádku. Obě metody lze v kvalifikační práci použít, nicméně často je vhodnější druhá z uvedených metod. Metody není vhodné kombinovat.

Jeden z výše uvedených způsobů je v šabloně nastaven jako výchozí, druhý můžete zvolit parametrem šablony „odsaz“.

Užitečné nástroje

Následující seznam není výčtem všech využitelných nástrojů. Máte-li vyzkoušený osvědčený nástroj, neváhejte jej využít. Pokud však nevíte, který nástroj si zvolit, můžete zvážit některý z následujících:

MikTeX L^AT_EX pro Windows – distribuce s jednoduchou instalací a vynikající automatizací stahování balíčků. MikTeX obsahuje i vlastní editor, ale spíše doporučuji TeXstudio.

TeXstudio Přenositelné opensource GUI pro L^AT_EX. Ctrl+klik umožňuje přepínat mezi zdrojovým textem a PDF. Má integrovanou kontrolu pravopisu³, zvýraznění syntaxe apod. Pro jeho využití je nejprve potřeba nainstalovat MikTeX případně jinou L^AT_EXovou distribuci.

WinEdt Ve Windows je dobrá kombinace WinEdt + MiKTeX. WinEdt je GUI pro Windows, pro jehož využití je nejprve potřeba nainstalovat **MikTeX** či **TeX Live**.

Kile Editor pro desktopové prostředí KDE (Linux). Umožňuje živé zobrazení náhledu. Pro jeho využití je potřeba mít nainstalovaný **TeX Live** a Okular.

JabRef Pěkný a jednoduchý program v Javě pro správu souborů s bibliografií (literaturou). Není potřeba se nic učit – poskytuje jednoduché okno a formulář pro editaci položek.

InkScape Přenositelný opensource editor vektorové grafiky (SVG i PDF). Vynikající nástroj pro tvorbu obrázků do odborného textu. Jeho ovládnutí je obtížnější, ale výsledky stojí za to.

GIT Vynikající pro týmovou spolupráci na projektech, ale může výrazně pomoci i jednomu autorovi. Umožňuje jednoduché verzování, zálohování a přenášení mezi více počítači.

Overleaf Online nástroj pro L^AT_EX. Přímě zobrazuje náhled a umožňuje jednoduchou spolupráci (vedoucí může průběžně sledovat psaní práce), vyhledávání ve zdrojovém textu kliknutím do PDF, kontrolu pravopisu apod. Zdarma jej však lze využít pouze s určitými omezeními (někomu stačí na disertaci, jiný na ně může narazit i při psaní bakalářské práce) a pro dlouhé texty je pomalejší. Pro vedoucí má FIT licenci a v případě, že student narazí na omezení, je s pomocí vedoucího situace řešitelná.

Pozn.: Overleaf nepoužívá Makefile v šabloně – aby překlad fungoval, je nutné kliknout pravým tlačítkem na `projekt.tex` a zvolit „Set as Main File“.

³Českou kontrolu pravopisu lze doinstalovat z <https://extensions.openoffice.org/de/project/czech-dictionary-pack-ceske-slovniky-cs-cz>

Príloha B

Psaní anglického textu

Tato příloha je převzata ze stránek doc. Černockého [?].

Spousta lidí píše zprávy k projektům anglicky (a to je dobře!), ale dělá v nich spoustu zbytečných chyb (a to je špatně). Nejsem angličtinář, ale tento jazyk už nějakých pár let používám k psaní, čtení i komunikaci – tato příloha obsahuje pár důležitých věcí. Pokud chcete napsat práci nebo článek opravdu 100 % dobře, nezbude Vám než si najmout rodilého mluvčího (a to by měl by být trochu technicky zdatný a aspoň trochu rozumět tomu, co píšete, ať to neskončí ještě hůř ...).

Obecně

- Předtím, než budete sami něco psát, si přečtěte pár anglických technických článků a zkuste si zapamatovat a získat „obecný pocit“, jak se to píše.
- Používejte vždy korektor pravopisu – zabudovaný ve Wordu, nebo v OpenOffice, pokud děláte na Linuxu, tak ISPELL a další (většina editorů pro L^AT_EX má již kontrolu pravopisu integrovanou).
- Používejte korektor gramatiky. Nevím, jestli je nějaký dostupný na Linuxu, ale ten ve Wordu celkem slušně funguje a pokud Vám něco zelené podtrhne, je tam většinou opravdu chyba. Můžete do něj nakopírovat i zdrojový text pro L^AT_EX, opravit, a pak uložit opět jako čistý text. Pokud používáte vim, je tam zabudovaný také a zvládne jak překlepy, tak základní gramatiku. V dokumentu `diplomka.tex` na první řádek napište:

```
% vim:spelllang=en_us:spell
```

(případně `en_gb` pro OED angličtinu) *Poznámka editora:* Existuje i velmi dobrý online nástroj Grammarly¹, který je v základní verzi zdarma.

- Online slovníky jsou dobré, ale nepoužívejte je slepě. Většinou dají více variant a ne každá je správně.

¹<https://www.grammarly.com/>

- Na vyhledávání a zjištění, co bude asi správné, můžete použít Google. Např.: nevíte, jak se řekne „výhoda tohoto přístupu“. Slovník na seznam.cz dá asi 10 variant. Napište je postupně do vyhledávání na googlu:

```
"advantage of this approach" 1100000 hits
"privilege of this approach" 6 hits
"facility of this approach" 16 hits
```

Neříkám, že je to 100 % správně, ale je to určité vodítko. Toto se dá použít i na dohledání správných spojek (třeba „among two cases“ nebo „between two cases“?)

SVOMPT a shoda

Struktura anglické věty je SVOPMT: SUBJECT VERB OBJECT MANNER PLACE TIME a přes to nejede vlak! Není volná jako v češtině. Jinak to je maximálně v nějaké divadelní hře, kde je potřeba něco zdůraznit. Hlavně podmět tam musí vždycky být, na to se často zapomíná, protože v CZ/SK může být zamlčený nebo nevyjádřený. SVOMPT platí i ve vedlejších větách!

```
BAD: We have shown that is faster than the other function.
GOOD: We have shown that it is faster than the other function.
```

Shoda podmětu s přísudkem – zní to šíleně, ale dělá se v tom spousta chyb.

```
he has
the users have
people were
```

Členy

Členy v angličtině jsou noční můra a téměř nikdo z nás je nedává dobře. Základní pravidlo je, že když je něco určitého, musí předtím být „the“. Členy musí být určité u těchto spojení:

```
the first, the second, ...
the last
the most (třetí stupeň přídavných jmen a příslovčí) ...
the whole
the following
the figure, the table.
the left, the right - on the left pannel, from the left to the right ...
```

Naopak člen NESMÍ být, pokud používáte přesné označení obrázku, kapitoly, atd.

```
in Figure 3.2
in Chapter 7
in Table 6.4
```


Pozor na „a“ vs. „an“, řídí se to podle výslovnosti a ne podle toho, jak je slovo napsané, takže:

```
an HMM
an XML
a universal model
a user
```

Slovesa

Pozor na trpné tvary sloves – u pravidelných je to většinou bez problémů, u nepravidelných často špatně, typicky

```
packet was sent (ne send)
approach was chosen (ne choosed)
```

... většinou to opraví korektor pravopisu, ale někdy ne.

Pozor na časy, občas je v nich pěkný nepořádek. Pokud něco nějak obecně je, přítomný čas. Pokud jste něco udělali, minulý. Pokud to dalo nějaký výsledek a ten výsledek teď existuje a třeba ho nějak diskutujete, přítomný. Nepoužívejte příliš složité časy jako je předpřítomný a vůbec ne předminulý pokud nevíte přesně, co děláte.

```
JFA is a technique that works for everyone in speaker recognition.
We implemented it according to Kenny's recipe in \cite{Kenny}.
12000 segments from NIST SRE 2006 were processed. When compared
with a GMM baseline, the results are completely bad.
```

Délka vět a struktura

- Pište kratší věty a souvětí, pokud máte něco na 5 řádku, většinou se to nedá číst.
- Strukturujte věty pomocí čárek (více než v češtině!), hlavně po úvodu věty, po kterém začíná vlastní věta. Někdy se dává čárka i před „and“ (na rozdíl od češtiny)

```
In this chapter, we will investigate ...
The first technique did not work, the second did not work as well,
and the third one also did not work.
```

Specifika technického textu

Píšete technický text, proto nepoužívejte zkratky

```
he's
gonna
Petr's working on ...
```

a podobně. Jediné, které je tolerované, je „doesn't“, ale neuděláte chybu, když napíšete „does not“.

V technických textech se spíš používá trpný rod než činný:

BAD: In this chapter, I describe used programming languages.

GOOD: In this chapter, used programming languages are described.

Pokud už činný použijete, dává se v technických textech spíše „we“, i když na práci děláte sami. „I“, „my“, atd. se používají pouze tam, kde jde o to zdůraznit, že jde o Vaši osobu, tedy třeba v závěru nebo v popisu „original claims“ v disertaci.

Časté chyby ve slovech

- Pozor na jeho/její, není to it's, ale its
- Obrázek není picture, ale figure.
- Spojka „než“ je „than“, ne „then“ – bigger than this, smaller than this ... hrozně častá chyba! „Then“ je pak, potom.

Príloha C

Checklist

Tento checklist byl převzat ze šablony pro kvalifikační práce, která je k dispozici na blogu prof. Herouta [?], který s laskavým dovolením využil nápadu dr. Szökeho¹.

Velká bezpečnost letecké dopravy stojí z části na tom, že lidé kolem letadel mají **checklisty** na úplně každý, třeba rutinní a dobře zažitý, postup. Jako pilot strpí to, že bude trochu za blbce a opravdu tužičkou do seznamu úkonů odškrtná dokonale zvládnuté akce, vytiskněte si a odškrtejte před odevzdáním diplomky i vy tento checklist a vyhněte se tak častým chybám, které by mohly mít až fatální následky na výsledné hodnocení Vaší práce.

Struktura

- ☐ Už ze samotných názvů a struktury kapitol je patrné, že bylo splněno zadání.
- ☐ V textu se nevyskytuje kapitola, která by měla méně než čtyři strany (kromě úvodu a závěru). Pokud ano, radil(a) jsem se o tom s vedoucím a ten to schválil.

Obrázky a grafy

- ☐ Všechny obrázky a tabulky byly zkontrolovány a jsou poblíž místa, odkud jsou z textu odkazovány, takže nebude problém je najít.
- ☐ Všechny obrázky a tabulky mají takový popisek, že celý obrázek dává smysl sám o sobě, bez čtení dalšího textu. Vůbec nevadí, když má popisek několik řádků.
- ☐ Pokud je obrázek převzatý, tak je to v popisku zmíněno: „Převzato z [X].“
- ☐ Písmenka ve všech obrázcích používají font podobné velikosti, jako je okolní text (ani výrazně větší, ani výrazně menší).
- ☐ Grafy a schémata jsou vektorově (tj. v PDF).
- ☐ Snímky obrazovky nepoužívají ztrátovou kompresi (jsou v PNG).
- ☐ Všechny obrázky jsou odkázány z textu.
- ☐ Grafy mají popsané osy (název osy, jednotky, hodnoty) a podle potřeby mřížku.

¹<http://blog.igor.szoke.cz/2017/04/predstartovni-priprava-letu-neni.html>

Rovnice

- ☐ Identifikátory a jejich indexy v rovnicích jsou jednopísmenné (kromě nečastých zvláštních případů jako t_{\max}).
- ☐ Rovnice jsou číslovány.
- ☐ Za (nebo vzácně před) rovnicí jsou vysvětleny všechny proměnné a funkce, které zatím vysvětleny nebyly.

Citace

- ☐ **Všechny použité zdroje jsou citovány.**
- ☐ Adresy URL odkazující na služby, projekty, zdroje, github apod. jsou odkazovány pomocí `\footnote{\url{...}}`.
- ☐ Všechny citace používají správné typy.
- ☐ Citace mají autora, název, vydavatele (název konference), rok vydání. Když některá nemá, je to dobře zdůvodněný zvláštní případ a vedoucí to odsouhlasil.

Typografie

- ☐ Žádný řádek nepřetéká přes pravý okraj.
- ☐ Na konci řádku nikde není jednopísmenná předložka (správí to nedělitelná mezera \sim).
- ☐ Číslo obrázku, tabulky, rovnice, citace není nikde první na novém řádku (správí to nedělitelná mezera \sim).
- ☐ Před číselným odkazem na poznámku pod čarou nikde není mezera (to jest vždy takto², nikoliv takto ³).

Jazyk

- ☐ Použil jsem kontrolu pravopisu a v textu nikde nejsou překlipy.
- ☐ Nechal jsem si text přečíst od (alespoň) jednoho dalšího člověka, který umí dobře česky / anglicky / slovensky.
- ☐ V práci psané česky nebo slovensky abstrakt zkontroloval někdo, kdo umí opravdu dobře anglicky.
- ☐ V textu se nikde nepoužívá druhá mluvnická osoba (vy/ty).
- ☐ Když se v textu vyskytuje první mluvnická osoba (já, my), vždy se popisuje subjektivní záležitost (*rozhodl jsem se, navrhl jsem, zaměřil jsem se na, zjistil jsem* apod.).
- ☐ V textu se nikde nepoužívají hovorové výrazy.
- ☐ V českém či slovenském textu se zbytečně nepoužívají anglické výrazy, které mají ustálené české překlady. Např. slovo *defaultní* se nahradí např. slovem *implicitní* nebo *výchozí*.

²příklad poznámky pod čarou

³jiný příklad poznámky pod čarou

Výsledek na datovém médiu, tj. software

- ☐ Mám připravené nepřepisovatelné datové médium
 - CD-R,
 - DVD-R,
 - DVD+R ve formátu ISO9660 (s rozšířením RockRidge a/nebo Joliet) nebo UDF,
 - paměťová karta SD (Secure Digital) ve formátu FAT32 nebo exFAT s nastavenou ochranou proti přepisu.
 - ☐ Pokud je výsledek online (služba, aplikace, ...), URL je viditelně v úvodu a závěru, aby bylo jasné, kde výsledek hledat.
 - ☐ Na médiu nechybí povinné:
 - zdrojové kódy (např. Matlab, C/C++, Python, ...)
 - knihovny potřebné pro překlad,
 - přeložené řešení,
 - PDF s technickou zprávou (je-li pro tisk 2. verze, tak obě),
 - zdrojový kód zprávy (L^AT_EX),
- a případně volitelně po dohodě s vedoucím práce
- relevantní (např. testovací) data,
 - demonstrační video,
 - PDF plakátku,
 - ...
- ☐ Zdrojové kódy jsou refaktorovány, komentovány a označeny hlavičkou s autorstvím, takže se v nich snadno vyzná i někdo další, než sám autor.
- ☐ Jakákoliv převzatá část zdrojového kódu je řádně citována – tedy označena úvodním a v případě převzetí více řádků i ukončovacím komentářem. Komentář obsahuje vše, co vyžaduje licence uvedená na webu (vždy je nutné se ji pokusit najít – např. Stack Overflow⁴ má striktní pravidla pro citace).

Odevzdání

- ☐ Chci práci (na max. 3 roky) utajit? Pokud ano, nejpozději měsíc před termínem odevzdání práce si podám žádost (v IS), ke které přiložím případné stanovisko firmy, jejíž duševní vlastnictví je třeba chránit.
- ☐ Mám splněný minimální počet normostran textu (lze spočítat pomocí Makefile a odhadem přičíst obrázky). Pokud jsem těsně pod minimem, konzultoval(a) jsem to s vedoucím.
- ☐ Pokud chci tisknout oboustranně, konzultoval(a) jsem to s vedoucím a mám správně nastavenou šablonu. Kapitoly začínají na liché stránce.

⁴<https://stackoverflow.blog/2009/06/25/attribution-required/>

- ☐ Technickou zprávu mám v deskách z knihařství (min. 1 výtisk, při utajení oba).
- ☐ Za titulním listem práce je zadání (tzn. mám jej stažené z IS a vložené do šablony).
- ☐ V IS jsou abstrakty a klíčová slova.
- ☐ V IS je PDF práce (s klikatelnými odkazy).
- ☐ Oba výtisky práce jsou podepsané.
- ☐ V jednom (při utajení obou) výtisku práce je paměťové médium, na kterém je fixkou napsaný login (fixku na CD lze zapůjčit v knihovně, na Studijním oddělení nebo až při odevzdání).

Príloha D

L^AT_EX pro začátečníky

V této kapitole jsou uvedeny některé často využívané balíčky a příkazy pro L^AT_EX, které mohou být při tvorbě práce potřeba.

Užitečné balíčky

Studenti při sazbě textu často řeší stejné problémy. Některé z nich lze vyřešit následujícími balíčky pro L^AT_EX:

- `amsmath` – rozšířené možnosti sazby rovnic,
- `float`, `afterpage`, `placeins` – úprava umístění obrázků/tabulek (specifikátor H),
- `fancyvrb`, `alltt` – úpravy vlastností prostředí Verbatim,
- `makecell` – rozšíření možností tabulek,
- `pdflscape`, `rotating` – natočení stránky o 90 stupňů (pro obrázek či tabulku),
- `hyphenat` – úpravy dělení slov,
- `picture`, `epic`, `eepic` – přímé kreslení obrázků.

Některé balíčky jsou využity přímo v šabloně (v dolní části souboru `fitthesis.cls`). Nahlédnutí do jejich dokumentace může být rovněž velmi užitečné.

Sloupec tabulky zarovnaný vlevo s pevnou šířkou je v šabloně definovaný „L“ (používá se jako „p“).

Pro odkazování v rámci textu použijte příkaz `\ref{navesti}`. Podle umístění návěští se bude jednat o číslo kapitoly, podkapitoly, obrázku, tabulky nebo podobného číslovaného prvku). Pokud chcete odkázat stránku práce, použijte příkaz `\pageref{navesti}`. Pro citaci literárního odkazu `\cite{identifikator}`. Pro odkazy na rovnice lze použít příkaz `\eqref{navesti}`.

Znak – (pomlčka) se V L^AT_EXu vkládá jako dvě mínus za sebou: --.

Často využívané příkazy pro L^AT_EX

Doporučuji nahlédnout do zdrojového textu této podkapitoly a podívat se, jak jsou následující ukázky vysázeny. Ve zdrojovém textu jsou i pomocné komentáře.

Příklad tabulky:

Tabulka D.1: Tabulka hodnocení

Jméno		
Jméno	Příjmení	Hodnocení
Jan	Novák	7.5
Petr	Novák	2

Příklad rovnice:

$$\cos^3 \theta = \frac{1}{4} \cos \theta + \frac{3}{4} \cos 3\theta \quad (\text{D.1})$$

a dvou horizontálně zarovnaných rovnic:

$$3x = 6y + 12 \quad (\text{D.2})$$

$$x = 2y + 4 \quad (\text{D.3})$$

Pokud je třeba rovnici citovat v textu, lze použít příkaz `eqref`. Například na rovnici výše lze odkázat (D.1). Pokud chcete srovnat číslo rovnic u soustavy, lze použít prostředí `split`:

$$\begin{aligned} 3x &= 6y + 12 \\ x &= 2y + 4 \end{aligned} \quad (\text{D.4})$$

Matematické symboly (α) a výrazy lze umístit i do textu $\cos \pi = -1$ a mohou být i v poznámce pod čarou¹.

Obrázek D.1 ukazuje široký obrázek složený z více menších obrázků. Klasický rastrový obrázek se vkládá tak, jak je vidět na obrázku D.2.



Obr. D.1: **Široký obrázek.** Obrázek může být složen z více menších obrázků. Chcete-li se na tyto dílčí obrázky odkazovat z textu, využijte balíček `subcaption`.

Další často využívané příkazy naleznete ve zdrojovém textu ukázkového obsahu této šablony.

¹Vzorec v poznámce pod čarou: $\cos \pi = -1$



Obr. D.2: Dobrý text je špatným textem, který byl několikrát přepsán. Nebojte se prostě něčím začít.